UNIVERSIDADE D
COIMBRA

Maria Olímpia Machado Dias

SALA-Z

SISTEMA DE GESTÃO DE EVENTOS MUSICAIS, ARTISTAS E SALAS DE ESPETÁCULOS

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pelo Professor Doutor Joel Perdiz Arrais e pelo Engenheiro Jonathan Magalhães apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Julho de 2022

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Maria Olímpia Machado Dias

Sala-Z

Sistema de gestão de eventos musicais, artistas e salas de espetáculos

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pelo Professor Doutor Joel Perdiz Arrais e pelo Engenheiro Jonathan Magalhães e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Julho 2022

Agradecimentos

Eu agradeço ao Professor Joel Perdiz Arrais pelo apoio prestado durante a realização do estágio curricular, pela sua orientação, disponibilidade, sugestões e imenso conhecimento.

À empresa Grama por terem apostado em mim e me acolherem num dos projetos mais enriquecedores que fiz parte. À equipa Sala-Z pelo tempo que dispuseram a ensinar e transmitir todo o vosso conhecimento. Em especial, ao meu orientador do estágio, Jonathan Magalhães, pelo seu suporte, disponibilidade e sugestões durante toda a realização deste estágio.

Também queria agradecer aos meus amigos por toda a diversão, motivação e apoio ao longo do meu percurso académico.

Por fim, a toda a minha família, em especial aos meus pais e irmãs por acreditarem e apoiarem as minhas decisões ao longo da minha vida, tornando possível que hoje escreva este documento.

Abstract

The organization of a music show is a process that requires logistics and previous organization. Since it is necessary to agree on dates, times, budgets, and contracts between all parties involved: event promoters, artists, venue owners, and staff, for example for setting the infrastructure and validation of tickets.

Nowadays, the communication between all entities is done by phone call or email, which makes the processing time consuming and confusing if each entity is in charge of numerous shows. The lack of a digital platform to organize the management of music shows in venues led to the creation of Sala-Z. The platform currently helps venues managers to manage events in their spaces but does not yet allow access from artists.

The project aims to implement features focused on the artist, allowing them to access their events and get all the information associated with it. It proposes to expand the event and venue management functionalities with the addition of a task organizer and an inventory. Additionally, it intends not only to improve communication between artists and venue managers but also, to increase the number of metrics that help both entities make decisions on future events. This document details the entire process done by the student and the enterprise, during the curricular year 2021/2022, to release a new version of Sala-Z with the features described above. The work includes a platform analysis, a study about the direct and indirect competitors, and potential architectural solutions.

Keywords

Microservices, Service, Venue, Artist, Amazon Web Services

Resumo

A organização de um espetáculo musical é um processo que requer logística e organização prévia, dado que é necessário acordar datas, horários, orçamentos e contratos entre todos os sujeitos envolvidos: promotores de eventos, artistas, donos das salas de espetáculo e staff, por exemplo para a montagem de infraestruturas e a validação de bilhetes.

Atualmente, a comunicação entre todas as entidades é realizada via chamada telefónica ou email o que se acaba por tornar o processo moroso e confuso se cada entidade tiver a seu encargo um grande número de espetáculos para gerir. A falta de uma plataforma digital que organize a gestão de espetáculos musicais em salas levou à criação do Sala-Z. A plataforma, atualmente, auxilia gestores de salas de espetáculo na gestão de eventos nas suas salas, no entanto, a plataforma ainda não permite o acesso a artistas.

O projeto tem como objetivo implementar funcionalidades focadas no artista, permitindo que este aceda aos seus eventos obtendo toda a informação associada. Propõe expandir as funcionalidades relativas ao evento e à gestão de salas com a adição de um organizador de tarefas e um inventário. Adicionalmente, pretende não só melhorar a comunicação entre artistas e gestores de salas de espetáculo como aumentar o número de métricas que ajudam ambas as entidades nas tomadas de decisões em eventos futuros. Este documento detalha todo o processo realizado pelo aluno e empresa, durante o ano letivo 2021/2022, para lançar uma nova versão do Sala-Z com as funcionalidades descritas em cima. O trabalho inclui uma análise à plataforma, um estudo sobre concorrentes diretos e indiretos e potenciais soluções a nível arquitetural.

Palavras-Chave

Microsserviços, Serviço, *Venue*, Artista, *Amazon Web Services*

Conteúdo

Acrónimos

AWS *Amazon Web Services*.

DOM *Document Object Model*.

EC2 *Amazon Elastic Compute Cloud*.

ELB *Elastic Load Balancing*.

ES *Epic Story*.

RPI *Remote Procedure Invocation*.

S3 *Amazon Simple Storage Service*.

SNS *Amazon Simple Notification Service*.

SQS *Amazon Simple Queue Service*.

US *User Story*.

ZAP *Zed Attack Proxy*.

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O presente documento foi produzido no seguimento da realização do estágio curricular no âmbito da disciplina Dissertação/Estágio em Engenharia de Software da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

O estágio consiste na continuação do desenvolvimento da plataforma Sala-Z [78], produto interno da empresa Grama sob orientação do Professor Joel Arrais, do Departamento de Engenharia Informática, e do Engenheiro Jonathan Magalhães, da empresa Grama.

1.1 Grama e Sala-Z

Fundada em 2017 e sediada em Coimbra, a empresa Grama fornece diversos serviços para qualquer fase do ciclo de desenvolvimento de um *software*. Possui serviços na gestão de projetos, *design* e desenvolvimento de produtos.

O Sala-Z é um produto interno da empresa Grama, idealizado devido à proximidade dos membros da empresa com o mundo musical. É uma plataforma com o objetivo de facilitar todo o processo de marcação de eventos e comunicação a artistas, promotores de eventos e gestores de salas de espetáculo.

Este produto já conta com duas versões: Sala-Z 1.0 e Sala-Z 2.0. Numa primeira versão, Sala-Z 1.0, o projeto permitia encontrar salas de espetáculo em Portugal através de uma pesquisa filtrada - localização, detalhes técnicos, tipo e capacidade - facilitando a sua procura por parte de artistas e promotores de eventos. Nesta versão também era possível a gestores de salas de espetáculo submeterem as informações das suas salas na plataforma, que seria posteriormente aprovada por administradores no *backoffice* da aplicação.

No ano passado, durante ano letivo 2020/2021, a plataforma sofreu uma reestruturação através da realização de um estágio curricular. A nova versão, Sala-Z 2.0, adiciona funcionalidades que auxiliam na gestão de salas de espetáculo e no planeamento e organização de eventos. Conta com a autenticação de utilizadores, criação de salas de espetáculo e eventos e ainda a edição de toda a informação associada. O *backoffice* foi novamente criado para a gestão das contas de utilizador

pelo administrador do Sala-Z.

1.2 Contexto e Motivação

A proximidade de membros da empresa com o mundo musical permitiu identificar a dificuldade no processo de procura e marcação de salas de espetáculo para a realização de eventos.

O processo engloba a procura de salas num determinado local, se as especificações - capacidade, detalhes técnicos, tipo - se adequam e, por fim, a sua disponibilidade. A falta de resposta e demora na marcação leva à perda de oportunidades ou ao adiamento de eventos.

A plataforma Sala-Z vem colmatar estes problemas, permite a gestores de salas de espetáculo gerir as suas salas, marcar eventos, associar eventos a artistas bem como facilitar toda a comunicação entre as diferentes entidades. Adicionalmente, é possível a visualização de diversas métricas para uma melhor gestão como por exemplo, saber o eventos com mais receitas.

Em Portugal, a inexistência de uma plataforma que reúna todas estas condições foi uma forte motivação para o seu desenvolvimento. Apesar de já contar com duas versões, a última versão, Sala-Z 2.0, ainda necessita de melhorias tais como: a adição de um módulo de gestão de inventário numa sala, um organizador de tarefas associado a cada evento, o acesso a artistas à plataforma para a visualização dos seus eventos e por fim, a visualização de métricas como receitas e o número de atuações associado a um determinado artista.

Com a duração de dois semestres, o estágio curricular surgiu da necessidade da continuação do desenvolvimento da plataforma, a nova versão denomina-se Sala-Z 2.1.

1.3 Objetivos

Numa primeira fase do estágio curricular é necessário fazer um reconhecimento do estado atual da plataforma, o que engloba uma análise da versão do Sala-Z 2.0. De seguida, deve proceder-se à procura de soluções com o levantamento tecnológico, análise da concorrência e dos riscos de desenvolvimento, bem como das estratégias de mitigação. Por fim, é realizado o levantamento de requisitos seguido da especificação da proposta de solução.

Os principais módulos a serem adicionadas com a realização do estágio curricular são:

- Gestão de inventário e de recursos das salas de espetáculo;
- Gestão de comunicação entre artistas, promotores de eventos e gestores de salas de espetáculo;

- Gestão de tarefas relacionadas com cada evento;
- Enriquecimento da base de dados de artistas através da ligação a fontes externas de informação como o *Bandcamp* [48] ou o *Discogs* [57];
- Implementação de métricas de acompanhamento de eventos.

1.4 Estrutura

O documento é dividido em 9 capítulos, para cada um, de seguida, são apresentados os temas abordados:

- O Capítulo 1 apresenta a plataforma Sala-Z, o contexto, a motivação e os objetivos que levaram à execução do estágio curricular;
- O Capítulo 2 descreve o estado atual da plataforma com a análise das funcionalidades existentes e da arquitetura;
- No Capítulo 3 é realizado o levantamento do estado de arte. Inicialmente, com a análise da concorrência seguida da procura de soluções para a adição das novas funcionalidades com o tipo de abordagem que a arquitetura deve seguir. Posteriormente, são analisadas as tecnologias a nível de *frontend* e *backend* a serem utilizadas no desenvolvimento da proposta de solução;
- No Capítulo 4 são comparadas metodologias de desenvolvimento de *software* e tomada a decisão da metodologia que melhor se adequa ao projeto. De seguida, são delineadas as tarefas para cada semestre recorrendo a diagramas de *Gantt* e, posteriormente no final de cada semestre, comparadas com o planeamento real obtido. Por fim, são levantados os riscos associados ao desenvolvimento bem como a criação de planos de mitigação;
- No Capítulo 5 são apresentados os requisitos funcionais, definidos com base em *user stories*, e os requisitos não funcionais, em forma de cenários de atributos de qualidade;
- No Capítulo 6 é apresentada a arquitetura seguindo o modelo C4 [71];
- O Capítulo 7 descreve toda a fase de desenvolvimento com a organização da equipa, das tarefas e funcionalidades desenvolvidas;
- No Capítulo 8 são descritos os testes efetuados para a validação da plataforma aos requisitos funcionais e não funcionais previamente levantados no Capítulo 5;
- Por fim, o presente documento termina com uma conclusão do trabalho no Capítulo 9.

Capítulo 2

Sala-Z

Sendo o âmbito desta tese de mestrado a continuação de um projeto já existente, o presente Capítulo dedica-se à analise do estado atual da plataforma Sala-Z 2.0, desenvolvida por um estágio curricular durante o ano letivo 2021/2022, com a correspondente descrição e estrutura.

Uma vez que o conceito *venue* é referido múltiplas vezes ao longo do documento, é alvo de uma definição. A *venue* é um local que recebe atuações de artistas destinados a uma audiência. Existem diversos tipos de *venues* como teatros, bares, cafés, conservatórios, arenas, pavilhões e estádios, cada um com características específicas como a lotação, sonoridade e o tamanho do palco. Por exemplo, teatros, conservatórios e óperas são espaços fechados onde artistas como orquestras, coros e óperas necessitam de boas condições sonoras pois estes não utilizam amplificadores nas suas atuações e necessitam que a voz e instrumentos sejam projetados de forma harmoniosa. Bares e cafés são espaços de atuação para bandas e artistas com um público alvo de menor dimensão enquanto que pavilhões, arenas e estádios são espaços frequentados por artistas com um público alvo de maior dimensão onde necessitam da montagem de grandes infraestruturas como palco e sistemas de som para o concerto. A *venue* é também um espaço que pode conter múltiplas salas.

2.1 Sala-Z 2.0

De seguida são levantadas todas as funcionalidades desenvolvidas, seguida da análise arquitetural da versão Sala-Z 2.0.

2.1.1 Funcionalidades

Esta secção é dedicada à análise dos requisitos funcionais desenvolvidos na versão Sala-Z 2.0, com principal foco nos requisitos relacionados com as funcionalidades a serem implementadas na realização do presente estágio curricular.

De seguida as funcionalidades são divididas por temas a fim de agrupar a infor-

mação facilitando a sua leitura e análise.

Autenticação e definições pessoais

O tema engloba todos os requisitos de gestão de utilizadores com funcionalidades CRUD (*create*, *read*, *update* e *delete*) de perfis de utilizadores e autenticação destes na plataforma. Bem como o envio de convites a outros utilizadores para a plataforma. Contudo, o registo de utilizadores não se encontra desenvolvido.

Definições das *venues*

As definições de uma *venue* associa a si todos os requisitos envolvidos na sua gestão, permitindo ao utilizador ver e editar toda a informação relativa a uma *venue*. No entanto, a gestão de recursos (itens disponíveis na *venue* para a realização de um evento) como adicionar, mudar, listar e eliminar não foram desenvolvidos.

Gestão de artistas

O tema na versão do Sala-Z 2.0 conta apenas com a adição e listagem de artistas no *backend*, ficando a faltar o desenvolvimento do *frontend*. A edição, remoção e visualização de um artista não se encontra implementado, bem como a procura com filtragem, a ordenação e a exportação da lista de artistas. Este módulo encontra-se numa fase inicial de desenvolvimento e por isso é uma das adições a realizar no lançamento do Sala-Z 2.1.

Gestão de eventos

A gestão de eventos engloba todos os requisitos envolvidos na sua gestão, como por exemplo, a visualização, edição, filtragem, ordenação e procura de eventos.

O evento é um processo que requer logística e organização prévia, dado que é necessário acordar datas, horários, orçamentos e contratos entre todos os sujeitos envolvidos: promotores de eventos, artistas, donos das *venues* e *staffs*, por exemplo para a montagem de infraestruturas. Os requisitos funcionais relacionados com as tarefas associadas a um evento não se encontram desenvolvidos: a adição, visualização, edição, pesquisa, ordenação e listagem de todas as tarefas, bem como a adição, eliminação e edição de um recurso a um evento e, por último, a sincronização dos calendários. Como a gestão de tarefas é algo muito importante na organização de eventos e não estando este módulo implementado, o Sala-Z 2.1 conta com a sua implementação.

Gestão de recursos

As especificações da *venue* são muito importantes na realização de um evento. Artistas procuram *venues* com determinadas especificações técnicas e, na realização

de um evento, é muito importante que tenham conhecimento dos recursos, itens, que têm à sua disposição ou o que precisam de adquirir. A gestão de inventário permite assim aos responsáveis listar e organizar todos os recursos disponíveis na *venue* para uma melhor gestão e, na contratação de artistas, facilitar a marcação de eventos, com a disponibilização dos recursos.

No tema gestão de recursos, nenhum dos requisitos se encontra desenvolvido desde a listagem de recursos da *venue*, à gestão, filtragem, ordenação, pesquisa e exportação, até à visualização da utilização de recursos no calendário. No Sala-Z 2.1, são adicionadas as funcionalidades relativas à gestão de recursos em *venues*.

Dashboard

O *dashboard* mostra diferentes tipos de métricas como a visualização do evento com mais receitas, o evento com mais entradas, o artista que gerou mais receita, o artista com maior número de eventos, a receita mensal e anual, o número de eventos por mês, e a visualização de eventos no calendário. A visualização do tipo de música que gerou mais receita (com filtros) e o tipo de música com mais eventos associados não se encontram desenvolvidos.

Uma vez que o sistema de gestão de inventário ou gestão de recursos das *venues*, gestão de artistas e gestão de tarefas relacionadas com cada evento, são módulos cruciais para a plataforma, serão adicionados e integrados na nova versão. Para além das funcionalidades referidas, pretende melhorar a comunicação entre artistas, promotores de eventos e gestores de *venues* e a implementação de métricas de acompanhamento de eventos. Por fim, será necessário enriquecer a base de dados de artistas através da ligação a fontes externas de informação como o ***Bandcamp*** [48] ou o ***Discogs*** [57]. O ***Bandcamp*** é uma plataforma que permite aos artistas a divulgação e venda das suas músicas de uma forma autónoma, já a ***Discogs*** é considerada a "maior base de dados musical" [57].

2.1.2 Arquitetura

A plataforma na versão Sala-Z 2.0, de momento, apenas pode ser acedida por perfis de administradores do Sala-Z ou administradores de *venues*. Os administradores do Sala-Z são responsáveis pela gestão do *backoffice* da plataforma com a gestão de perfis de utilizadores e criação de novas *venues*. Cada administrador de uma *venue* após a autenticação, poderá gerir os espaços em que foi associado como administrador por um administrador do Sala-Z, gerindo toda a informação associada. Ainda, o gestor de *venues* conta com uma *dashboard* para a visualização de métricas associadas às suas *venues*.

O *frontend* da aplicação foi desenvolvido numa *single page web application*. O utilizador após a autenticação é redirecionado para o *backoffice* da plataforma caso se trate de um administrador do Sala-Z, caso contrário é redirecionado para o portal da plataforma para a gestão das suas *venues* caso de se trate de um administrador

de uma *venue*. O *frontend* da aplicação de momento comunica através de pedidos *Rest* para o *backend* da aplicação para os diversos *endpoints* mediante o pedido do utilizador.

O *backend* da aplicação segue uma arquitetura baseada em microsserviços, composta por dois serviços desenvolvidos com recurso à *framework Quarkus*. O serviço *Venue Management* engloba todas as funcionalidades referentes à gestão das *venues* enquanto o serviço *Authentication* agrega todas as funcionalidades de gestão de utilizadores e autenticação dos mesmos na plataforma.

O Sala-Z 2.0 interage com os seguintes serviços externos da ***Amazon Web Services (AWS)***: o ***Amazon Cognito*** para a autenticação de utilizadores, máquinas ***Amazon Elastic Compute Cloud (EC2)*** para alojar os diferentes serviços, o ***Amazon Simple Storage Service (S3)*** para armazenar imagens e documentos e por fim o ***Amazon CloudFront*** para a distribuição do *frontend* da aplicação.

Na Figura 2.1 está representada a atual arquitetura do Sala-Z 2.0.

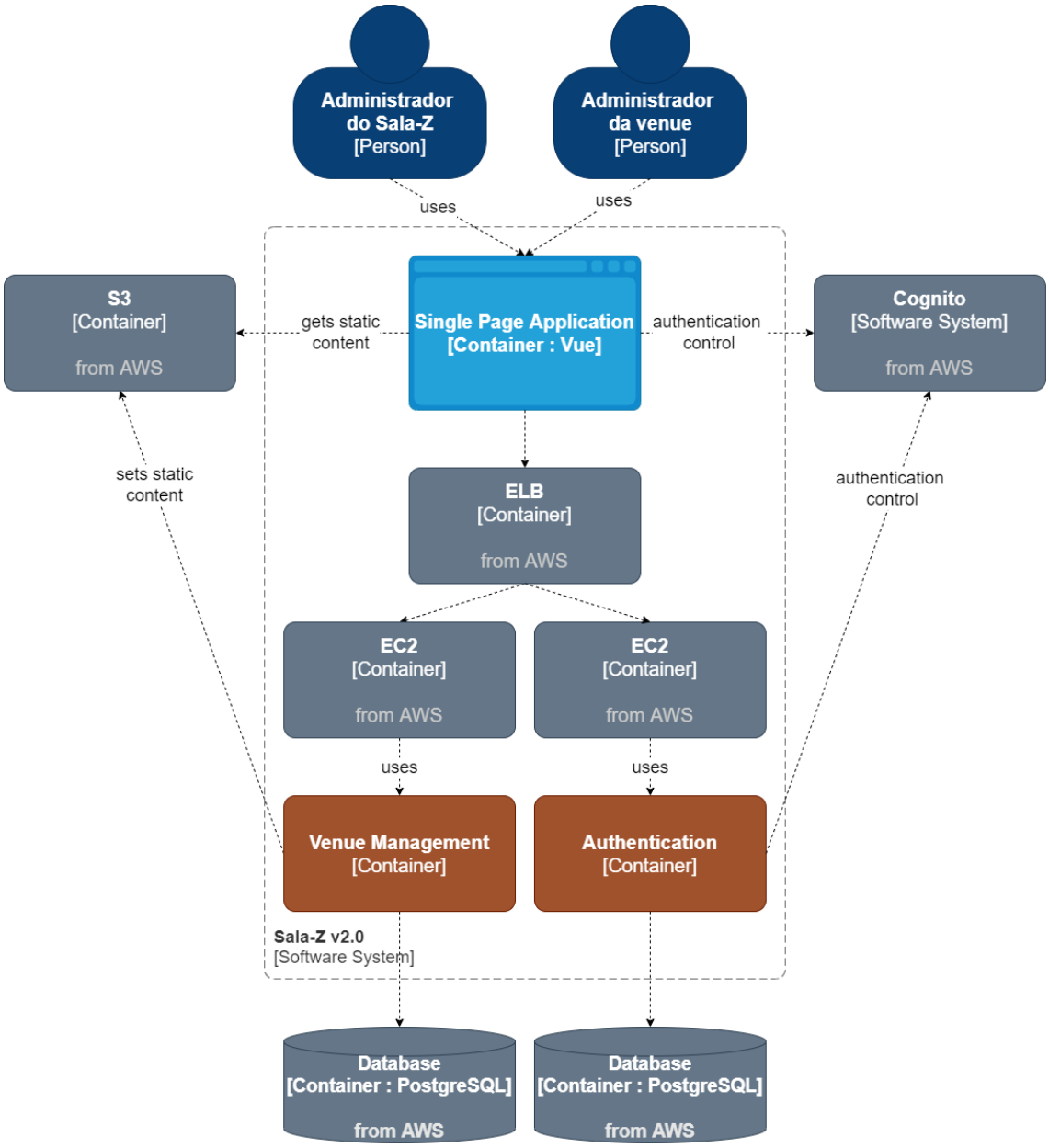

Figura 2.1: Arquitetura da versão Sala-Z 2.0

2.2 Sala-Z 2.1

A nova versão do Sala-Z para além da adição das funcionalidades descritas na Secção 1.4, referente à execução do presente estágio, passa também pela realização paralela de outro estágio curricular. O estágio ocorre no presente ano, também no Departamento de Engenharia Informática da Universidade de Coimbra, em engenharia de software. Foca-se em implementar um sistema para venda de bilhetes dos eventos presentes no Sala-Z. Embora as funcionalidades a serem adicionadas pelos dois estágios, não sejam dependentes, todas as tomadas de decisão e reuniões ao longo do ano letivo serão realizadas em conjunto.

Capítulo 3

Estado de Arte

No presente Capítulo é realizado o levantamento e a análise do estado de arte.

Inicialmente é realizado uma análise de concorrência a fim de encontrar competidores. De seguida, é realizado um estudo sobre potenciais soluções para a reestruturação da plataforma e tecnologias para o desenvolvimento da plataforma a nível de *frontend* e *backend*.

3.1 Análise de Concorrência

Nesta Secção é realizada a análise de concorrência, serão analisadas plataformas que permitam a gestão de inventário e tarefas relacionadas com cada evento, a comunicação entre entidades e que disponibilizem métricas relativos a eventos.

Os concorrentes analisados dividem-se entre concorrentes diretos e indiretos. Os concorrentes diretos referem-se a produtos que tenham as mesmas funcionalidades para o mesmo público-alvo que a nossa solução. Por outro lado, os concorrentes indiretos, apesar de não oferecerem produtos desenvolvidos para o nosso público-alvo, as suas funcionalidades formam uma alternativa que podem ser utilizados por estes.

No final, são analisados dois potenciais concorrentes do Sala-Z em Portugal. Apesar de estes não oferecerem o mesmo produto, caso desenvolvam as mesmas funcionalidades acabam por se tornar concorrentes diretos. Por essa razão e uma vez que são potenciais concorrentes a nível nacional, optou-se pela sua análise.

3.1.1 Concorrentes Diretos

De seguida são analisados os concorrentes diretos da plataforma Sala-Z.

Gigwell

A ***Gigwell*** é uma plataforma que permite a promotores, artistas e gestores de *venues* negociar contratos, gerir a logística associada a um artista, possui um sistema de venda de bilhetes e a monitorização de receitas em tempo real [63].

Gigplanner

O ***Gigplanner*** oferece a artistas e promotores de eventos uma plataforma para a gestão de eventos. Permite tanto gerir as suas tarefas e horário como armazenar informações sobre a localização, *catering*, configurações técnicas e hotéis para partilha [62].

Prism.fm

A plataforma ***Prism.fm*** permite ao utilizador gerir o seu calendário com a possibilidade de partilha com promotores de eventos ou agentes. Integra um calendário onde podem ser adicionados eventos, associar tarefas que podem ser atribuídas a *stakeholders*. A plataforma ainda permite controlar os custos e receitas e ter estimativas de receitas, lucro e *break-even* [76].

Muzeek

O ***Muzeek*** é uma plataforma que permite a criação de eventos direcionados a artistas, promotores de eventos e *venues*. Tem à disposição um calendário que organiza e ajuda na gestão de tarefas, vem facilitar toda a parte de documentação e gestão de pagamentos [73].

Theatron

O ***Theatron*** é um *software as a service*, que permite gerir o staff envolvido em um evento, parte artística, técnica e administrativa. Gerir *venues* independentemente do número de salas e espaços e ainda, planear eventos com calendários associados [84].

3.1.2 Comparações entre concorrentes diretos

Na Tabela 3.1 são comparadas os concorrentes diretos da plataforma Sala-Z 2.1.

	Concorrentes diretos					
	Gigwell	*Gigplanner*	*Prism.fm*	*Muzeek*	*Theatron*	*Sala Z 2.1*
Gestão de inventário	Não	Não	Não	Não	Sim	Sim
Gestão recursos das *venues*	Não	Não	Não	Não	Sim	Sim
Gestão tarefas relacionadas com o evento	Sim	Sim	Sim	Sim	Sim	Sim
Comunicação entre entidades	Sim	Sim	Não	Sim	Não	Sim
Métricas de acompanhamento de eventos	Sim	Não	Sim	Sim	Não	Sim

Tabela 3.1: Comparações entre concorrentes diretos

3.1.3 Concorrentes Indiretos

Como referido anteriormente, serão analisados concorrentes indiretos. Apesar destes concorrentes não disponibilizarem um produto direcionado a artistas e gerentes de *venues*, são utilizados por estes como forma de solucionar problemas que enfrentam.

O ***Trello***, ***Jira***, ***Linear***, ***Asana***, ***Bootcamp*** e ***EasyNote*** são exemplos de organizadores de tarefas. Observa-se que seguem o mesmo estilo de plataforma: é facultada a funcionalidade de criação de projetos onde, dentro dos projetos, podem ser criadas listas de tarefas onde são associadas a tarefas. Visto que todos os *websites* seguem o mesmo estilo, será apenas analisado um concorrente: ***Trello***.

Trello

O ***Trello*** é uma plataforma que permite gerir qualquer tipo de projeto, fluxo de trabalho ou gestão de tarefas [87]. Para diferentes projetos podem ser criados *boards*, a Figura 3.1 representa uma *board*, onde podem ser criadas listas como por exemplo, *Meeting Template*, *January 30th*. Em cada lista é possível associar cartões onde cada um representa uma tarefa. Os cartões podem ser movidos entre listas, ter descrições, prazos de conclusão, *checklists*, anexos e associar um

ou mais responsáveis por cada tarefa.

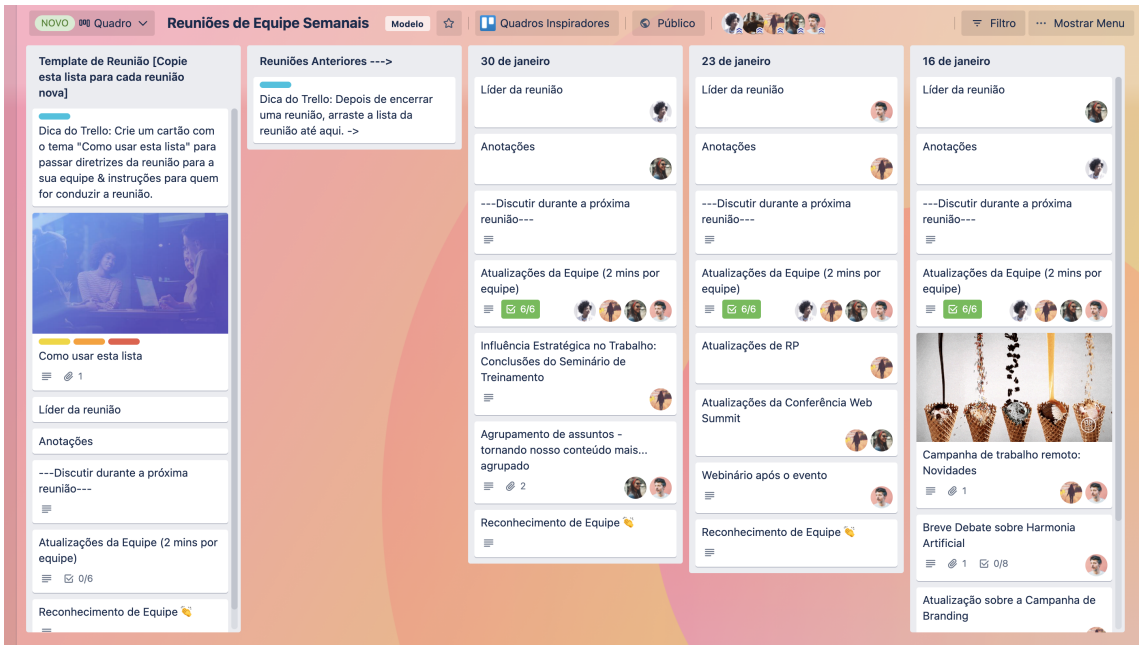

Figura 3.1: Exemplo de uma *board* do *Trello* [83]

3.1.4 Potenciais concorrentes

São analisados agora potenciais concorrentes a nível nacional da plataforma Sala-Z.

Viral Agenda

O **Viral Agenda**, Figura 3.2, é uma plataforma para a visualização de eventos em todo o país desde concertos, exposições a festivais. Apresenta os concertos com filtragem por distrito, concelho e categoria. O utilizador pode visualizar no evento o artista, o promotor, o local e a hora. Tem ainda à sua disposição um registo gratuito em que o utilizador pode guardar os eventos, criar novos, seguir artistas e promotores de eventos [43].

All Venues

O ***All Venues***, Figura 3.3, é um *website* que permite ao utilizador procurar concertos que ocorram tanto em Espanha como em Portugal. Através deste é possível encontrar eventos com pesquisa filtrada e ter uma apresentação sobre o evento. Neste momento o *All Venues* tem *venues* maioritariamente do norte de Portugal e região da Galícia em Espanha [90].

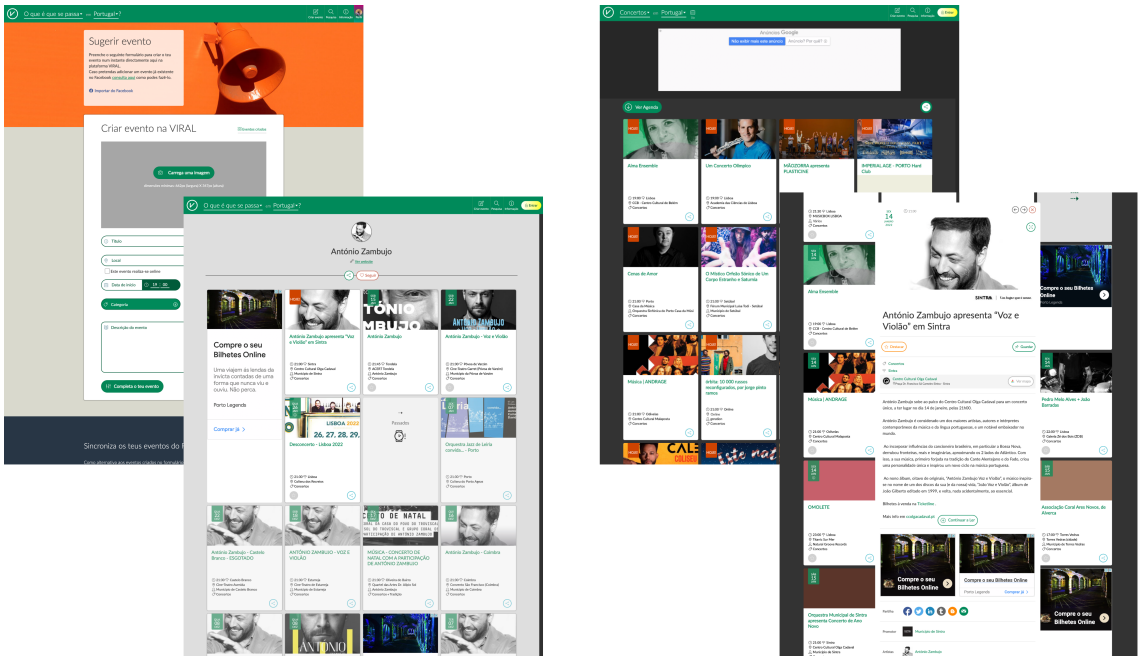

Figura 3.2: *Viral Agenda Website* [43]

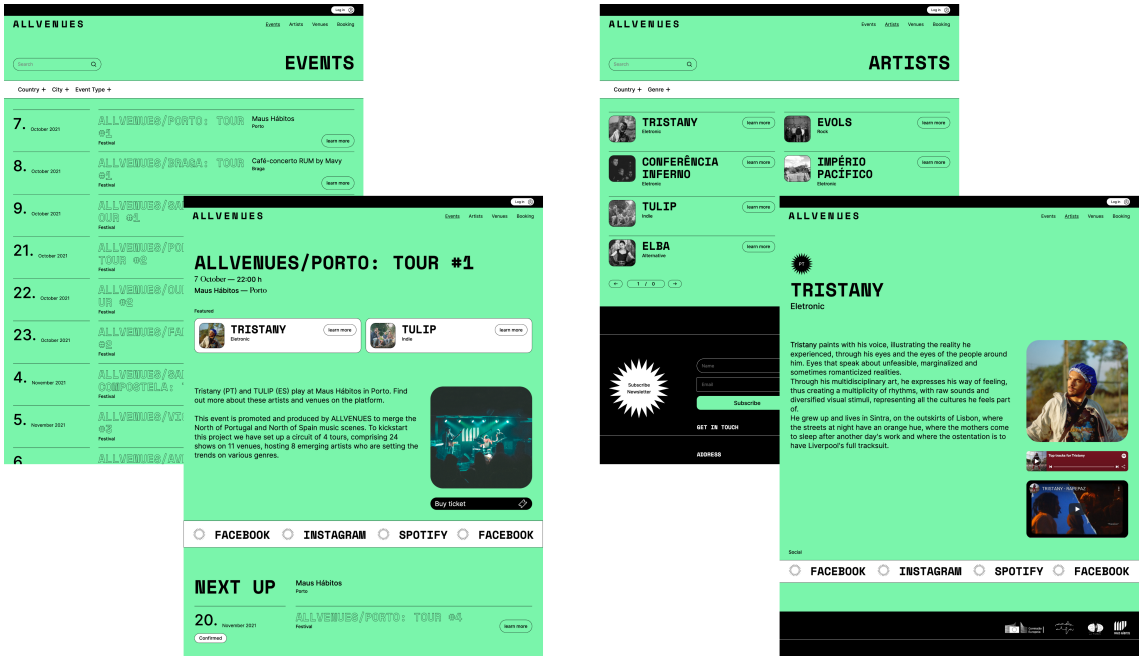

Figura 3.3: *All Venues Website* [90]

3.2 Arquitetura de *Software*

A arquitetura é uma estratégia de *design* que permite visualizar uma solução estruturada que completa todas as necessidades técnicas e estruturais de um *software*. Um diagrama que permite estabelecer a comunicação e coordenação entre diferentes elementos tendo como objetivo otimizar os atributos de qualidade associados a este, sendo guiada por requisitos não funcionais [64].

Sendo o âmbito da tese de mestrado a implementação de novas funcionalidades numa plataforma já existente, deve escolher-se a abordagem utilizada para o desenvolvimento do desenho de proposta de solução. A atual versão do Sala-Z, segue uma arquitetura baseada em dois serviços:

- ***Authentication*** - serviço responsável por gerir a autenticação dos utilizadores e dos seus dados pessoais.
- ***Venue Managment*** - serviço responsável por toda a gestão das *venues* como listagem de eventos, recursos, artistas associados, entre outros.

A implementação das novas funcionalidades passa pela decisão destas serem implementadas nos serviços já existentes como a antiga proposta de solução, seguindo uma abordagem de arquitetura semelhante à monolítica. Ou, a criação de mais serviços repartindo as funcionalidades por estes, seguindo uma arquitetura baseada em microsserviços. Serão analisados os dois tipos de abordagem no desenvolvimento de uma arquitetura para a tomada de decisão do desenho de proposta de solução.

3.2.1 Arquitetura monolítica

A arquitetura monolítica baseia-se numa só aplicação que inclui múltiplos serviços. Estes serviços comunicam com sistemas externos e consumidores através de diferentes interfaces como: *Webservices*, páginas *HTML* ou *REST API* [44].

Como todos os serviços são desenvolvidos num único *codebase*, quando são adicionadas novas funcionalidades, os *developers* têm de garantir que essas mudanças não interferem com outros serviços [91]. Deste modo, quando a aplicação começa a crescer a dificuldade de corrigir bugs, implementar novas funcionalidades e testar acaba por ser mais difícil podendo ocorrer o não cumprimento das *deadlines*. Adicionalmente, outra desvantagem é a aplicação fornecer um único ponto de falha, se a aplicação falhar, todos os serviços vão abaixo [91].

3.2.2 Arquitetura baseada em microsserviços

Grandes empresas como a *Amazon* e a *Netflix*, migraram as suas aplicações e serviços na *cloud* para uma arquitetura baseada em microsserviços [44].

A arquitetura baseia-se em desenvolver serviços de pequenas dimensões como uma única aplicação sendo que, cada serviço corre de forma independente [91], diminuindo o acoplamento e aumentando a coesão nos serviços.

A diminuição do acoplamento deve-se a cada serviço não necessitar de saber informações de outros para o seu funcionamento. Por outro lado, a alta coesão resulta de cada funcionalidade estar bem definida em cada serviço, ou seja, caso seja necessário adicionar uma nova funcionalidade as mudanças serão realizadas apenas num único serviço [13].

Esta arquitetura apresenta diversas vantagens, uma delas é a heterogeneidade, cada serviço pode ser desenvolvido numa linguagem diferente [91] para alcançar o objetivo e *performance* desejados. Como os serviços são independentes, no caso de algum falhar, este não afeta outros [91]. Outra vantagem associada a este tipo de arquitetura é a possibilidade de escalar recursos apenas nos serviços que são necessários e os serviços poderem ser desenvolvidos de forma independente [91].

3.2.3 Tomada de decisão

O desenvolvimento de uma arquitetura baseada em microsserviços possui várias vantagens, permite a utilização separada dos diversos serviços consoante a necessidade do utilizador. A manutenção e adição de serviços não afeta o funcionamento dos restantes, ao contrário de uma arquitetura monolítica que gera mais dependências entre funcionalidades. Ainda, a possibilidade de implementar cada serviço com tecnologias diferentes permite ao aluno selecionar as tecnologias que pretende usar no desenvolvimento.

A arquitetura da versão do Sala-Z 2.0, baseada em microsserviços, contém dois serviços. O serviço ***Authentication*** que contém as funcionalidades de autenticação e gestão de perfis e o serviço ***Venue Management*** com as restantes funcionalidades da plataforma. O serviço ***Venue Management*** já se revela extenso e com a continuação da adição de funcionalidades a este serviço, fará com que este serviço seja mais suscetível a dependências. Por estas razões, as funcionalidades já desenvolvidas e as novas devem ser repartidas por mais serviços mais pequenos, facilitando a manutenção da plataforma.

Desta forma, o desenho da proposta de solução segue uma arquitetura baseada em microsserviços, com a reestruturação de novas funcionalidades e das já implementadas por mais serviços para além dos dois já existentes.

3.3 Arquitetura baseada em Microsserviços

A arquitetura em microsserviços consiste na divisão da arquitetura em serviços.

Os serviços são pequenas unidades, independentes e pouco acopladas que comunicam através de *APIs* [19]. Os microsserviços promovem uma organização de equipas pequenas e independentes e a manutenção de cada serviço não tem impacto no correto funcionamento de outros serviços [19]. Cada serviço resolve um problema e caso seja necessário aumentar a complexidade este pode ser subdividido em pequenos serviços [52].

De seguida são analisados padrões associados a uma arquitetura baseada em microsserviços para serem aplicados na arquitetura da versão Sala-Z 2.1.

Decomposição

Existem estratégias associadas à divisão dos serviços como por exemplo: a decomposição por capacidades de negócio e por subdomínios [33]. A decomposição por capacidades de negócio baseia-se em dividir os serviços em função do que a empresa faz e que gera valor, ou seja, preocupa-se com o que a empresa faz ao invés de como faz [33]. Já a decomposição por subdomínios, baseia-se em dividir o sistema em domínios e subdomínios, de acordo com a análise de processos e fluxo da informação [33].

A decomposição dos diferentes serviços será debatida entre o aluno e orientador da empresa, tendo em conta a necessidade do projeto e que estes sejam mantidos pouco acoplados entre si e coesos.

Deployment

Na arquitetura em microsserviços é possível dar *deploy* de uma aplicação através de um *Multiple Service per Host* ou com um *Single Service per Host* [24].

O *Multiple Service per Host* permite correr múltiplas instâncias de diferentes serviços em apenas um *host*. Esta estratégia permite uma utilização mais eficiente dos recursos [25]. Por outro lado, existem riscos de dependências entre versões uma vez que os serviços correm todos no mesmo *host* [25].

Em *Single Service per Host*, cada serviço corre em um *host* diferente. Permite isolar os diferentes serviços, não há dependências entre versões e permite monitorizar e limitar os recursos de cada serviço [28].

Na versão do Sala-Z 2.0, a abordagem utilizada para o *deployment* dos serviços é a utilização de uma abordagem que parte do padrão *Single Service per Host*, o padrão *Service Instance per VM* [27]. O *Service Instance per VM* baseia-se em correr cada instância de um serviço numa máquina virtual diferente [27]. Esta abordagem apresenta como vantagens como a escalabilidade de recursos, tal como permite se permanecer com a utilização de máquinas virtuais da *Amazon Web Services (AWS)* [27].

No Sala-Z 2.0 cada serviço corre numa máquina virtual diferente, *Amazon Elastic Compute Cloud (EC2)*. O *EC2*, é um serviço da *Amazon* em nuvem que permite aos utilizadores instanciar máquinas virtuais para correr aplicações [59]. O serviço é escalável, permitindo escalar as máquinas virtuais necessárias e pagar pelos os recursos consumidos.

A versão Sala-Z 2.1, continuará com a utilização desta abordagem uma vez que permite a diminuição das dependências e promove a escalabilidade dos recursos.

Armazenamento

No armazenamento, a arquitetura baseada em microsserviços associa a cada serviço uma base de dados [24]. A versão Sala-Z 2.0 conta com a utilização de serviços da *AWS* para a base de dados:

- ***Amazon Simple Storage Service (S3)*** - serviço de armazenamento de dados da *Amazon* na nuvem, permite a criação de diferentes *buckets*, base de dados [5]. No Sala-Z 2.0 existem dois *buckets* na *S3*, um para o armazenamento do conteúdo estático, *frontend* da aplicação, e outro para o armazenamento de conteúdos como imagens, vídeos, e sons.
- ***Amazon Cognito*** - serviço de autenticação de utilizadores da Amazon [54]. Armazena os dados de contas de utilizadores. Acedido pelo serviço ***Authentication***.
- **Base de Dados *PostgreSQL*** - conta com dois *schemas*, cada um acedido por um serviço. Um *schema* guarda os dados relativos às *venues*, outro *schema* os detalhes relativos a utilizadores.

Na adição de novos serviços, será utilizada uma base de dados para cada serviço, assim caso seja necessário repartir os serviços existentes no futuro, a sua repartição será mais fácil de executar.

Testagem

Existem diferentes tipos de testagem. Segundo um artigo publicado [92] pela Universidade de *Wuhan*, os tipos de testagem mais utilizados são: *unit testing*, *integration testing* e *end-to-end testing*.

Unit testing é utilizado para testar e assegurar que pequenas unidades de código como por exemplo funções e serviços funcionam corretamente. O *integration testing* assegura que todos os serviços funcionam em conjunto atingindo determinado objetivo. A testagem *end-to-end* baseia-se na testagem de ações do utilizador a fim de verificar que cumpre o objetivo esperado e se não há quebras de fluxos de interação do utilizador com a aplicação na execução de tarefas.

Após avaliação conjunta entre o aluno e o orientador da empresa foi considerada que a melhor solução seria realizar *unit testing* e *integration testing*. Caso seja necessário completar o plano de testes com outros tipos de testagem será alterado durante o decorrer do desenvolvimento do projeto.

API externa

Existem dois padrões para aceder à aplicação, através de uma *API Gateway* ou utilizando *Backend for Frontend* [24]. *API gateway* oferece um único ponto de entrada à aplicação [22]. Já o *Backend for Frontend* separa o código *backend* do *frontend*, pode ser criado um ou diferentes serviços *backend* para serem consumidos

por *frontend* específico para as diferentes interfaces [10], por exemplo interfaces móveis e *desktop*.

O Sala-Z 2.0 neste momento segue uma abordagem mais próxima do *Backend for Frontend*, apesar de o plano inicial ter sido a implementação de uma *API Gateway*. Contudo, na realização do estágio curricular pretende-se implementar uma *API Gateway* como ponto de acesso aos diferentes serviços.

Comunicação

Os serviços precisam de comunicar entre si para assegurar os pedidos do cliente. Essa comunicação pode ser realizada através de um *Remote Procedure Invocation (RPI)* ou através de *Messaging* [24]. A utilização de *RPI* tem como vantagens a fácil integração, no entanto possui como desvantagens o não suporte de notificações, mecanismos *publish/subscribe*, respostas assíncronas e necessita da localização dos caminhos para instanciar os diferentes serviços [26]. *Messaging* é vantajoso na medida em que o *message broker*, que recebe as mensagens, está sempre disponível e guarda as mensagens até o *consumer* estar disponível, suporta notificações e mecanismos *publish/subscribe* [23]. Contudo, este sistema é mais complexo de implementar do que *RPI* [23].

O Sala-Z 2.0 utiliza *RPI* com mecanismos *REST* para comunicar entre os diferentes serviços, essa abordagem será mantida quando é necessário sincronismo. Porém, quando for necessário enviar notificações, será implementado um *message broker*.

Segurança

Access Token é um padrão utilizado em segurança que garante que utilizadores não autorizados não acedam a informação de terceiros, mantendo os dados confidenciais [21].

A versão Sala-Z 2.0 utiliza o *Amazon Cognito* [54], serviço da *AWS* para garantir a autenticação de utilizadores. Este serviço guarda os utilizadores e a cada pedido de *login*, o utilizador é validado e é retornado um *token* de autenticação. O *token* é um *JSON Web Token*, uma assinatura digital [86] em formato *JSON*, que posteriormente é utilizado no acesso a recursos do *backend*. Esta abordagem será mantida na nova versão da plataforma.

3.4 Tecnologias

Nesta Secção será realizada a análise de tecnologias a utilizar na nova versão do Sala-Z. A análise de diferentes tecnologias deve-se à utilização de uma arquitetura em microsserviços que apresenta como vantagem a utilização de diferentes linguagens nos diferentes serviços. Com isto, faz sentido analisar as tecnologias

para verificar se estas são as adequadas ao desenvolvimento das novas funcionalidades da plataforma.

Numa primeira Secção serão analisadas tecnologias de *frontend* e de seguida tecnologias de *backend*.

As tecnologias utilizadas no Sala-Z 2.0 também serão incluídas nesta comparação. A plataforma está desenvolvida com a linguagem *Vue* no *frontend* e um servidor implementado com a *framework Quarkus* no *backend*. Por conseguinte, ambas as *frameworks* serão inseridas na análise.

3.4.1 *Frontend*

Existem várias *frameworks* para o desenvolvimento do *frontend* de aplicações como por exemplo: ***Angular***, ***React***, ***Vue***, ***Backbone.js*** e ***Ember.js*** [1]. Analisando a relevância na comunidade de todas estas tecnologias Figura 3.4, as mais populares são: ***Angular***, ***React*** e ***Vue***. Todas, apresentam suporte por elementos da empresa.

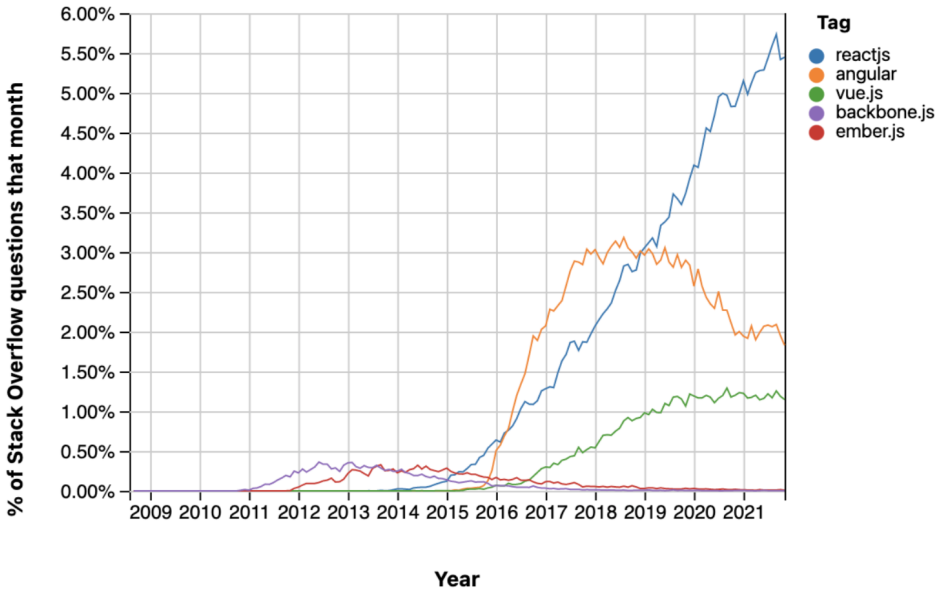

Figura 3.4: *Stack Overflow Trends* [88]

React

O ***React*** é uma biblioteca *javascript* utilizada para construir interfaces [77]. Destaca-se pela seu *Document Object Model (DOM)* virtual que oferece funcionalidades que permitem uma manipulação rápida e fácil do *DOM* [1].

Vue

Vue, *framework* utilizada no desenvolvimento da versão 2.0 do Sala-Z, é uma das *frameworks* que ganhou popularidade ao longo dos anos [1].

O *DOM* é virtual, apresenta uma arquitetura baseada em componentes e uma ligação bidirecional [67]. A ligação bidirecional fornece vantagens como um aumento da performance e melhora a facilidade na atualização de componentes e verificação de alterações de dados, algo muito importante em aplicações em que é necessário saber as alterações em tempo real [67].

Vue em comparação com ***React*** apresenta um menor tamanho nos ficheiros[67]. Em comparação com a *framework* ***Angular***, é uma *framework* simples, boa para remover complexidades que a *framework* ***Angular*** apresenta [67].

Angular

Angular é uma *framework* em *typeScript*. Esta linguagem destaca-se pela capacidade de sincronização em tempo real entre o *model* e a *view*, ou seja, qualquer tipo de mudança que ocorra no *model* passa para a *view*, e qualquer mudança que ocorra na *view* passa para o *model* [56].

3.4.2 Comparação de tecnologias de *frontend*

Na Tabela 3.2 são comparadas as linguagens para o desenvolvimento do *frontend* da plataforma Sala-Z 2.1.

3.4.3 Tomada de decisão

Comparando as três *frameworks* todas possuem vantagens e desvantagens de utilização.

Analisando agora as *frameworks* em termos de popularidade dentro da comunidade, verificando os repositórios do *GitHub*, o ***React*** conta com 180 mil estrelas [31], o ***Angular*** com cerca de 78600 mil [8] e o ***Vue*** com 191 mil [38]. Como ***Angular*** é uma *framework* difícil de aprender, ao contrário do ***Vue*** e de ***React***, é explicado o porquê de este ter uma comunidade menor. ***Vue*** apesar de ser uma *framework* recente já possui uma comunidade maior, no *GitHub*, relativamente ao ***React***.

Uma vez que ***Vue*** é uma linguagem de fácil aprendizagem, tem uma grande comunidade e existe apoio por parte da empresa, o Sala-Z 2.1 utilizará esta linguagem no seu desenvolvimento. Por outro lado, o *frontend* do Sala-Z 2.0 estar desenvolvido com esta *framework* permite manter a homogeneidade do projeto.

	Tecnologias de *Frontend*		
	React	*Vue*	*Angular*
Lançamento	2013	2014	2016
Linguagem	*JavaScript* *TypeScript*	*JavaScript* *TypeScript*	*TypeScript*
Vantagens [56][67]	→ Permite reutilizar componentes; → O *DOM* virtual permite que as operações executem rápido; → Pode ser implementado com outras bibliotecas.	→ Documentação detalhada; → Ligação bidirecional; → Sintaxe simples para iniciantes.	→ Arquitetura baseada em componentes; → Ligação bidirecional; → Aplicações testáveis e reutilizáveis.
Desvantagens [67]	→ Documentação pouco elaborada devido às constantes atualizações.	→ Falta de *plugins*;	→ Difícil de aprender para iniciantes.
Quando utilizar [56]	→ Quando se pretende desenvolver uma interface interativa num curto espaço de tempo devido à capacidade de reutilização de componentes.	→ Quando se pretende criar estruturas de *design* flexíveis.	→ Quando se pretende aumentar a *performance* do *browser* devido à capacidade de rápida *performance* na atualização de componentes com o *two-way data binding*.

Tabela 3.2: Comparação de tecnologias de *frontend*

3.4.4 *Backend*

Spring Boot e ***Express.js*** são *frameworks* populares na comunidade, ***Spring Boot*** conta com 59.1 mil estrelas no *Github* [36], já ***Express.js*** com 55.6 mil [14]. A seleção de ambas as *frameworks* passou pela sugestão por parte da empresa uma vez que ambas as *frameworks* são utilizadas no desenvolvimento de *software*, existindo um grande suporte por parte desta. Além destas, a *framework* ***Quarkus*** também será analisada uma vez que foi utilizado no *backend* da versão Sala-Z 2.0.

Express.js

Express.js é uma *framework web* para *Node.js* que fornece um conjunto de recursos tanto para aplicações *mobile* como para *web*. Utiliza métodos *HTTP* e *middleware* para criar uma API robusta de uma forma fácil e rápida [61].

Spring Boot

Spring Boot é uma ferramenta Java que permite o desenvolvimento de aplicações *standalone* e *production-grade* de forma fácil e rápida utilizando a *framework* ***Spring***. A utilização da *framework* ***Spring*** permite a criação de aplicações com um *setup* e as configurações iniciais mínimas necessárias [49].

Quarkus

Quarkus é uma *framework* Java desenvolvida pela *Red Hat*. É uma *framework* Java *full-stack nativa Kubernetes* feita para máquinas virtuais Java. Focado na otimização de *containers* que permite que se torne uma plataforma para ambientes *serverless*, *cloud* e *Kubernetes* [41].

3.4.5 Comparação de tecnologias de *backend*

Na Tabela 3.3 são comparadas as linguagens para o desenvolvimento do *backend* da plataforma Sala-Z 2.1.

3.4.6 Tomada de decisão

Uma vez que ***Express.js*** apresenta um baixo poder computacional e como ***Quarkus*** é mais rápido do que ***Spring Boot***, opta-se pela continuação da utilização de ***Quarkus*** no *backend*. Como a versão sala-Z 2.0 implementa um servidor com esta *framework*, a utilização novamente desta tecnologia permite manter a homogeneidade do projeto. Ao aluno, aprender apenas uma ferramenta para o *backend* uma vez que este terá que reestruturar os serviços existentes.

	Tecnologias de *Backend*		
	Express.js	*Spring Boot*	*Quarkus*
Lançamento	2010	2002	2018
Linguagem	*JavaScript*	*Java*	*Java*
Vantagens	→ Eventos são *single-threaded* e *non-blocking* I/O - permite que uma *thread* trate de outra tarefa enquanto espera por outra. Consume menos memória, uma vez que só existe uma *thread*. [66] → Grande comunidade (55 600 estrelas no *Github* [14]);	→ É *multi-threathing*, permite que se a *main thread* estiver a ser consumida, outras *threads* possam tratar dos pedidos à aplicação e a lidar com tarefas longas e repetitivas. [29] → Altamente documentado. [37] → Grande comunidade (59.1 mil estrelas no *Github* [36])	→ Tempo de inicialização do *Quarkus* é mais rápido que *Spring Boot*. [37] → Altamente documentado. [37]
Desvantagens	→ Baixo poder computacional, quando recebe uma tarefa de tamanho considerável orientada ao *CPU*, este usa todo o poder computacional para realizar a tarefa. [66]	→ No momento da instalação o *Spring Boot* instala dependências adicionais desnecessárias que ocupam espaço. [37]	→ Pouco suporte da comunidade (9.8mil de estrelas no *Github* [30]).

Tabela 3.3: Comparação de tecnologias de *backend*

Apesar de ***Quarkus*** apresentar uma comunidade mais pequena do que as restantes *frameworks* analisadas, uma vez que é recente, lançado em 2018, esta ferramenta é altamente documentada e, por outro lado, existe um apoio por parte da empresa, uma vez que a versão Sala-Z 2.0 utiliza ***Quarkus***.

Capítulo 4

Planeamento

Nesta secção é descrita a metodologia de trabalho adotada no desenvolvimento do projeto, seguido do planeamento das tarefas para cada um dos semestres do estágio curricular. Por fim, será descrito o planeamento dos testes e levantados os riscos associados ao desenvolvimento da plataforma.

4.1 Metodologia

Os *Software Development Processes* dividem o desenvolvimento do *software* em fases distintas, atividades com o objetivo de melhorar o planeamento e a gestão na produção de *software* [35]. As metodologias de desenvolvimento dividem-se entre metodologias tradicionais e ágeis [69]. De seguida, é dado um exemplo de uma metodologia tradicional e de metodologia ágil e vantagens e desvantagens da utilização de cada uma.

4.1.1 Metodologia Tradicionais

As metodologias tradicionais são baseadas em fases fixas no ciclo de desenvolvimento de um produto, onde não há espaço para mudanças e, o produto, só é entregue ao cliente no final do desenvolvimento.

Um exemplo de uma metodologia tradicional é o modelo *waterfall* que promove um fluxo de desenvolvimento de produto unidirecional [69]. A Figura 4.1, ilustra um exemplo de uma abordagem de desenvolvimento deste modelo dividido nas seguintes etapas - análise de requisitos, *design*, implementação, testagem e manutenção. Cada etapa só é iniciada após a anterior ser terminada, sendo obtido um *output* utilizado como *input* na fase seguinte.

O modelo *waterfall* é vantajoso por ser simples, fácil de usar, cada etapa está bem definida e os resultados e processos estão bem documentados [72]. No entanto possui várias desvantagens [72]. Primeiramente, a entrega ao cliente só é realizada após a última etapa concluída, o que obriga a que os requisitos sejam bem conhecidos e documentados, pois após a fase de análise de requisitos, estes per-

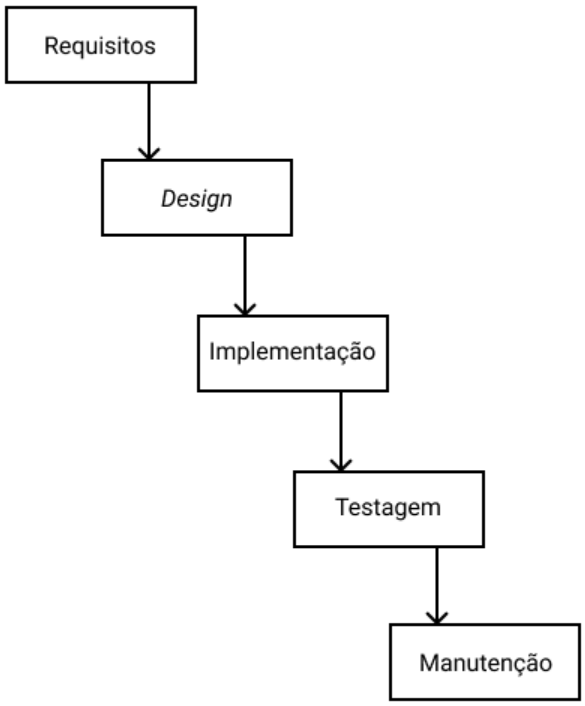

Figura 4.1: Exemplo do modelo *Waterfall*

manecem inalteráveis podendo o produto não atingir as expectativas do cliente. Outra desvantagem associada é o produto ser funcional apenas no final do desenvolvimento, caso o orçamento seja esgotado ao longo do processo o produto obtido ainda não é funcional. Por fim, este não é um bom modelo para projetos, que podem conter uma grande volatilidade dos requisitos em função dos diferentes *stakeholders*.

4.1.2 Metodologias Ágeis

As metodologias ágeis surgiram na década de 1990 da necessidade de substituição de toda a documentação e burocracias impostas pelas metodologias tradicionais [3]. É em 2001, que 17 *developers* que utilizavam metodologias ágeis como *Xtreme Programming*, *Kanban* e *Scrum* no desenvolvimento dos seus projetos, criaram o documento Manifesto Ágil após notarem que estas seguiam os mesmos valores [3]. Assim, os 4 valores em que uma metodologia ágil assenta são:

- As pessoas e as relações são mais importantes que os procedimentos e as ferramentas;
- *Software* a funcionar é mais importante do que ter a documentação completa;
- Colaborar com o cliente é mais importante do que negociar contratos;
- Capacidade de responder a alterações é mais importante que seguir um plano.

Estes valores não pretendem desvalorizar toda a parte de documentação, procedimentos e contratos, mas pretende estabelecer prioridades e introduzir mais flexibilidade ao projeto [3].

O desenvolvimento da versão Sala-Z 2.1, segue uma metodologia ágil baseada em *Scrum*, esta escolha parte da equipa uma vez que a empresa utiliza esta abordagem no desenvolvimento dos seus projetos.

4.1.3 *Scrum*

O *Scrum* é uma metodologia ágil que se baseia na divisão do projeto em *sprints*, Figura 4.2. O *sprint* é um período de tempo em que a equipa de desenvolvimento realiza um conjunto de tarefas e cada *sprint* começa após a *sprint* anterior terminar. No final de cada *sprint* é obtido um novo lançamento do produto funcional, com adição de funcionalidades.

A *scrum team* é dividida em três papéis [39]:

- ***Product owner*** - responsável por identificar as funcionalidades do projeto, construção do *product backlog*, escrever as *user stories* e critérios de priorização;
- ***Scrum master*** - responsável por ajudar e orientar a equipa e o *product owner*, conduzindo ao sucesso do projeto, removendo qualquer impedimento;
- ***Equipa de desenvolvimento*** - responsável por implementar as tarefas listadas no *product backlog*.

O aluno integra a equipa de desenvolvimento em conjunto com o designer responsável pela criação das *mockups*, para as diferentes interfaces da plataforma e o aluno do outro estágio que decorre em paralelo. Embora a criação das *user stories* serem da responsabilidade do *product owner*, o aluno participou na sua criação como forma de aprendizagem.

O *Scrum* possui diversas cerimónias durante a fase de desenvolvimento [50]:

- ***Sprint planning*** - este momento acontece no inicio de cada *sprint* e é presenciado por toda a equipa. Aqui as tarefas são priorizadas e escolhidas aquelas que serão realizadas na presente *sprint*;
- ***Daily scrum*** - reuniões diárias entre os membros de equipa. Cada membro deve referir o que fez, o que espera fazer e se precisa de ajuda para resolver algum problema;
- ***Sprint retrospective*** - toda a equipa deve participar neste evento, onde é verificado o que correu bem e mal durante o *sprint* e como deve ser melhorado nas próximas iterações;
- ***Backlog refinement*** - verifica que o *product backlog* continua atualizado e bem prioritizado;

- ***Sprint review ou demos*** - ocorre no final de cada *sprint* para que a equipa veja o resultado do produto produzido e se este cumpre os objetivos impostos no *sprint*.

Para além das cerimónias, o *Scrum* também possui artefactos [2]:

- ***Product backlog*** - lista de tarefas para o desenvolvimento do trabalho, baseado nos requisitos e funcionalidades a serem implementadas;
- ***Sprint backlog*** - lista de tarefas retiradas do *product backlog* para serem implementadas durante o *sprint*;
- ***Increment*** - é o produto utilizável obtido no final de cada *sprint*.

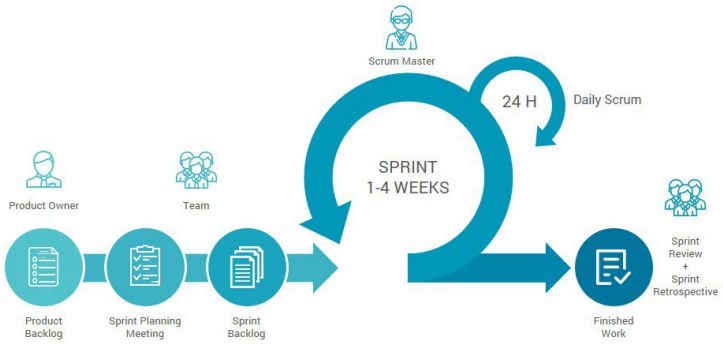

Figura 4.2: Exemplo da metodologia *Scrum* [32]

A utilização de **Scrum** como método de trabalho fornece diversas vantagens [51], como por exemplo:

- permite que equipas de desenvolvimento pequenas desenvolvem um produto de forma rápida e eficiente;
- existe *feedback* constante do cliente permitindo validar se o produto vai de encontro às suas expectativas;
- o esforço é visível em cada *sprint*;
- projetos grandes são divididos em *sprints* mais pequenos o que permite a mudança dos requisitos.

Durante o primeiro semestre, uma vez que o aluno ainda possui unidades curriculares, foram realizadas apenas reuniões semanais entre o aluno, orientador da empresa e equipa de desenvolvimento para saber o estado do trabalho. O segundo semestre segue uma metodologia semelhante ao *Scrum*, com *daily meetings* e *sprints* quinzenais.

4.2 Planeamento Temporal

Na presente secção é apresentado o planeamento do estágio curricular para cada semestre.

Inicialmente foi definida uma lista de tarefas no início de cada semestre com base no plano de trabalhos definido na proposta do estágio curricular e em reuniões entre o orientador da empresa e o aluno. Posteriormente, foi criado um diagrama de *Gantt* com base nas tarefas definidas.

No final de cada semestre, foi construído novamente um diagrama de *Gantt* com a ordem cronológica com que foram executadas as tarefas e efetuada uma comparação relativamente ao plano inicial.

4.2.1 1º Semestre

O primeiro semestre é mais focado na exploração da plataforma Sala-Z, na construção do estado de arte com a análise de concorrentes, nas tecnologias a serem utilizadas no desenho da proposta de solução e na definição dos requisitos e arquitetura.

Para além de todo o processo de escrita de documentação para o relatório intermédio e preparação da defesa intermédia, foi definida uma tarefa para que o aluno explore a plataforma. Assim, o aluno pode preparar-se e conhecer as tecnologias e a plataforma em que irá trabalhar durante o segundo semestre.

Planeamento Temporal Inicial

De seguida, na Figura 4.3 é apresentado o diagrama de *Gantt* com as tarefas definidas para o primeiro semestre. As tarefas estão divididas por subtarefas.

Resultado

A Figura 4.4, ilustra a cronologia real do 1º semestre. Face ao planeamento inicial ocorreram desvios, a expectativas de esforço do aluno associado ao levantamento do estado de arte foram mal calculadas, terminando mais tarde do que o esperado. O número elevado de *user stories* provocou uma demora no processo de criação levando a um atraso da finalização da proposta de solução. Por fim, uma vez que o levantamento do estado de arte e requisitos terminaram mais tarde do que o esperado, impossibilitaram a exploração da plataforma e das tecnologias, o que implica a passagem desta tarefa para o planeamento do 2º semestre.

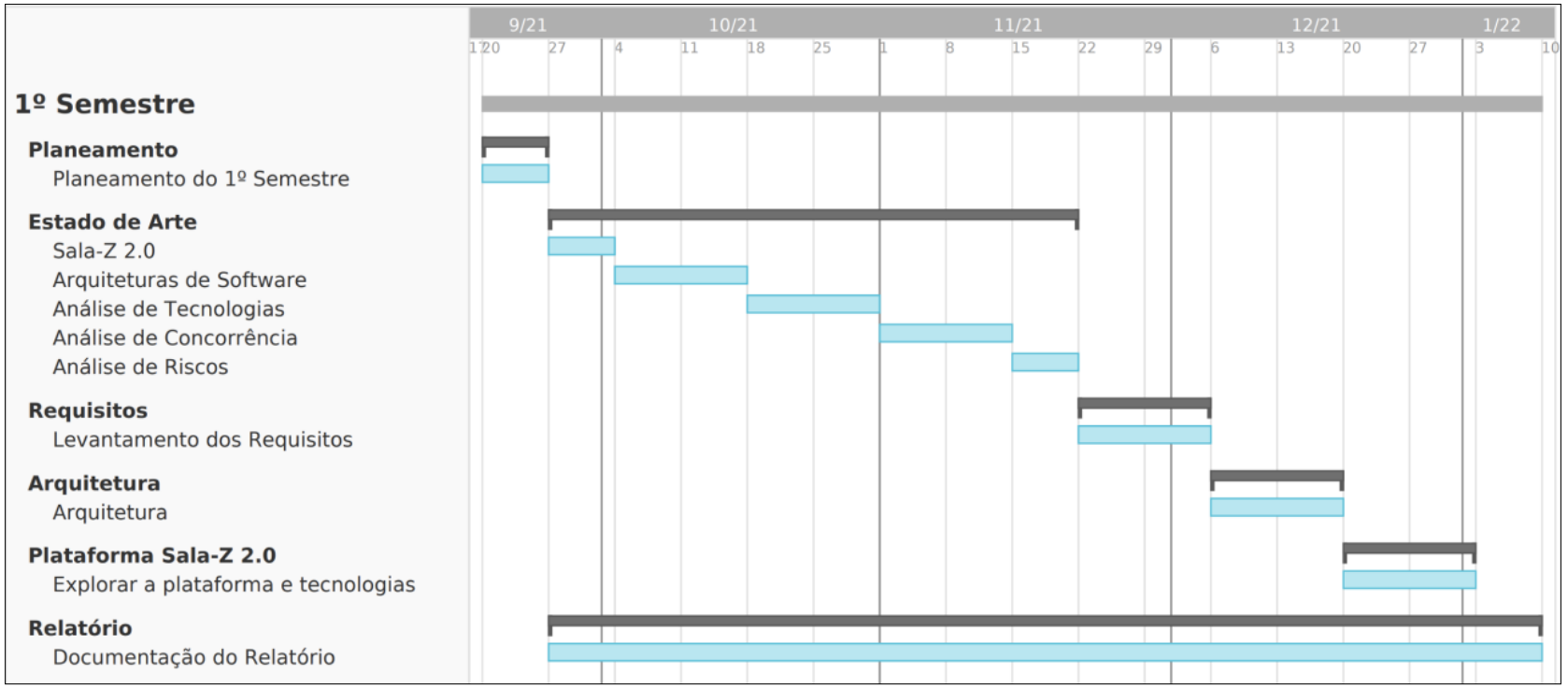

Figura 4.3: Planeamento Temporal 1º Semestre

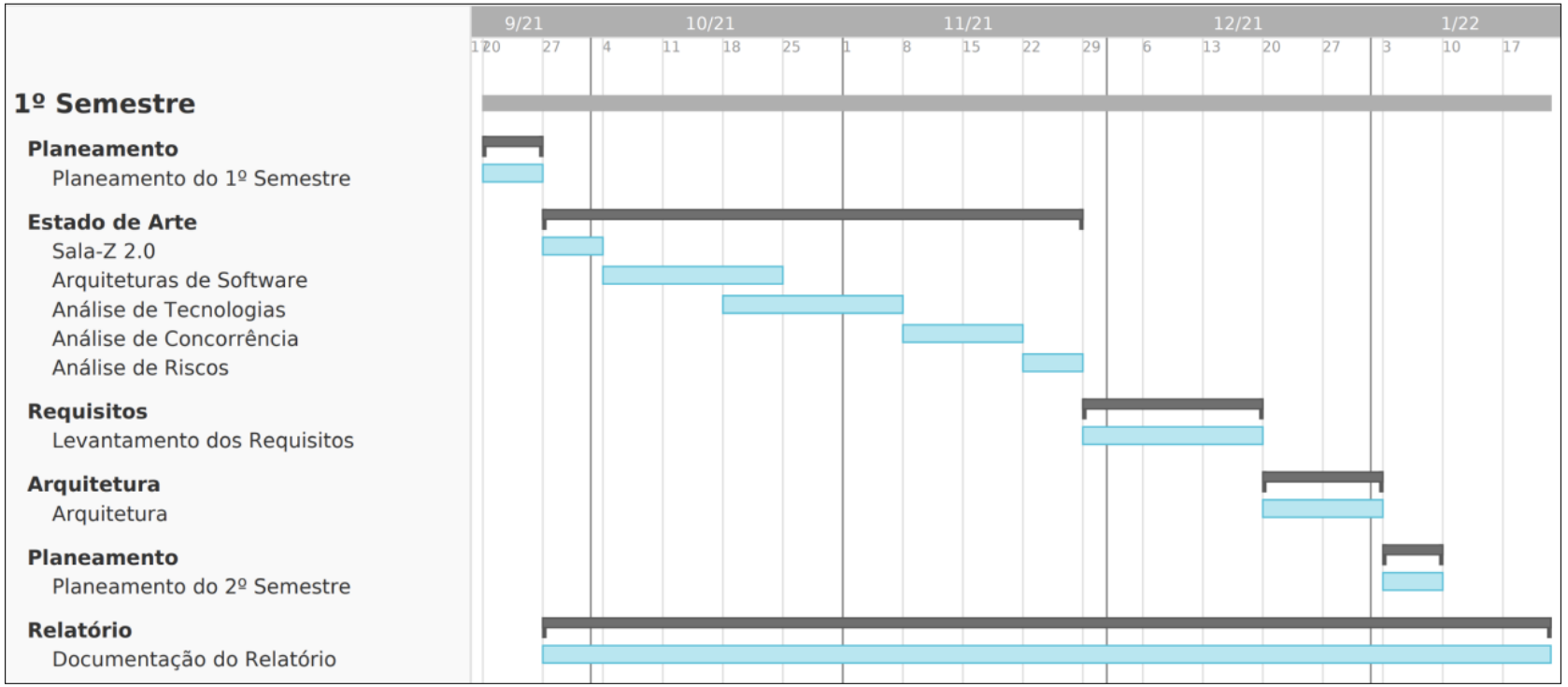

Figura 4.4: Resultado do 1º Semestre

4.2.2 2º Semestre

O segundo semestre é mais focado no desenvolvimento da plataforma Sala-Z 2.1. Segue uma metodologia semelhante ao *Scrum*, com *daily meetings* e *sprints* quinzenais.

Planeamento Temporal

O segundo semestre começa com a primeira semana dedicada à exploração da plataforma Sala-Z e das tecnologias, tarefa pendente do semestre anterior.

O resto do semestre é dividido num total de 10 *sprints* (ver Figura 4.5), os primeiros 8 *sprints* tem como objetivos o desenvolvimento das funcionalidades, o penúltimo é dedicado à testagem e o último à finalização do relatório final. Os *sprints* serão planeados durante o *sprint planning*, representado a verde, segue-se o *sprint*, representado a azul, e termina, com o *sprint review* e o *sprint retrospective*, a vermelho.

Resultado

Uma vez que os *sprints* começaram e terminaram nas datas previstas, não será inserido um gráfico. Contudo, serão descritas por temas as funcionalidades desenvolvidas em cada *sprint*:

- Sprint #1 - painel de administração da *venue* e gestor de inventário;
- Sprint #2 - gestão de inventário;
- Sprint #3 - gestão de artistas;
- Sprint #4 - gestão de artistas;
- Sprint #5 - gestão de artistas e painel de administração do artista;
- Sprint #6 - gestão de tarefas;
- Sprint #7 - gestão de tarefas;
- Sprint #8 - gestão de eventos do artista e configuração de relatórios;
- Sprint #9 - realização dos testes às funcionalidades;
- Sprint #10 - escrita do relatório final.

Os primeiros dois *sprints* focam-se no desenvolvimento de requisitos funcionais relacionadas com a gestão de inventário. Já o terceiro, quarto e quinto *sprint* passa pela adição do novo serviço *Artist Management* com o desenvolvimento das funcionalidades de gestão de artistas tanto para um gestor de uma *venue* como para

um gestor de artistas. Durante o sexto e sétimo *sprint* as funcionalidades relacionadas com a gestão de tarefas e durante o último *sprint* do desenvolvimento foram desenvolvidas as funcionalidades que permite a um artista visualizar os eventos a que foi associado, bem como a configuração de relatórios tanto para um gestor de *venue* como um artista. As funcionalidades relacionadas com o painel de administração da *venue* foram desenvolvidas durante o primeiro, enquanto o painel de administração relacionado com o artista foi desenvolvido durante o quinto *sprint*.

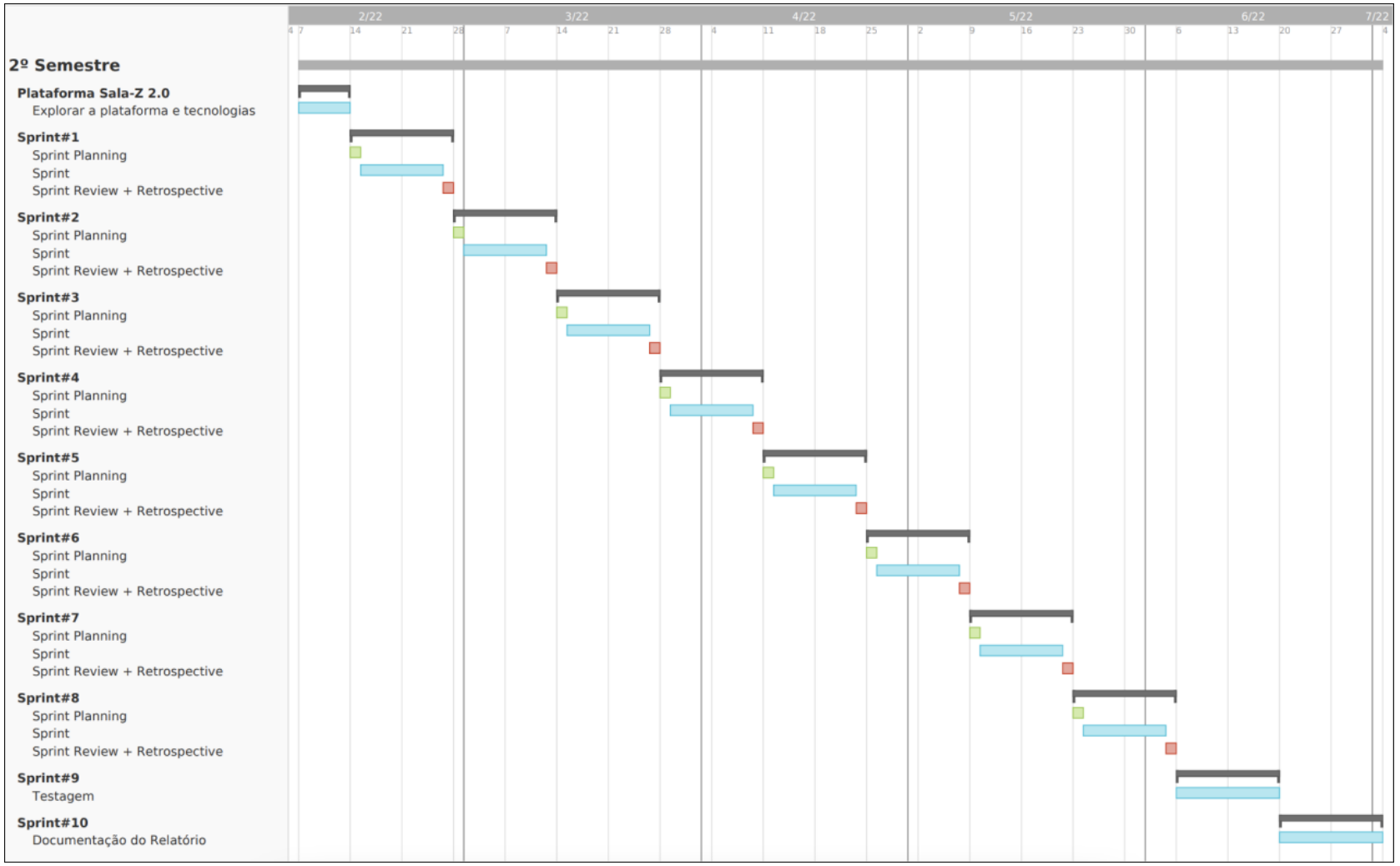

Figura 4.5: Planeamento Temporal do 2º Semestre

4.3 Planeamento dos Testes

4.3.1 Testes funcionais

Como referido na **Testagem**, da Subsecção 3.2.2, os tipos de testagem escolhidos sobre as funcionalidades desenvolvidas são *Unit Testing* e *Integration Testing*.

Os *Unit Tests* serão realizados sobre todos os *endpoints*, recorrendo à ferramenta ***JUNIT*** [65]. Assim, será verificado se a resposta retornada é a esperada com a validação de um dos seus campos. Caso o teste falhe, a funcionalidade terá de ser corrigida. Este tipo de testagem será realizada após a adição de cada *endpoint*.

No final do desenvolvimento, durante o *Sprint #9*, serão realizados testes de integração à interface do utilizador, através de um plano testes que cobrirá cada *User Story (US)* definida. Uma vez encontrados erros, estes serão corrigidos e o teste será novamente executado até ser validado.

Resultado

Numa fase inicial cada *endpoint* começou por ser testado recorrendo à ferramenta ***JUNIT*** com testes automatizados, no entanto, devido aos atrasos no desenvolvimento numa fase inicial do projeto e para que o foco fosse o desenvolvimento das funcionalidades, passou-se à utilização de testes manuais sobre *endpoints*, com testes funcionais, recorrendo à plataforma *Swagger* [80].

No final do desenvolvimento durante o Sprint #9, foram realizados *smoke tests* com testes de integração ao *frontend* e *backend* da plataforma. Os *smoke tests* são realizados após o *build* da aplicação e antes de uma *release*, sobre o UX da plataforma [82], para isso, foi criado um plano de testes que valida cada *US*.

4.3.2 Testes não funcionais

Ao longo do semestre, serão realizados testes de validação de modo a verificar os requisitos não funcionais levantados. Na Secção 8.2 são descritas as ferramentas utilizadas na validação dos requisitos e o processo de testagem.

4.4 Riscos

A identificação dos riscos associados a um projeto é de extrema importância. Os riscos são potenciais ameaças que possam levar ao falhanço de um projeto. Assim, é muito importante que cada risco tenha associado um plano de mitigação. O impacto e plano de mitigação foram definidos com base na relevância do risco para a execução do processo [74].

De seguida, são apresentados os diversos riscos onde é introduzida a descrição do risco, impacto, plano de mitigação e consequência.

1. Pandemia

Descrição do risco: Devido à situação pandémica, existiram restrições associadas ao método de trabalho como adotar o regime remoto.

Consequência: O desempenho e motivação do aluno pode diminuir uma vez que este não terá um acompanhamento tão próximo da empresa.

Plano de Mitigação: Acompanhamento remoto de toda a equipa envolvida no projeto e empresa, através de várias ferramentas de comunicação como o *Slack* [34] e o *Discord* [12].

Impacto: Baixo

Probabilidade: 11% a 30%

2. Tecnologias

Descrição do risco: O desenvolvimento da plataforma depende das tecnologias escolhidas. Sendo estas tecnologias desenvolvidas por terceiros, eventuais atualizações ou falta de documentação e funcionalidades podem dificultar o desenvolvimento do projeto.

Consequência: Atraso no desenvolvimento do projeto na procura de soluções face aos problemas encontrados podem conduzir a um atraso das entregas nos prazos estabelecidos.

Plano de Mitigação: A utilização de uma arquitetura baseada em microsserviços permite a escolha das tecnologias a serem utilizadas, permitindo que as escolhidas sejam do conhecimento de membros da empresa para que exista suporte ao aluno. Sempre que houver dificuldade na procura de soluções face aos problemas encontrados, estes serão discutidos com o orientador da empresa a fim de solucionar o problema.

Impacto: Alto

Probabilidade: 31% a 50%

3. Falta de experiência

Descrição do risco: O aluno desconhece as tecnologias escolhidas e o projeto Sala-Z.

Consequência: Atraso no desenvolvimento do projeto pode conduzir a um atraso nas entregas nos prazos estabelecidos.

Plano de Mitigação: O aluno irá preparar-se previamente, resolvendo pequenas funcionalidades em falta na plataforma para que este conheça as tecnologias utilizadas e o projeto.

Com a utilização do *Scrum* no desenvolvimento os *sprints* serão curtos para que exista *feedback* constante do cliente. O *backlog* será priorizado e organizado não só consoante as necessidades do cliente mas também da equipa de desenvolvimento, constituída pelo aluno. Caso o aluno apresente dificuldade na execução das tarefas, as tarefas mais fáceis serão implementadas primeiro para que este se familiarize com a tecnologia.

Impacto: Alto

Probabilidade: 31% a 50%

Matriz de Riscos

Após o levantamento dos riscos e dos cálculos do impacto e da probabilidade associada a cada risco, foi construída uma matriz de riscos [74].

A matriz de riscos (ver Figura 4.6) permite visualizar os riscos. No eixo horizontal é associada a probabilidade de ocorrência do risco enquanto que no eixo vertical tem-se o impacto que o risco gera no projeto caso aconteça.

Probabilidade	Muito Baixo	Baixo	Moderado	Alto	Muito Alto
90%	Média	Média	Alta	Alta	Alta
70%	Baixa	Média	Média	Alta	Alta
50%	Baixa	Baixa	Média	Alta (2, 3)	Alta
30%	Baixa	Baixa (1)	Média	Média	Alta
10%	Baixa	Baixa	Baixa	Baixa	Média

Impacto

Figura 4.6: Matriz de riscos

Capítulo 5

Requisitos

De modo a documentar as funcionalidades a serem desenvolvidas, foram levantados os requisitos funcionais em forma de *user stories*, com base em *epic stories* disponibilizadas pela empresa. Após a construção das *user stories*, os requisitos foram priorizados seguindo o método MosCoW [20].

Após o levantamento dos requisitos funcionais, definiram-se os requisitos não funcionais em forma de cenários de atributos de qualidade para que seja possível validar a qualidade do comportamento da plataforma.

5.1 Atores

Na utilização da plataforma, existem diferentes tipos de atores e papéis:

- **Administrador do Sala-Z** - administrador da plataforma. Gere *venues*, artistas e contas de utilizadores no *backoffice* da plataforma. Tem acesso a toda a plataforma e a todas as informações associadas;
- **Administrador da *venue*** - qualquer utilizador registado no sistema para gerir *venues* e a sua informação associada, bem como adicionar novos colaboradores às suas *venues*;
- **Colaborador da *venue*** - qualquer utilizador registado no sistema para colaborar em *venues* e gerir informação associada a estas;
- **Administrador do artista** - qualquer utilizador registado no sistema para gerir artistas e a sua informação associada, bem como adicionar novos colaboradores aos seus artistas;
- **Colaborador do artista** - qualquer utilizador registado no sistema para colaborar em perfis de artistas e gerir informação associada a estes;
- **Utilizador não autenticado** - qualquer utilizador que tenha acesso à informação pública disponível na plataforma Sala-Z, poderá procurar e comprar bilhetes, pesquisar *venues* e artistas disponíveis.

Na implementação das novas funcionalidades apenas serão tidos em conta os seguintes atores: administrador da *venue*, colaborador da *venue*, administrador do artista, colaborador da artista. Assim, nas *user stories*, será utilizado o termo "utilizador" para referir tanto o administrador como o colaborador de uma determinada conta:

- **Utilizador da *venue*** - representa tanto o administrador como o colaborador de uma *venue*;
- **Utilizador do artista** - representa tanto o administrador como o colaborador de um artista.

5.2 *User Stories*

A empresa forneceu um conjunto de *epic stories* para o aluno fazer o levantamento dos requisitos da plataforma. As *epic stories* consistem numa funcionalidade do produto que é demasiado grande para que seja concluída num único *sprint*, por isso é dividida em funcionalidades mais pequenas, as *user stories* [40]. Uma *User Story (US)*, é uma forma informal de especificar as funcionalidades de um *software* da perspetiva de utilizador [85]. Podem ser definidas por uma frase que define um **tipo de utilizador** que tem uma **intenção** - uma ação - que pretende realizar devido a uma **razão** [85].

Assim, para cada funcionalidade incluída em uma *Epic Story (ES)* foi construída uma *US* seguindo a seguinte estrutura:

"**Como um** [tipo de utilizador], **eu quero** [intenção] **para que** [razão]."

Todas as *user stories* construídas foram incluídas no apêndice A. De seguida, é apresentado um exemplo do módulo de gestão de salas, relativo ao tema de painel de administração.

Módulo: Gestão de *Venues*

Painel de Administração

ES-1: **Como um** administrador da *venue* autenticado, **eu quero** gerir as contas de utilizadores com acesso à minha *venue* **para que** eu possa gerir cada utilizador (administrador e colaboradores de *venue*) que pode aceder e gerir a informação da minha *venue* na plataforma do Sala-Z.

US-1: Enviar convite a um novo utilizador para a minha *venue*

(a) **Como um** administrador de *venue* autenticado, **eu quero** enviar um convite a um novo utilizador **para que** este possa gerir a informação relativa à minha *venue*. Quero inserir:

- Voltar a adicionar utilizador removido;
- Nome*;
- Apelido*;
- Email*;

(* campos obrigatórios)

US-2: Reenviar convite a um novo utilizador para a minha *venue*

(a) **Como um** administrador de *venue* autenticado, **eu quero** reenviar um convite a um novo utilizador **para que** o convite seja novamente enviado. Quero inserir:

- Voltar a adicionar utilizador removido;
- Nome*;
- Apelido*;
- Email*;

(* campos obrigatórios)

US-3: Listar utilizadores com acesso à minha *venue*

(a) **Como um** administrador de *venue* autenticado, **eu quero** listar utilizadores com acesso à minha *venue* **para que** possa ver os utilizadores com permissão de gestão da minha *venue*.

US-4: Remover utilizador da *venue*

(a) **Como um** administrador de *venue* autenticado, **eu quero** remover utilizador com acesso à minha *venue* **para que** possa remover a permissão de gestão da minha *venue* a um utilizador.

5.3 Requisitos Funcionais

Com base nas *user stories* foram listados os requisitos funcionais divididos por módulos e temas. Cada requisito funcional foi priorizado seguindo o método ***MoScoW*** [20]. Assim, temos as seguintes categorias:

- ***Must Have*** **(M)**: representa os requisitos necessários para o projeto e não-negociáveis.
- ***Should Have*** **(S)**: representa os requisitos importantes para o projeto mas não necessários, acrescentam valor ao projeto.
- ***Could Have*** **(C)**: representa os requisitos que se não forem implementados apresentam pouco impacto no projeto.
- ***Will Not Have*** **(W)**: representa os requisitos que não serão implementados de momento, nesta versão, mas que poderão ser implementados no futuro.

Módulo: Gestão de *Venues*

Tema	Requisitos	Prioridade
Painel de administração	Enviar convite a um novo utilizador para a minha sala	(M)
	Reenviar convite a um novo utilizador para a minha sala	(C)
	Listar utilizadores com acesso à *venue*	(M)
	Remover utilizador da *venue*	(M)

Tabela 5.1: Gestão de *Venues* - Painel de administração

Tema	Requisitos	Prioridade
Gestão de inventário	Criar item no inventário	(M)
	Criar item no inventário com um número de série	(C)
	Listar itens da *venue*	(M)
	Ordenar itens na lista de itens	(S)
	Pesquisar item na lista de itens	(S)
	Filtrar itens na lista de itens	(S)
	Ver detalhes de um item	(M)
	Editar item	(M)
	Duplicar item	(S)
	Eliminar item	(M)
	Associar um item a um evento	(C)

Tabela 5.2: Gestão de *Venues* - Gestão inventário

Tema	Requisitos	Prioridade
Gestão de artistas	Criar artista com importação	(C)
	Criar artista manualmente	(M)
	Listar artistas	(M)
	Listar artistas favoritos	(C)
	Procurar artista na lista de artistas	(S)
	Filtrar lista de artistas	(S)
	Ver detalhes de um artista	(M)
	Editar artista	(M)
	Eliminar artista	(M)
	Marcar um artista como favorito	(C)
	Ver as estatísticas associadas a artistas	(S)

Tabela 5.3: Gestão de *Venues* - Gestão artistas

Tema	Requisitos	Prioridade
Dashboard	Ver as estatísticas associadas à *venue*	(S)

Tabela 5.4: Gestão de *Venues* - *Dashboard*

Tema	Requisitos	Prioridade
Gestão de tarefas de um evento	Rápida criação da tarefa	(C)
	Criar tarefa com detalhes	(M)
	Listar tarefas	(M)
	Ver detalhes de uma tarefa	(C)
	Ordenar lista de tarefas	(S)
	Procurar tarefa na lista de tarefas	(S)
	Filtrar lista de tarefas	(S)
	Editar tarefa	(M)
	Duplicar tarefa	(S)
	Eliminar tarefa	(M)
	Ver a atividade associada à tarefa	(C)
	Adicionar comentário à tarefa	(M)
	Editar comentário	(S)
	Eliminar comentário	(S)

Tabela 5.5: Gestão de *Venues* - Gestão de tarefas de um evento

Tema	Requisitos	Prioridade
Gestão de tarefas sala/utilizador	Ver as estatísticas associadas a tarefas	(C)

Tabela 5.6: Gestão de *Venues* - Gestão de tarefas sala/utilizador

Tema	Requisitos	Prioridade
Comunicação entre *stakeholders*	Mencionar outros utilizadores da *venue* com comentários	(S)

Tabela 5.7: Gestão de *Venues* - Comunicação entre *stakeholders*

Tema	Requisitos	Prioridade
Notificações	Ver notificações	(S)

Tabela 5.8: Gestão de *Venues* - Notificações

Tema	Requisitos	Prioridade
Configurar relatórios de eventos	Configurar relatórios de eventos	(C)

Tabela 5.9: Gestão de *Venues* - Configurar relatórios de eventos

Tema	Requisitos	Prioridade
Gerir templates de eventos	Criar template	(C)
	Listar templates	(C)
	Editar template	(C)
	Eliminar template	(C)
	Criar evento a partir de um template	(C)
	Criar template a partir de um evento	(C)

Tabela 5.10: Gestão de *Venues* - Gerir templates de eventos

Tema	Requisitos	Prioridade
Gerir eventos	Duplicar evento	(C)

Tabela 5.11: Gestão de *Venues* - Gerir eventos

Módulo: Gestão de Artistas

Tema	Requisitos	Prioridade
Painel de administração	Enviar convite a um novo utilizador para o meu artista	(M)
	Reenviar convite a um novo utilizador para o meu artista	(C)
	Listar utilizadores com acesso ao meu artista	(M)
	Remover utilizador do artista	(M)

Tabela 5.12: Gestão de Artistas - Painel de administração

Tema	Requisitos	Prioridade
Gestão de artistas	Criar artista com importação	(C)
	Criar artista manualmente	(M)
	Listar artistas	(M)
	Filtrar lista de artistas	(S)
	Ver detalhes de um artista	(M)
	Editar artista	(M)
	Eliminar artista	(M)
	Adicionar carimbo de verificação a artista	(S)

Tabela 5.13: Gestão de Artistas - Gestão artistas

Tema	Requisitos	Prioridade
Dashboard	Ver as estatísticas associadas a artistas	(S)
	Filtrar lista de artistas na tabela	(S)
	Ordenar artistas na tabela	(S)

Tabela 5.14: Gestão de Artistas - *Dashboard*

Tema	Requisitos	Prioridade
Gestão de eventos	Listar eventos do artista	(M)
	Filtrar lista de eventos do artista	(S)
	Ver detalhes do evento	(M)

Tabela 5.15: Gestão de Artistas - Gestão de eventos

Tema	Requisitos	Prioridade
Comunicação entre *stakeholders*	Mencionar outros utilizadores do evento com comentários	(S)

Tabela 5.16: Gestão de Artistas - Comunicação entre *stakeholders*

Tema	Requisitos	Prioridade
Notificações	Ver notificações	(S)

Tabela 5.17: Gestão de Artistas - Notificações

Tema	Requisitos	Prioridade
Configurar relatórios de eventos	Configurar relatórios de eventos	(C)

Tabela 5.18: Gestão de Artistas - Configurar relatórios de eventos

5.4 Requisitos Não Funcionais

Os requisitos não funcionais ou atributos de qualidade, definem os critérios para avaliar a qualidade de um sistema [46]. De seguida são representados os requisitos não funcionais do sistema, em forma de cenários de atributos de qualidade.

5.4.1 Disponibilidade

O atributo de qualidade disponibilidade expressa o rácio de disponibilidade do sistema para o tempo total que o sistema tem de estar disponível [46]. Assim, foca-se no tempo em que o sistema está fora de serviço, como é que falhas podem ser impedidas e que tipo de notificações são necessárias quando o sistema está em baixo [17].

Cenário: Utilizador acede à plataforma.

Fonte:	Utilizador não autenticado
Estimulo:	Acede à plataforma
Artefacto:	Plataforma
Ambiente:	Operação Normal
Resposta:	O sistema está em baixo e a página não é apresentada.
Medida:	O sistema está disponível 99% do tempo ao utilizador.

Tabela 5.19: Cenário de Atributo de Qualidade - Disponibilidade

5.4.2 Escabilidade

O atributo de qualidade escabilidade representa a habilidade do sistema manter uma boa *performance* mesmo que o tempo para suportar o carregamento aumente [46]. Há duas formas de melhorar a escabilidade de um sistema, verticalmente, aumentando o número de recursos como memória, discos ou processadores no sistema [46]. Por outro lado pode ser melhorada horizontalmente, com o aumento do número de unidades de computação ou dividindo o carregamento [46]. Segundo a *Google*, o tempo recomendado de carregamento de uma página deve ser inferior a dois segundos e, 57% dos utilizadores abandona a página se o tempo de carregamento ultrapasar os 3 segundos [16]. Assim, o tempo de carregamento da informação não deve ultrapassar os três segundos.

Cenário: O número de pedidos realizados por utilizadores aumenta.

Fonte:	Utilizador autenticado
Estimulo:	O número de pedidos ao servidor aumenta
Artefacto:	Plataforma
Ambiente:	Operação Normal
Resposta:	O sistema adapta-se, disponibilizando mais recursos para dar resposta à sobrecarga de pedidos.
Medida:	O sistema deverá responder em menos de 3 segundos.

Tabela 5.20: Cenário de Atributo de Qualidade - Escabilidade

5.4.3 Portabilidade

A portabilidade é o atributo de qualidade que define a capacidade do sistema ser acedido por outros *hardwares* [17]. No caso da plataforma, esta deve se adaptar quando acedida por ecrãs de diferentes dimensões ou seja, deve ser garantida responsividade.

Cenário: Utilizador acede à plataforma a partir de um dispositivo móvel.

Fonte:	Utilizador autenticado
Estimulo:	Acede à plataforma num telemóvel
Artefacto:	Plataforma
Ambiente:	Operação Normal
Resposta:	Página responsiva é apresentada ao utilizado.r
Medida:	A página deve ser responsiva para qualquer dispositivo móvel.

Tabela 5.21: Cenário de Atributo de Qualidade - Portabilidade

5.4.4 Segurança

O atributo de qualidade, segurança, garante a capacidade do sistema responder a ataques, a tentativas de terceiros acederem a informação confidencial ou que não lhes pertença, à perda de informação [46]. Várias medidas são utilizadas para proteção como, por exemplo, encriptação e autenticação.

Cenário: Utilizador tenta fazer uma operação para a qual não está autorizado

Fonte:	Utilizador
Estimulo:	Tenta aceder a uma página para o qual não está autorizado
Artefacto:	Plataforma
Ambiente:	Sistema Protegido
Resposta:	Redireciona o utilizador não autenticado para uma página autorizada.
Medida:	O sistema deve bloquear o acesso quando um utilizador acede a algo para o qual não está autorizado.

Tabela 5.22: Cenário de Atributo de Qualidade - Segurança

Para além do referido anteriormente, será também utilizado a ferramenta *OWASP Zed Attack Proxy (ZAP)* [93], na procura de problemas de segurança que possam permitir que um utilizador realize operações para o qual não está autorizado ou que possibilitam o ataque à plataforma.

Como referido na Subsecção 4.3.2, ao longo do semestre serão realizados testes de validação de modo a verificar os requisitos não funcionais levantados. No caso da portabilidade com recurso a diferentes dispositivos móveis.

Capítulo 6

Arquitetura

Este capítulo apresenta a arquitetura da plataforma Sala-Z 2.1, reestruturada e desenhada para a adição das novas funcionalidades, seguindo uma arquitetura baseada em microsserviços.

Numa primeira fase, serão apresentados os serviços da *Amazon Web Services (AWS)* presentes na arquitetura, uma vez que a empresa pretende continuar com a sua utilização para o alojamento da plataforma, como na versão Sala-Z 2.0.

De seguida, é apresentada a arquitetura seguindo o modelo C4 [71] que divide a arquitetura em 4 níveis: *Context*, *Container*, *Component* e *Code*.

6.1 *AWS*

A *AWS* é a infraestrutura em nuvem mais extensa, confiável e segura a nível global [11]. Conta com mais de mais de 200 serviços para uma ampla variedade de tecnologias sendo o preço calculado mediante o número de recursos utilizados e difere de serviço para serviço [11]. "A *AWS* é compatível com 90 normas de segurança e certificações de conformidade, e todos os 117 serviços que armazenam dados de clientes oferecem criptografia de dados"[11]. É reconhecida por ter "várias zonas de disponibilidade conectadas por redes altamente redundantes com baixa latência e alta taxa de transferência"[11], cerca de 84 zonas de disponibilidade em 26 regiões geográficas em todo o mundo [11].

Os serviços da *AWS* presentes na nova arquitetura estão a seguir especificados:

- ***API Gateway*** - é um serviço de *APIs* que age como "porta de entrada" para o *backend* de uma aplicação [4].
- ***Amazon Elastic Compute Cloud (EC2)*** - permite instanciar máquinas virtuais para correr aplicações [59].
- ***Elastic Load Balancing (ELB)*** - "distribui automaticamente o tráfego de aplicações de entrada entre vários destinos e dispositivos virtuais em uma ou mais zonas de disponibilidade" [47].

- ***Amazon Simple Notification Service (SNS)*** - message broker da *AWS*, que se baseia em mecanismos de *publish*, *subscribe*, ou seja, um *publisher* comunica de forma assíncrona com os *subscribers* de determinado tópico [6].
- ***Amazon Simple Queue Service (SQS)*** - um serviço de fila de mensagens da *AWS* que permite enviar, armazenar e receber mensagens [7].
- ***Amazon Simple Storage Service (S3)*** - serviço de armazenamento de dados da *Amazon* na nuvem, permite a criação de diferentes *buckets* e base de dados [5].
- ***Amazon Cognito*** - serviço de autenticação de utilizadores da *Amazon* [54].

6.2 Modelo C4

Na secção é apresentada a arquitetura, dividida por níveis, seguindo o modelo C4 [71].

6.2.1 *Context*

O diagrama de contexto é o ponto de partida do modelo C4 no primeiro nível. Neste nível são representados o *software*, os sistemas externos que interagem com este e os utilizadores.

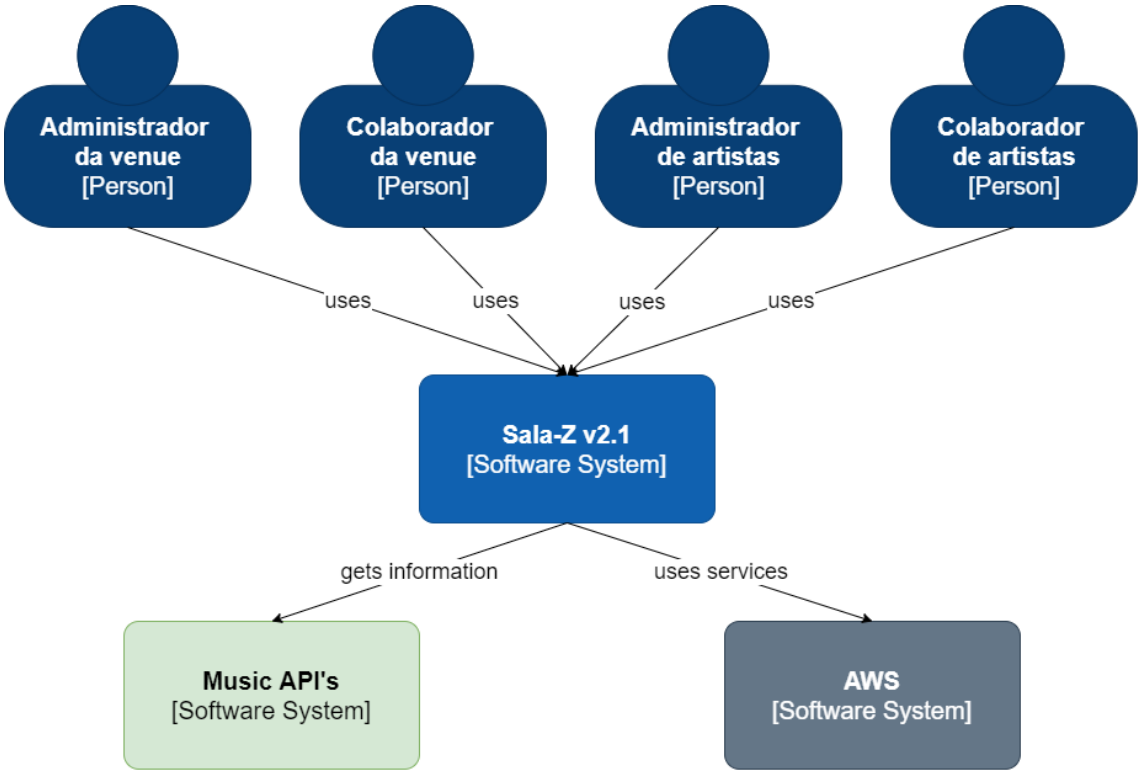

Figura 6.1: *Context diagram*

No diagrama, apenas estão definidos os utilizadores especificados nas *user stories*, que utilizam as funcionalidades a serem implementadas:

- **Administrador da *venue*** - qualquer utilizador registado no sistema para gerir *venues* e a sua informação associada, bem como adicionar novos colaboradores às suas *venues*;
- **Colaborador da *venue*** - qualquer utilizador registado no sistema para colaborar em *venues* e gerir informação associada a estas;
- **Administrador do artista** - qualquer utilizador registado no sistema para gerir artistas e a sua informação associada, bem como adicionar novos colaboradores aos seus artistas.
- **Colaborador do artista** - qualquer utilizador registado no sistema para colaborar em perfis de artistas e gerir informação associada a estes.

Os sistemas externos que interagem com a plataforma são:

- ***Music API's*** - representam a integração da plataforma com as plataformas *Discogs*, *Bandcamp* e *Spotify*.
- ***AWS*** - representa a interação da plataforma com os serviços da *AWS* a alto nível, nos seguintes níveis estão apresentados os serviços a serem utilizados.

6.2.2 Container

O segundo nível representa o diagrama de *containers*, onde é representado os diferentes serviços e as suas interações no sistema de *software* "Sala-Z v2.1", representado no primeiro nível.

A arquitetura segue uma abordagem semelhante ao Sala-Z 2.0, com a utilização de diferentes serviços da *AWS*.

Os serviços a castanho representam os serviços desenvolvidos na versão Sala-Z 2.0:

- ***Authentication*** - serviço responsável pela gestão de autenticação dos utilizadores e dos seus dados pessoais.
- ***Venue Management*** - serviço responsável por toda a gestão das *venues* como listagem de eventos, recursos, artistas associados, entre outros.

O serviço ***Venue Management*** será reestruturado, as funcionalidades desenvolvidas neste módulo que não fazem parte da gestão de *venues*, como funcionalidades

relacionadas com a gestão de eventos, serão migradas para o serviço ***Event Management***. A migração destas funcionalidades será especificada durante a apresentação dos próximos serviços. Ao serviço ***Venue Management***, serão também adicionadas novas funcionalidades, relacionadas com a gestão de inventário.

Representados a roxo, tem-se os serviços partilhados com o outro estágio que decorre em paralelo, apresentado na Secção 2.2.

- ***Event Management*** - implementa todas as funcionalidades relativas aos eventos e gestão de tarefas do evento.
- ***Notifications Center*** - implementa todas as funcionalidades relativas a notificações necessárias para a comunicação entre utilizadores da plataforma.

Como referido anteriormente, a plataforma terá se sofrer uma reestruturação as funcionalidades referentes aos eventos e suas base de dados que se encontram no serviço ***Venue Management*** serão migradas para o novo serviço ***Event Management***. As tarefas relacionadas com a migração das funcionalidades já existentes e base de dados serão repartidas igualmente por ambos os estágios, ou seja o nível de esforço para a execução será calculado por ambos os estagiários durante o *sprint planning* e porteriormente atribuídas igualmente tendo em conta o esforço de cada tarefa. A criação de ambos os serviços será dividida, enquanto um estagiário será responsável pela criação do ***Event Management***, o outro será responsável pela criação do ***Notifications Center***. Posteriormente cada um irá proceder à adição das novas funcionalidades referentes a cada um dos estágios.

A verde encontra-se o serviço a ser implementado pelo aluno:

- ***Artist Management*** - implementa todas as funcionalidades relativas ao artista. Este módulo terá a integração com os sistemas externos *Discogs*, *Bandcamp* e *Spotify* no perfil de um artista.

O *backend* da aplicação terá como ponto de entrada um *API Gateway* que comunicará com um *ELB* para este distribuir os pedidos para cada um dos serviços. Cada serviço será alojado uma máquina *EC2*.

Para o envio de notificações, os serviços utilizam o *SNS*, deste modo os diferentes serviços podem enviar notificações para o tópico do *SNS* que o ***Notifications Center*** está subscrito.

O módulo ***Notifications Center***, serviço responsável pelas notificações, tem uma *SQS* associada para receber as notificações publicadas no tópico do *SNS*.

O esforço da configuração dos diferentes serviços da *AWS* necessários na nova versão do Sala-Z 2.1, que ainda não foram integrados com a plataforma, será repartidos por ambos os estagiários.

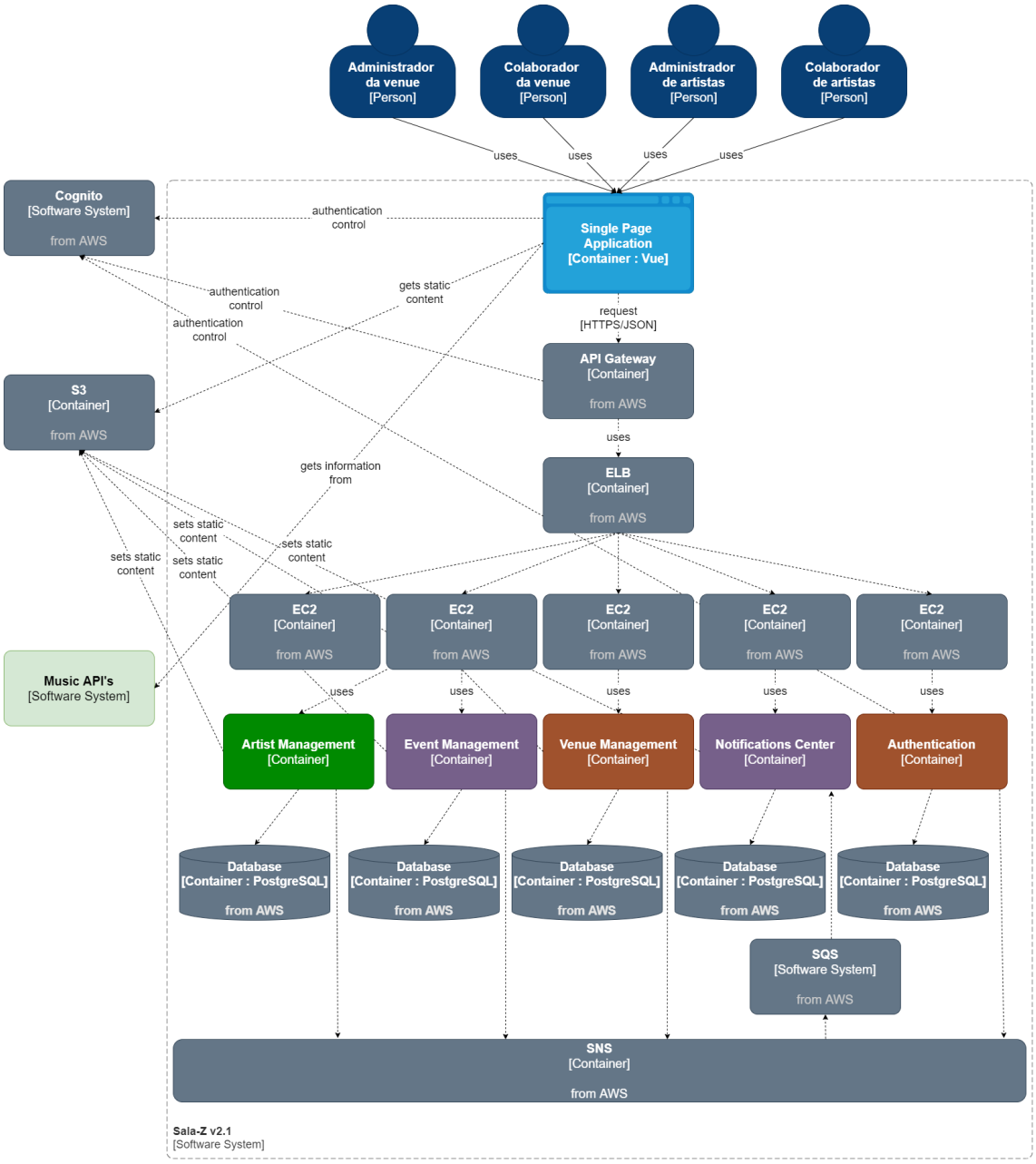

Figura 6.2: *Container diagram*

6.2.3 Component

O terceiro nível, nível do diagrama de componentes do sistema, mostra as componentes dentro de cada *container*. Ou seja, neste nível é representado os componentes para cada serviço.

A Figura 6.3, ilustra os diferentes componentes no serviço responsável pela gestão do artista bem como as diferentes interações com os sistemas externos. Temos o componente relacionado com a gestão de um artista, administração para o envio de convites para colaboradores, a componente relacionada com as estatísticas e métricas e por fim, relacionado com os relatórios.

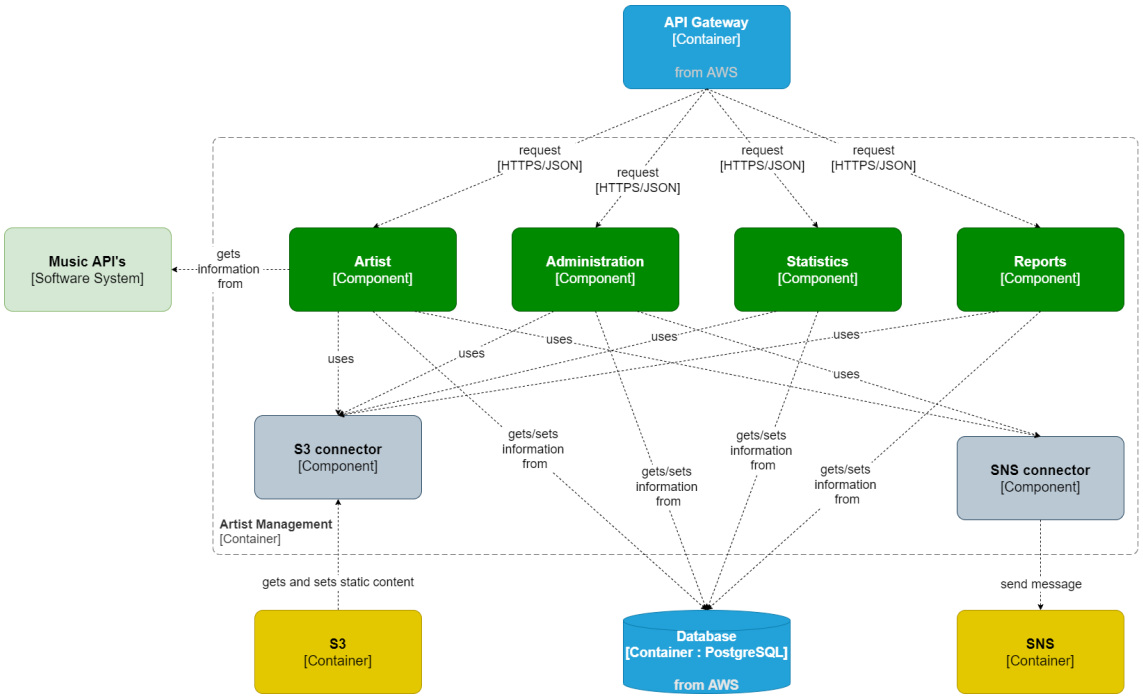

Figura 6.3: *Component diagram - Artist Management*

Na Figura 6.4 são representados os componentes do serviço de gestão de eventos. O componente ***Event*** é responsável pela gestão do evento, a ***Task*** pela gestão de tarefas e por último, os ***Templates*** pela criação de templates.

Na figura 6.5 é representada a componente do serviço ***Notifications Center***, apenas constituído pela componente ***Notifications***, responsável por toda a parte relativa a notificações.

A Figura 6.6 representa os componentes do serviço ***Venue Management***, responsável pela gestão das *venues*. Componente ***Inventory*** responsável pela gestão de inventário, ***Administration*** pelo painel de administração, ***Statistics***, pelas estatísticas e métricas e ***Reports*** pelos relatórios. A componente ***Venue*** é responsável pela gestão de *venues*, é representada pela cor roxa pois esta foi desenvolvida na versão Sala-Z v2.0.

A Figura 6.7 detalha as componentes presentes no *frontend* da aplicação. O ***Router*** encaminha os utilizadores para os diferentes módulos, certificando que este têm permissões de acesso através da componente ***Authentication Guard***. Os módulos são responsáveis pela apresentação visual da interface aos utilizadores, caso estes necessitem de informação do *backend* da aplicação, fazem pedidos aos diferentes serviços que se encarregam por sua vez de fazer pedidos *HTTPS* ao *backend* da plataforma. O módulo ***Authentication*** comunica com o sistema externo ***Amazon Cognito*** a fim de verificar a veracidade do utilizador que se tenta autenticar na plataforma.

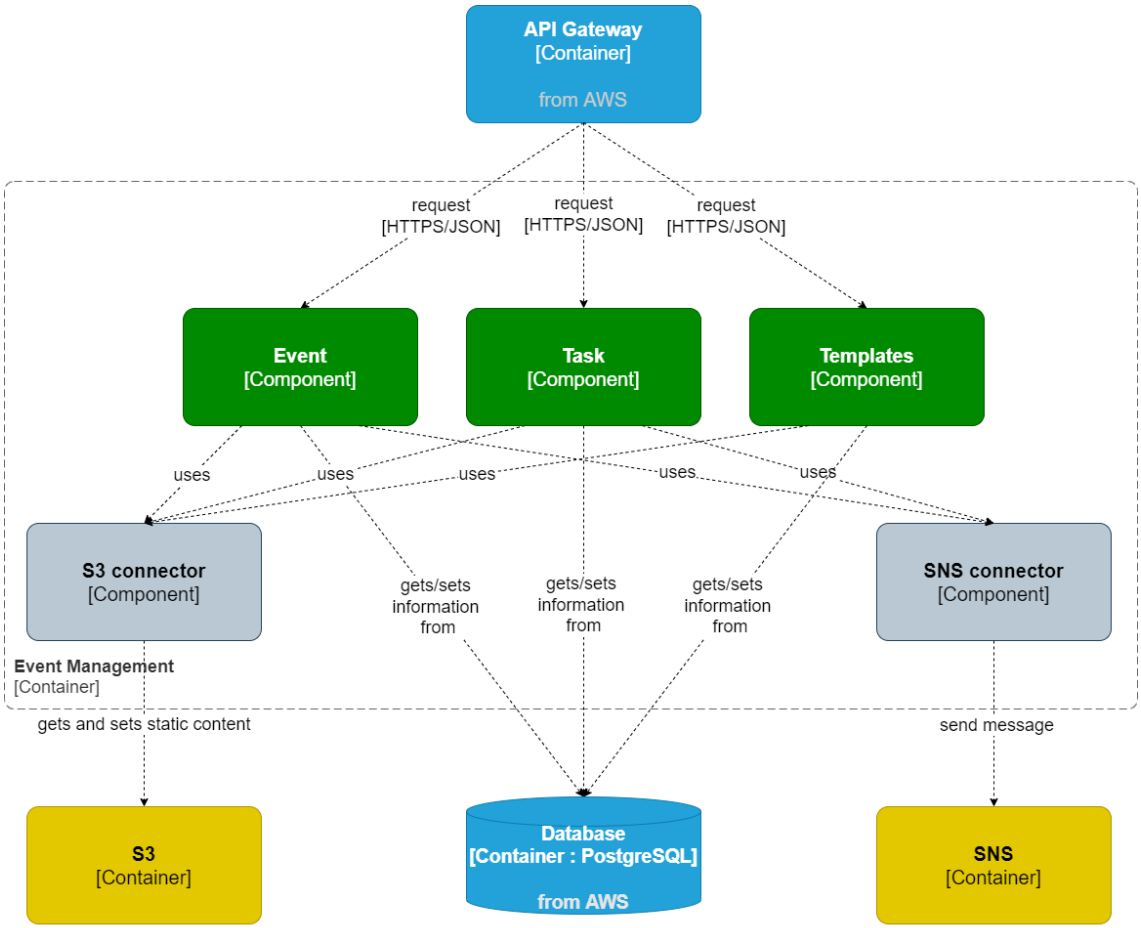

Figura 6.4: *Component diagram - Event Management*

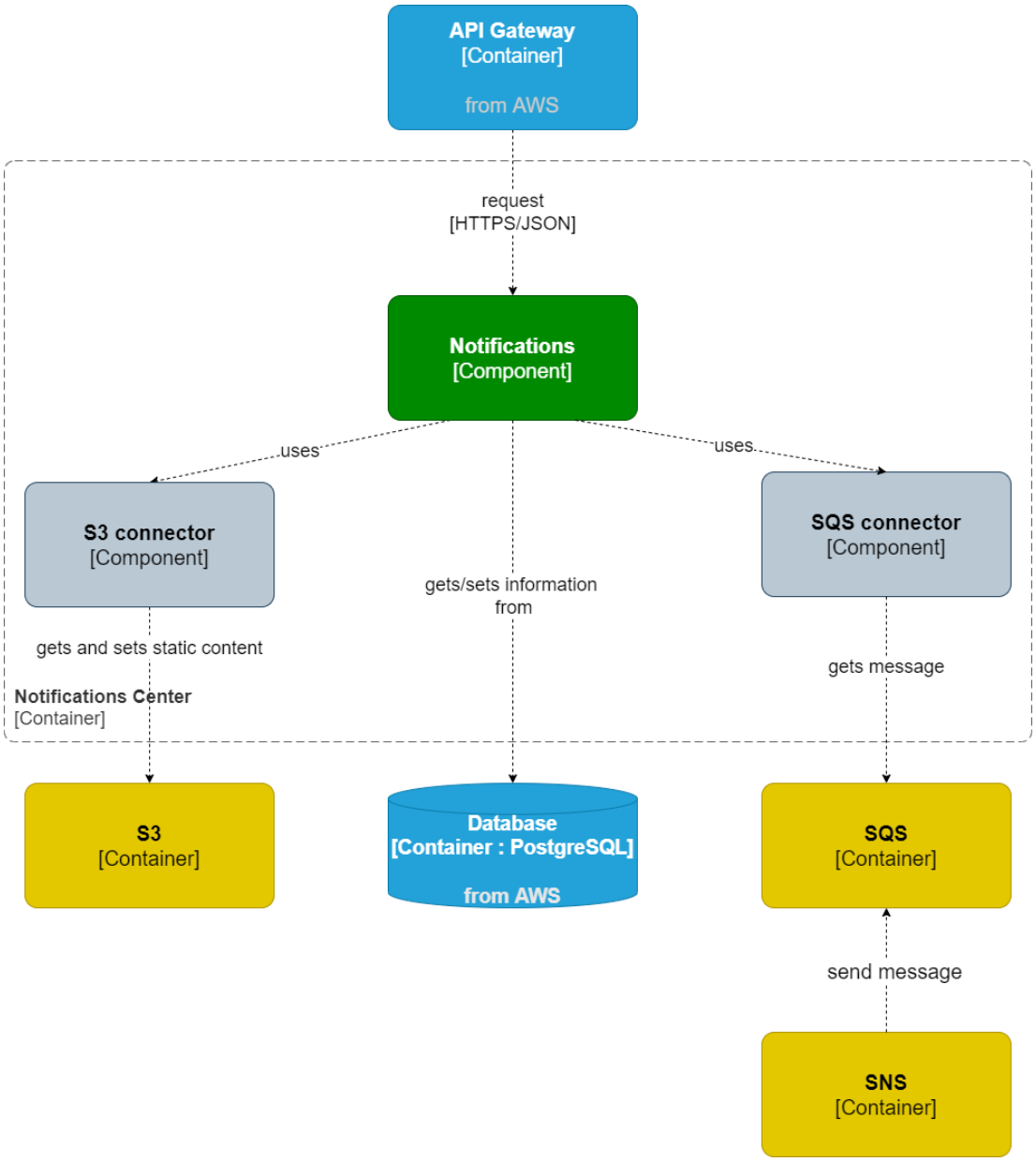

Figura 6.5: *Component diagram - Notifications Center*

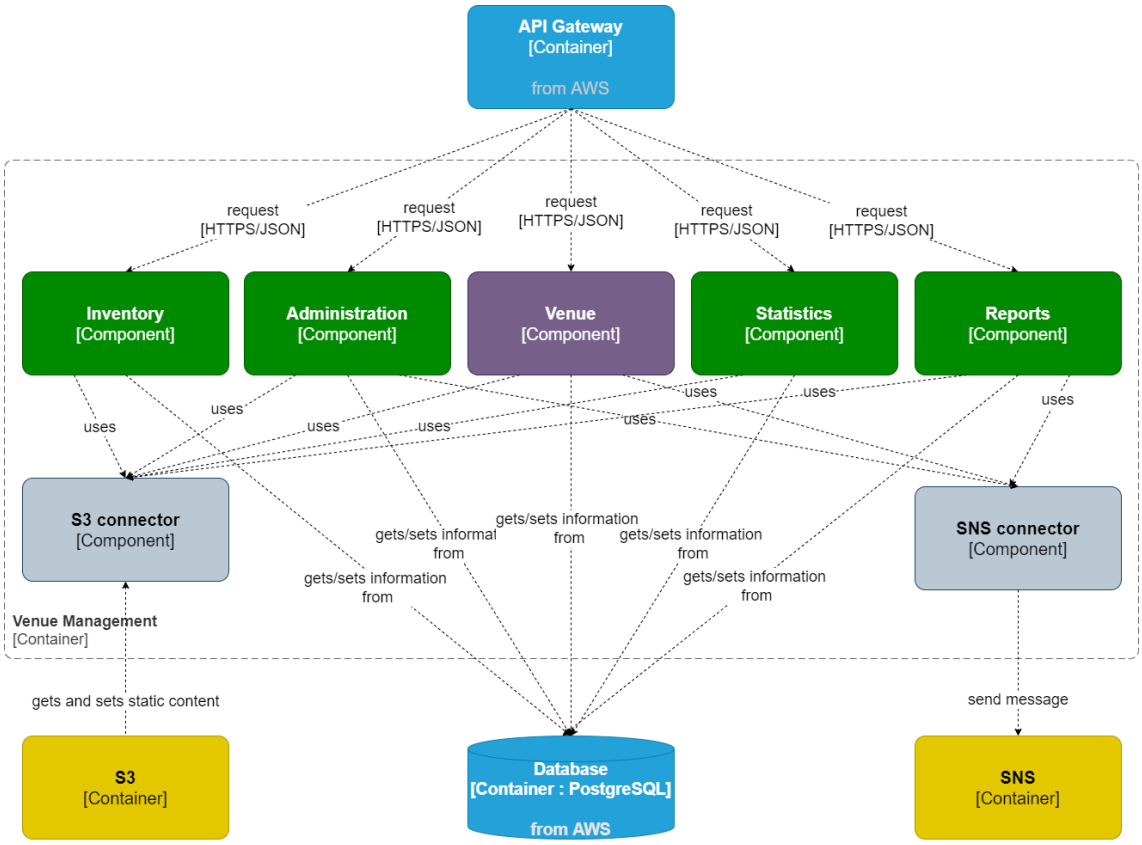

Figura 6.6: *Component diagram - Venue Management*

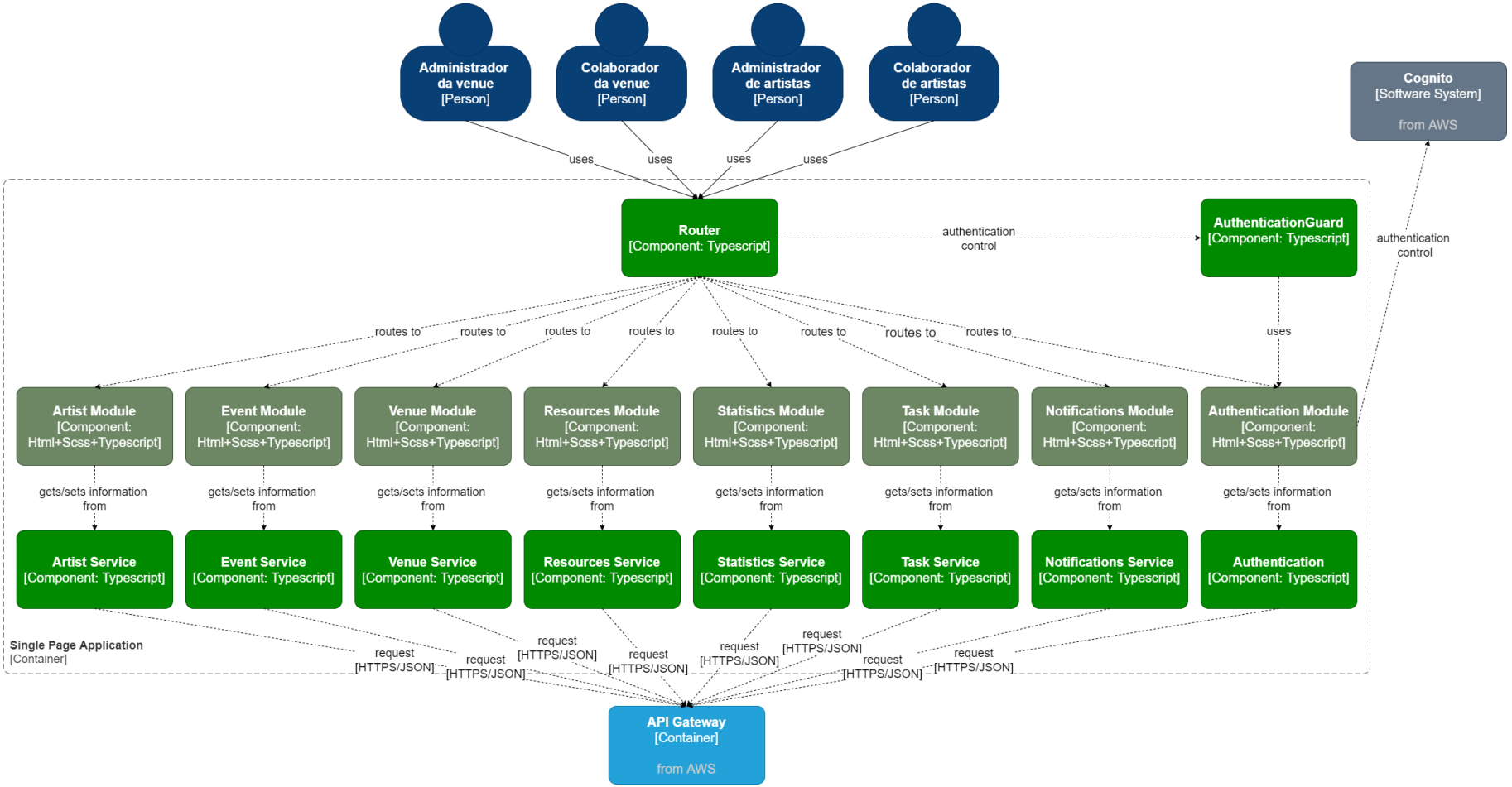

Figura 6.7: *Component diagram - Frontend*

6.2.4 Code

O nível 4, relativo ao diagrama de código, mostra como cada componente é implementado. Uma vez que este nível envolve um grande nível de detalhe e sendo este opcional [71], não será especificado.

6.3 Impacto dos requisitos não funcionais na arquitetura

Em termos de disponibilidade, de momento, esta é assegurada pela utilização de uma máquina *EC2* por serviço que assegura 99.99% de disponibilidade da máquina [59]. No futuro, a disponibilidade será garantida com a utilização de duas máquinas *EC2* por serviço, garantindo redundância e, consecutivamente, que os serviços se encontram disponíveis caso uma das máquinas falhe. A utilização de máquinas *EC2* permite também uma escabilidade vertical automática mediante a necessidade da utilização de recursos para os diferentes pedidos, permitindo que o *developer* não se preocupe com a alocação de recursos [58]. No caso da portabilidade, a utilização de uma *web single page* permite a construção de *layouts* responsivos através da utilização de responsividade para diferentes ecrãs. Em termos de segurança, a utilização do *Amazon Cognito* com a utilização de um token *JSON web token* (JWT) permite que o utilizador só aceda à aplicação caso tenha credenciais e estas sejam válidas para efetuar a autenticação.

6.4 Diagrama Entidade-Relacionamento

O Diagrama Entidade-Relacionamento permite visualizar como as entidades - pessoas, objetos, conceitos - se relacionam entre si no sistema.

No levantamento dos requisitos verifica-se a adição de três entidades à plataforma Sala-Z: a tarefa, o recurso da *venue* e o artista. Assim, para cada uma das entidades definiram-se as suas relações e os seus atributos.

Nos diferentes diagramas, definiram se a roxo, as entidades previamente criadas no Sala-Z 2.0.

A Figura 6.8, ilustra as relações entre entidades com foco na entidade *Resource*.

A Figura 6.9, ilustra a entidade *Task* e as suas relações.

Por fim, na Figura 6.10 são ilustradas as relações estabelecidas entre entidades com foco na entidade artista.

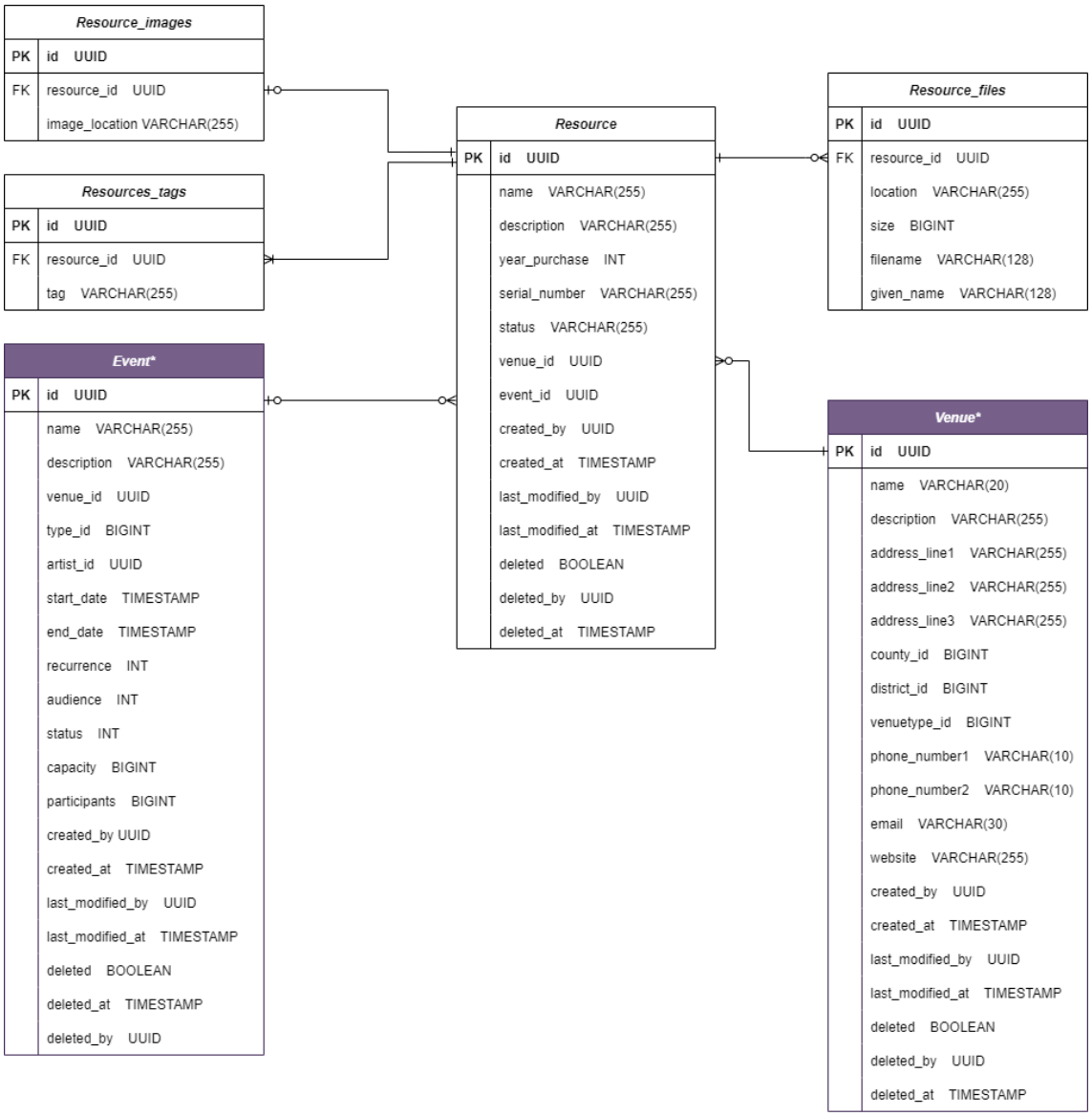

Figura 6.8: Relações da entidade *Resource*

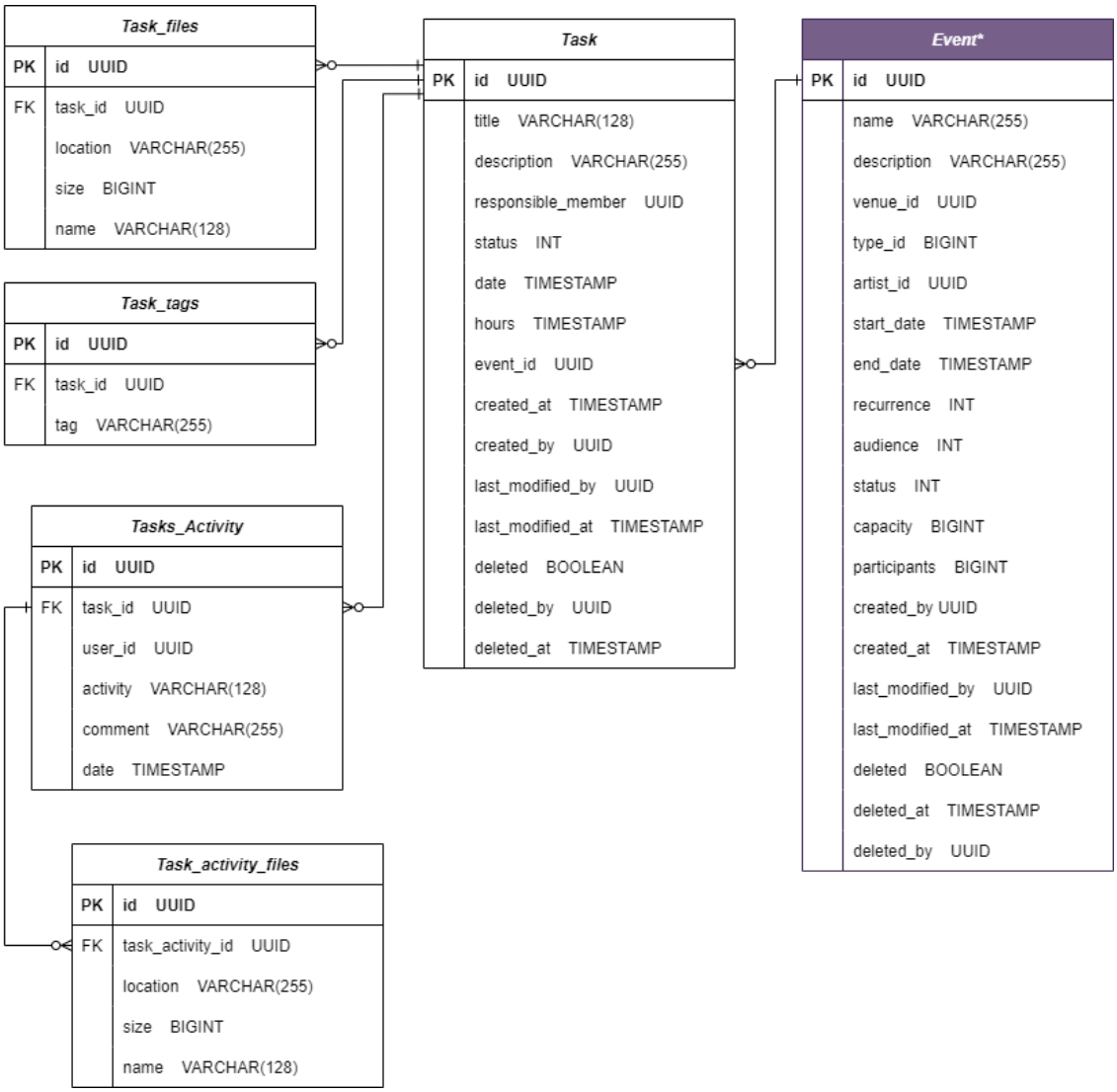

Figura 6.9: Relações da entidade *Task*

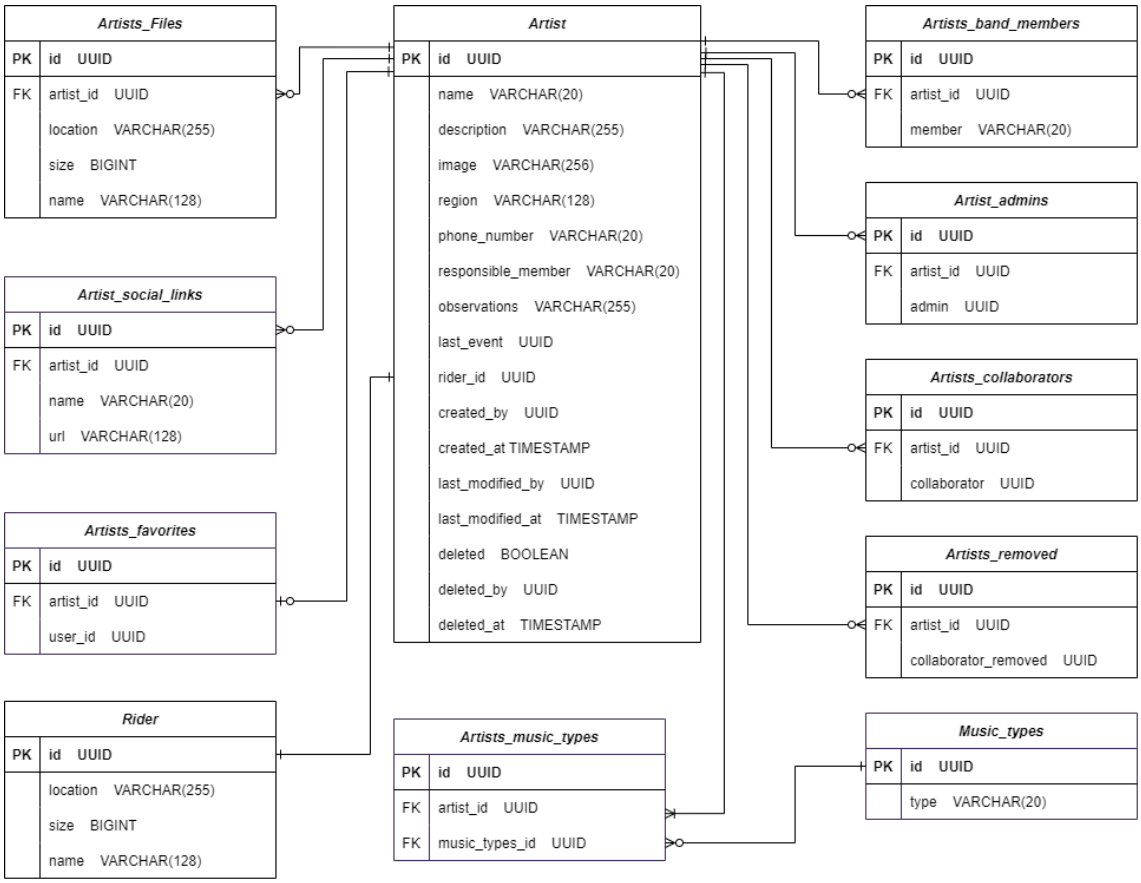

Figura 6.10: Relações da entidade *Artist*

Capítulo 7

Desenvolvimento

O Capítulo 7 é dedicado à descrição do desenvolvimento realizado durante o 2º semestre do presente ano letivo. Primeiramente, será descrito o processo de desenvolvimento, de seguida as funcionalidades desenvolvidas, a apresentação das componentes, a análise sobre a ocorrência dos riscos levantados na Secção 4.4 durante o desenvolvimento do projeto e, por fim, uma nota sobre o trabalho futuro a ser desenvolvido.

7.1 Processo de desenvolvimento

De forma a detalhar melhor o processo de desenvolvimento, este é dividido na organização da equipa e na organização de tarefas ao longo do semestre, detalhando como cada processo é executado.

7.1.1 Organização da equipa

A equipa de desenvolvimento foi composta pelos dois alunos dos dois estágios curriculares, *developers* da equipa, sendo cada um responsável pelas funcionalidades referentes aos respetivos estágios. Para além dos *developers*, a equipa é constituída pelo designer, equipa responsável pelo *code review* e orientadores dos estágios.

O *designer* da equipa é o responsável pela construção das *mockups* da plataforma, *mockups* desenvolvidas antes do início da fase de desenvolvimento. Posteriormente, o aluno implementou as funcionalidades com base nas tarefas geradas pelas *user stories*. No final da execução de cada tarefa, estas são movidas para *code review* onde uma equipa de *developers*, membros da empresa, verifica o código produzido. Após as correções levantadas no *code review* serem realizadas pelo aluno e validadas novamente pela equipa de *code review*, as tarefas são movidas para a fase de testes. Os testes foram realizados pelo aluno seguindo um plano de testes durante o *Sprint #9*.

Como a metodologia adotada se baseia em *Scrum*, as diferentes cerimónias - *daily*

meeting, *sprint planning*, *sprint review* e *sprint retrospective* - foram presenciadas ativamente pelos estagiários, designer, orientadores da empresa e equipa de *code review*.

7.1.2 Organização das tarefas

A ferramenta utilizada para a organização das tarefas durante os *sprints* foi a plataforma ***Linear*** [18]. Na plataforma foi criado um quadro com as seguintes colunas:

- ***Backlog*** - lista com as tarefas a serem desenvolvidas;
- ***Todo*** - listas com as tarefas a desenvolver durante o *sprint*;
- ***In progress*** - tarefas que estão em desenvolvimento;
- ***In Review*** - tarefas em *code review*;
- ***Ready to test*** - tarefas prontas a serem testadas;
- ***Done*** - tarefas que passaram nos testes, dadas como concluídas;
- ***Canceled*** - tarefas que não foram implementadas e foram canceladas com devida justificação.

No inicio de cada *sprint*, durante o *sprint planning*, o esforço para cada tarefa é calculado pelo aluno e validado pelo resto dos membros presentes na cerimónia. O esforço é definido com base nos números da sequência de *Fibonacci*, em que o padrão da sequência é 0, 1, 2, 3, 5, 8, 13... [15].

Os números representam o seguinte esforço por tarefa:

- **1** - tarefas que demorem meio dia;
- **2** - tarefas que demorem um dia;
- **3** - tarefas que demorem um dia e meio;
- **5** - tarefas que demorem dois dias e meio;
- **8** - tarefas que demorem quatro dias.

Por *sprint* foram atribuídos em média 18 pontos ao *developer* sendo que, durante o *sprint planning*, as tarefas são movidas para *Todo*, até estarem completos pelo menos 16 pontos. Ao longo do *sprint* o aluno é responsável por mover as tarefas entre listas, mediante o estado de execução da tarefa. Na Figura 7.1 está representado um exemplo do estado das tarefas na plataforma ***Linear*** durante a fase de desenvolvimento.

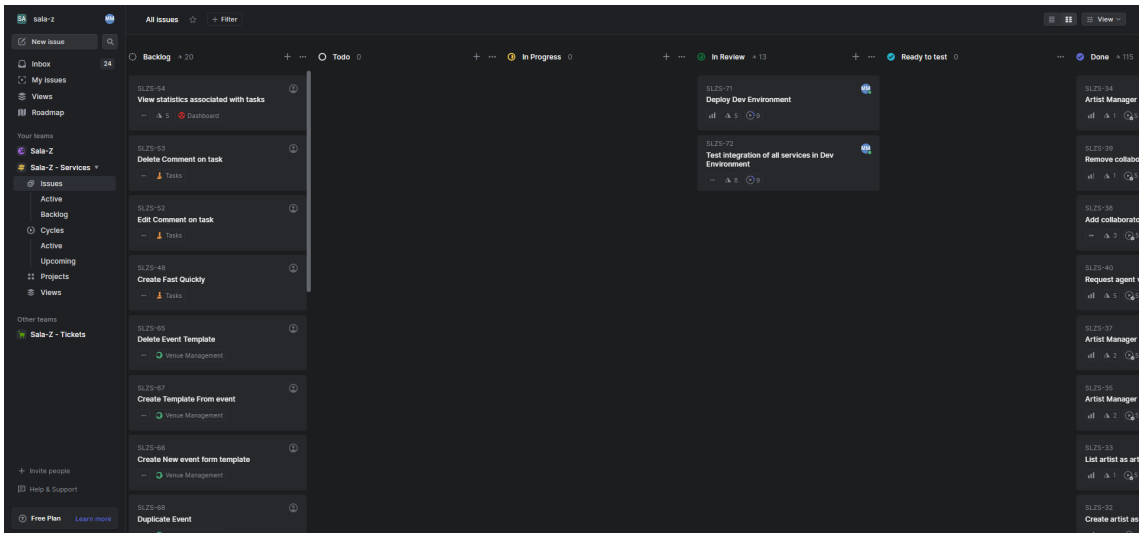

Figura 7.1: Exemplo da organização das tarefas durante o desenvolvimento no quadro da plataforma *Linear*

7.2 Requisitos funcionais desenvolvidos

Com base nos requisitos definidos previamente na Secção 5.3 foram listadas as funcionalidades desenvolvidas e não desenvolvidas durante o semestre nas seguintes tabelas.

Módulo: Gestão de *Venues*

Tema	Requisitos	Prioridade	Concluída
Painel de administração	Enviar convite a um novo utilizador para a minha sala	(M)	Sim
	Reenviar convite a um novo utilizador para a minha sala	(C)	Sim
	Listar utilizadores com acesso à *venue*	(M)	Sim
	Remover utilizador da *venue*	(M)	Sim

Tabela 7.1: Gestão de *Venues* - Painel de administração

Tema	Requisitos	Prioridade	Concluída
Dashboard	Ver as estatísticas associadas à *venue*	(S)	Não

Tabela 7.4: Gestão de *Venues* - *Dashboard*

Tema	Requisitos	Prioridade	Concluída
Gestão de inventário	Criar item no inventário	(M)	Sim
	Criar item no inventário com um número de série	(C)	Não
	Listar itens da *venue*	(M)	Sim
	Ordenar itens na lista de itens	(S)	Sim
	Pesquisar item na lista de itens	(S)	Sim
	Filtrar itens na lista de itens	(S)	Sim
	Ver detalhes de um item	(M)	Sim
	Editar item	(M)	Sim
	Duplicar item	(S)	Sim
	Eliminar item	(M)	Sim
	Associar um item a um evento	(C)	Sim

Tabela 7.2: Gestão de *Venues* - Gestão inventário

Tema	Requisitos	Prioridade	Concluída
Gestão de artistas	Criar artista com importação	(C)	Sim
	Criar artista manualmente	(M)	Sim
	Listar artistas	(M)	Sim
	Listar artistas favoritos	(C)	Sim
	Procurar artista na lista de artistas	(S)	Sim
	Filtrar lista de artistas	(S)	Sim
	Ver detalhes de um artista	(M)	Sim
	Editar artista	(M)	Sim
	Eliminar artista	(M)	Sim
	Marcar um artista como favorito	(C)	Sim
	Ver as estatísticas associadas a artistas	(S)	Não

Tabela 7.3: Gestão de *Venues* - Gestão artistas

Tema	Requisitos	Prioridade	Concluída
Gestão de tarefas de um evento	Rápida criação da tarefa	(C)	Não
	Criar tarefa com detalhes	(M)	Sim
	Listar tarefas	(M)	Sim
	Ver detalhes de uma tarefa	(C)	Sim
	Ordenar lista de tarefas	(S)	Sim
	Procurar tarefa na lista de tarefas	(S)	Sim
	Filtrar lista de tarefas	(S)	Sim
	Editar tarefa	(M)	Sim
	Duplicar tarefa	(S)	Sim
	Eliminar tarefa	(M)	Sim
	Ver a atividade associada à tarefa	(C)	Não
	Adicionar comentário à tarefa	(M)	Sim
	Editar comentário	(S)	Não
	Eliminar comentário	(S)	Não

Tabela 7.5: Gestão de *Venues* - Gestão de tarefas de um evento

Tema	Requisitos	Prioridade	Concluída
Gestão de tarefas sala/utilizador	Ver as estatísticas associadas a tarefas	(C)	Não

Tabela 7.6: Gestão de *Venues* - Gestão de tarefas sala/utilizador

Tema	Requisitos	Prioridade	Concluída
Comunicação entre *stakeholders*	Mencionar outros utilizadores da *venue* com comentários	(S)	Não

Tabela 7.7: Gestão de *Venues* - Comunicação entre *stakeholders*

Tema	Requisitos	Prioridade	Concluída
Notificações	Ver notificações	(S)	Não

Tabela 7.8: Gestão de *Venues* - Notificações

Tema	Requisitos	Prioridade	Concluída
Configurar relatórios de eventos	Configurar relatórios de eventos	(C)	Sim

Tabela 7.9: Gestão de *Venues* - Configurar relatórios de eventos

Tema	Requisitos	Prioridade	Concluída
Gerir templates de eventos	Criar template	(C)	Não
	Listar templates	(C)	Não
	Editar template	(C)	Não
	Eliminar template	(C)	Não
	Criar evento a partir de um template	(C)	Não
	Criar template a partir de um evento	(C)	Não

Tabela 7.10: Gestão de *Venues* - Gerir templates de eventos

Tema	Requisitos	Prioridade	Concluída
Gerir eventos	Duplicar evento	(C)	Não

Tabela 7.11: Gestão de *Venues* - Gerir eventos

Módulo: Gestão de Artistas

Tema	Requisitos	Prioridade	Concluída
Painel de administração	Enviar convite a um novo utilizador para o meu artista	(M)	Sim
	Reenviar convite a um novo utilizador para o meu artista	(C)	Sim
	Listar utilizadores com acesso ao meu artista	(M)	Sim
	Remover utilizador do artista	(M)	Sim

Tabela 7.12: Gestão de Artistas - Painel de administração

Tema	Requisitos	Prioridade	Concluída
Gestão de artistas	Criar artista com importação	(C)	Sim
	Criar artista manualmente	(M)	Sim
	Listar artistas	(M)	Sim
	Filtrar lista de artistas	(S)	Sim
	Ver detalhes de um artista	(M)	Sim
	Editar artista	(M)	Sim
	Eliminar artista	(M)	Sim
	Adicionar carimbo de verificação a artista	(S)	Sim

Tabela 7.13: Gestão de Artistas - Gestão artistas

Tema	Requisitos	Prioridade	Concluída
Dashboard	Ver as estatísticas associadas a artistas	(S)	Não
	Filtrar lista de artistas na tabela	(S)	Não
	Ordenar artistas na tabela	(S)	Não

Tabela 7.14: Gestão de Artistas - *Dashboard*

Tema	Requisitos	Prioridade	Concluída
Gestão de eventos	Listar eventos do artista	(M)	Sim
	Filtrar lista de eventos do artista	(S)	Sim
	Ver detalhes do evento	(M)	Sim

Tabela 7.15: Gestão de Artistas - Gestão de eventos

Tema	Requisitos	Prioridade	Concluída
Comunicação entre *stakeholders*	Mencionar outros utilizadores do evento com comentários	(S)	Não

Tabela 7.16: Gestão de Artistas - Comunicação entre *stakeholders*

Tema	Requisitos	Prioridade	Concluída
Notificações	Ver notificações	(S)	Não

Tabela 7.17: Gestão de Artistas - Notificações

Tema	Requisitos	Prioridade	Concluída
Configurar relatórios de eventos	Configurar relatórios de eventos	(C)	Sim

Tabela 7.18: Gestão de Artistas - Configurar relatórios de eventos

Através da análise das tabelas é possível verificar que as funcionalidades essenciais à plataforma foram desenvolvidas, sendo que todos os requisitos *Must Have* foram implementados. Porém, através da análise das tabelas é possível verificar que existem requisitos *Could Have* implementados ao invés de *Should Have*, que acrescentam valor à plataforma. Isto deve-se ao tempo de desenvolvimento ser curto que levou à tomada de decisão, entre aluno e orientador da empresa, de certas funcionalidades, mais adequado de implementar tendo em conta a aprendizagem do aluno e o estado da plataforma, serem implementadas.

Para além da adição do módulo de gestão de inventário, tarefas e artistas, foram adicionados novos utilizadores à plataforma como o colaborador da *venue*, o administrador e colaborador do *artista*. No enriquecimento da base de dados de artistas a fontes externas, o *Discogs* foi utilizado para a importação de dados de artistas integrado com o serviço ***Artist Management***. Ainda, foram efetuadas melhorias da comunicação entre gestores de artistas e de *venues* com a adição de comentários em eventos, por exemplo, com a receção de emails sempre que exista uma alteração associada a um evento.

7.3 Funcionalidades

Nesta secção são apresentados exemplos de ecrãs de funcionalidades desenvolvidas durante o presente estágio, os restantes ecrãs podem ser consultados no apêndice B. As funcionalidades desenvolvidas são apresentadas de seguida por temas.

Módulo: Gestão de *Venues* - Painel de Administração

No ecrã de edição da *venue*, funcionalidade que já existe na versão Sala-Z 2.0, foi adicionada a possibilidade de adicionar novamente utilizadores que foram removidos da mesma, Figura 7.2.

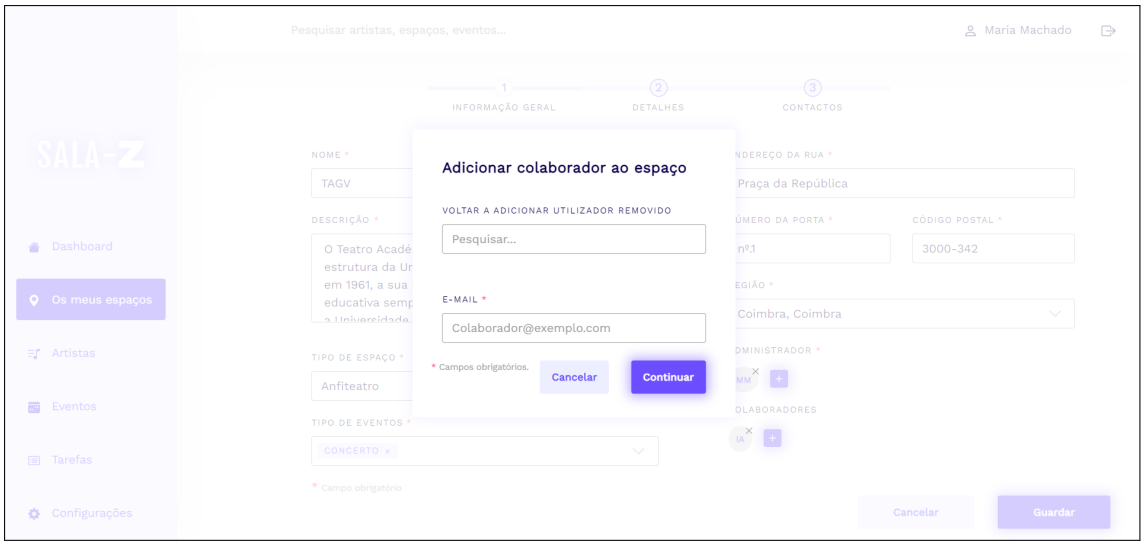

Figura 7.2: Adição de colaboradores novamente que foram removidos da *venue*

Módulo: Gestão de *Artistas* - Painel de Administração

Caso o utilizador autenticado seja uma administrador do artista, este poderá adicionar colaboradores ao seu artista, remover ou adicionar novamente um utilizador que tenha sido removido, Figura 7.3.

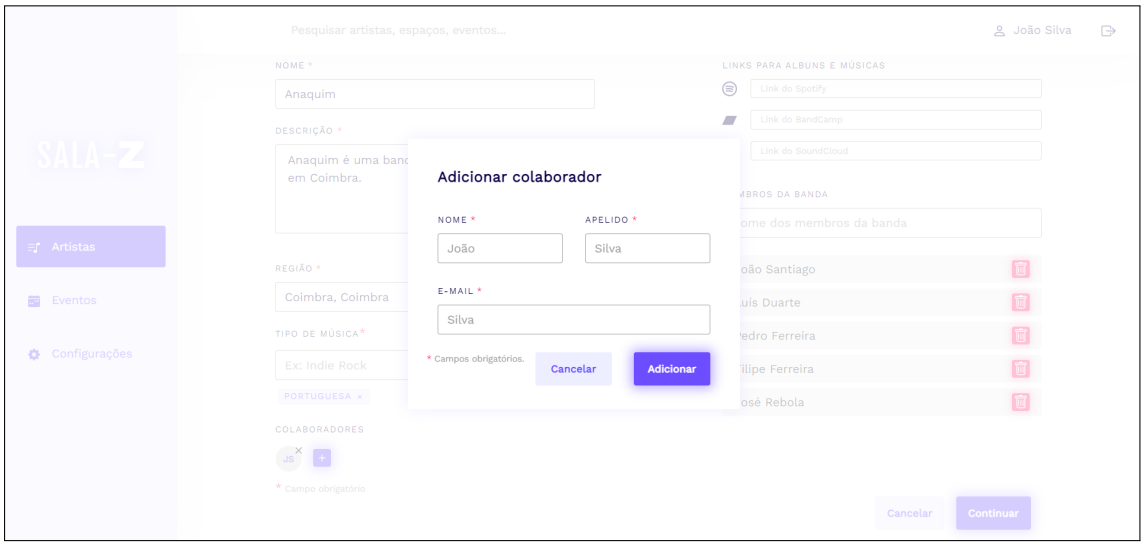

Figura 7.3: Gestão de colaboradores no artista

Gestão de Inventário

Na Figura 7.4 estão representados os items presentes no inventário das *venues* de um gestor de *venues*. Neste ecrã o utilizador pode filtrar e ordenar os itens bem como aceder aos ecrãs de edição, criação, duplicação e remoção de itens das suas *venues*.

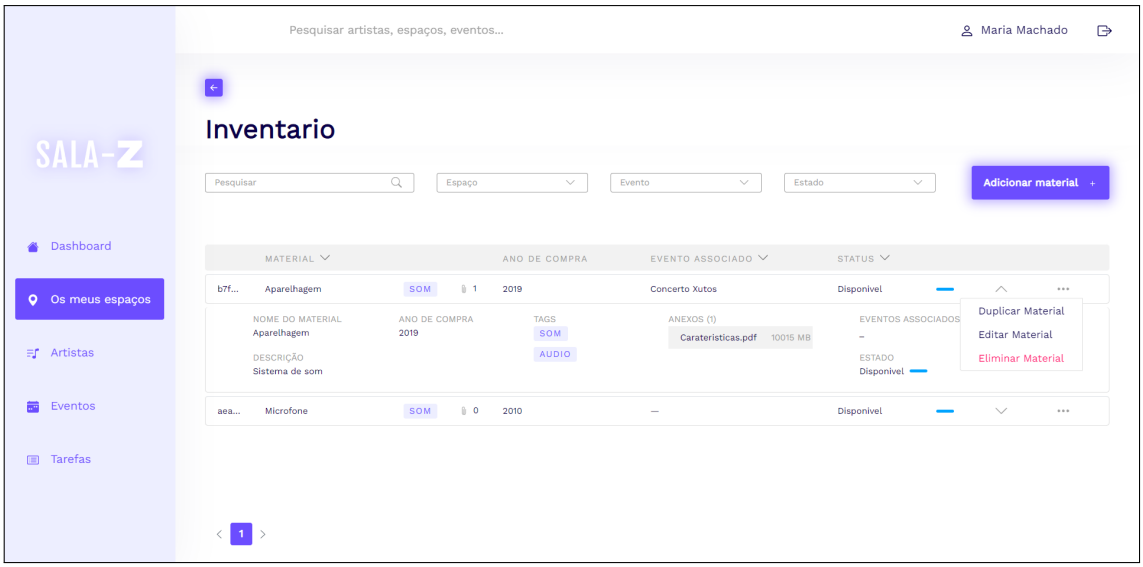

Figura 7.4: Lista de itens presentes no inventário

Gestão de Artistas

Na Figura 7.5 estão representados os artistas criados por um determinado utilizador, estes artistas podem posteriormente serem associados a eventos. O ecrã em termos de *design* é o mesmo para uma conta de gestor de artistas ou de *venues*.

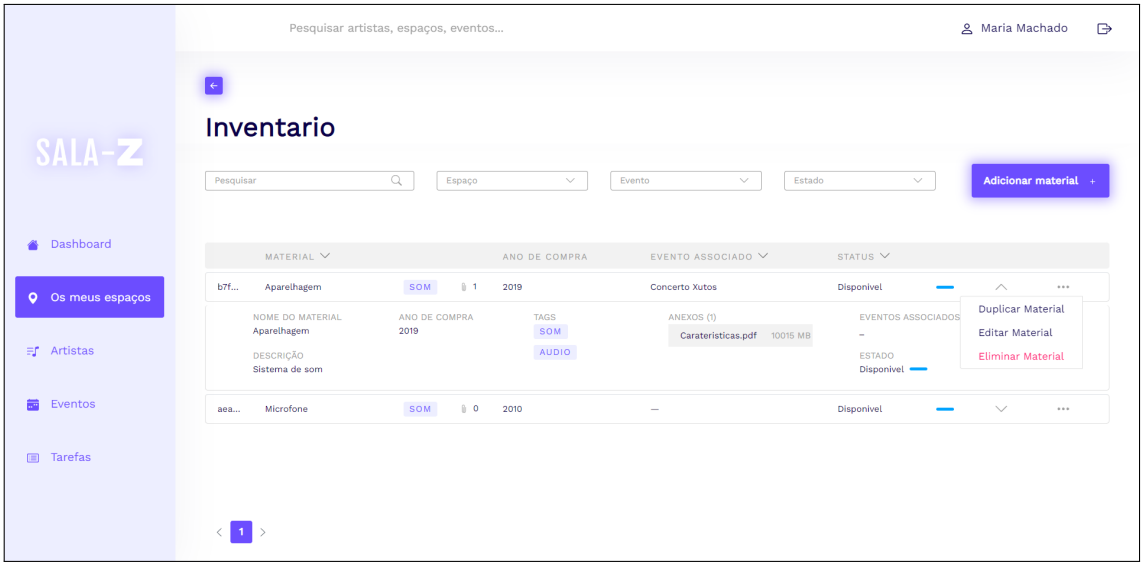

Figura 7.5: Lista de artistas

Gestão de Tarefas

Um gestor de uma *venue* poderá aceder à sua lista de tarefas associada aos seus eventos. A partir desta listagem poderá gerir cada uma das tarefas ou criar novas, Figura 7.6.

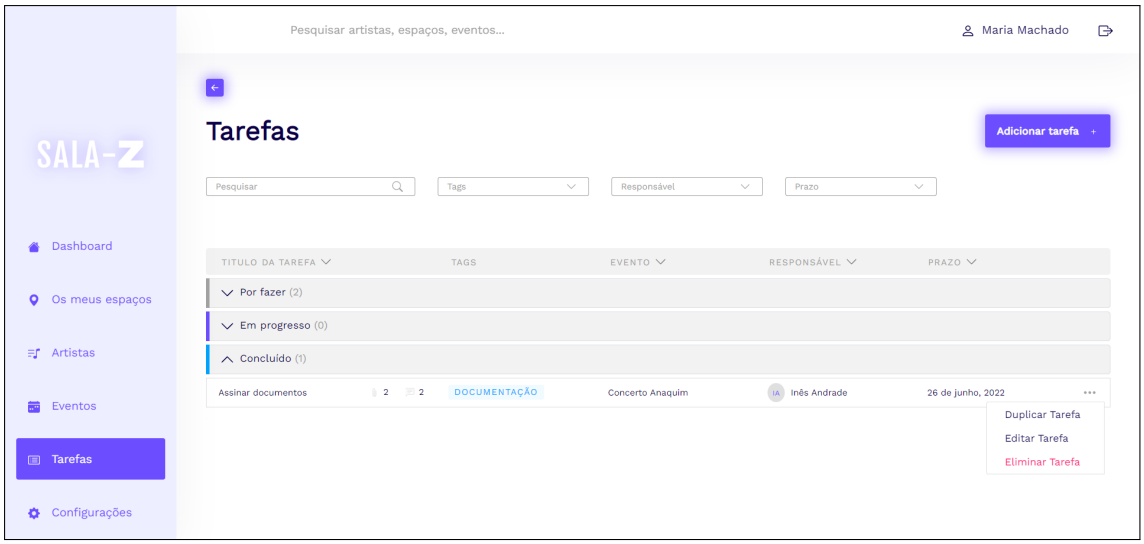

Figura 7.6: Lista de tarefas

Configuração de Relatórios de Eventos

Através da Figura 7.7 tem-se um exemplo da ativação de notificações frequentes de um gestor de artistas, caso este pretenda receber notificações sempre que um evento em que esteja associado seja alterado.

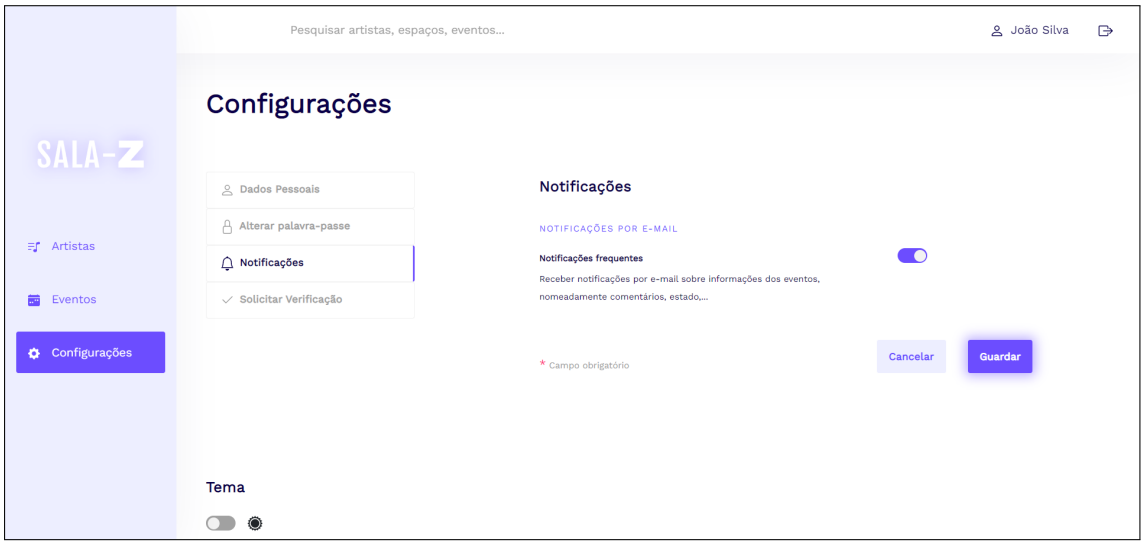

Figura 7.7: Configuração de Relatórios de um gestor de artistas

7.4 Análise dos riscos levantados

Previamente, na Secção 4.4 foram levantados os possíveis riscos associados ao desenvolvimento do projeto. De seguida será analisado se estes ocorreram e se estes foram mitigados com o plano de mitigação estabelecido:

1. Pandemia - Durante o primeiro semestre o aluno permaneceu em regime de trabalho remoto utilizando as ferramentas descritas no plano de mitigação, *Slack* e *Discord*, para reuniões, dúvidas e comunicação com membros da empresa. Durante o segundo semestre houve um levantamento das restrições e por opção, o aluno passou a regime de trabalho presencial. No entanto, os canais de comunicação continuaram a ser utilizados, por exemplo, para as reuniões.

2. Tecnologias - Ao longo do projeto qualquer dificuldade sentida com as tecnologias foi debatida entre aluno e orientador da empresa. Como as tecnologias são utilizadas por membros da empresa houve um grande apoio ao longo do desenvolvimento.

3. Falta de experiência - No início do desenvolvimento foram atribuídas tarefas ao aluno para a correção de *bugs* na plataforma. As tarefas também foram priorizadas, sendo que as mais simples foram atribuídas primeiro ao aluno para que este conheça tanto a plataforma como as tecnologias. Numa fase inicial existiu uma demora na execução das tarefas, contudo com o apoio da empresa e do plano de mitigação estabelecido, foi possível verificar ao longo dos *sprints* uma melhoria no tempo de execução das tarefas.

Apesar de no início do projeto ter-se realizado um levantamento dos riscos associados, existiram outros que foram encontrados ao longo do desenvolvimento. De seguida, estão descritos os riscos encontrados e os respetivos planos de mitigação para cada um:

4. *Code Review*

Descrição do risco: Atraso na realização do *code review* por parte da equipa de *review*. Isto levou ao acumulo de tarefas a serem concluídas no final do *sprint*.

Plano de Mitigação: Foram alocadas mais membros da empresa à equipa de *code review* para que o esforço exigido a cada elemento fosse menor e o processo mais rápido.

5. Plano de reestruturação da plataforma

Descrição do risco: A reestruturação da plataforma, migração de funcionalidades já existentes que não fazem parte do *scope* dos projetos de estágio para outros serviços, será realizada pelos dois estágios que ocorrem em paralelo. Assim, a realização desta reestruturação dependente da *performance* de cada um dos estagiários na execução das suas tarefas individuais.

Plano de Mitigação: Como a adição das funcionalidades referentes a cada estágio tem mais prioridade do que a migração de funcionalidades já existentes na plataforma. Caso não exista tempo suficiente para a reestruturação, as funcionalidades não serão migradas para outros serviços.

7.5 Trabalho Futuro

Apesar da maioria das funcionalidades se encontrarem desenvolvidas, o trabalho futuro da plataforma passa pela implementação dos requisitos funcionais que não foram desenvolvidos e a reestruturação da plataforma, com a migração de funcionalidades. A estrutura proposta neste estágio curricular apresenta uma mais-valia ao projeto e por isso a empresa pretende continuar com o seu desenvolvimento para que a estrutura da plataforma seja a proposta.

Em termos de requisitos não funcionais a portabilidade, possibilidade da plataforma ser acedida através de qualquer dispositivo móvel, não foi implementada devido ao curto espaço de tempo para o desenvolvimento. Assim, tornar a plataforma responsiva e suportável em ecrãs de diferentes dimensões é algo que também será implementado no futuro.

Para além do referido anteriormente, a empresa pretende colocar a plataforma em produção e convidar artistas e gestores de *venues* a testarem a plataforma para receção de *feedback* sobre possíveis melhorias.

Capítulo 8

Testing

O presente capítulo é dedicado à descrição dos testes efetuados na plataforma. Os testes ocorreram ao longo dos *sprints* e durante o *Sprint* #9 para que as funcionalidades fossem validadas. Primeiramente, serão descritos os testes funcionais efetuados sobre os requisitos funcionais definidos na Secção 5.3, seguidos dos testes sobre os requisitos não funcionais, definidos na Secção 5.4.

8.1 Testes funcionais

Ao longo do desenvolvimento, o plano de testes pretendido sofreu alterações. Numa fase inicial foram realizados testes de integração aos *endpoints*, recorrendo à ferramenta **JUNIT** [65] como referido anteriormente. Na Figura 8.1 encontra-se representado um teste de integração sobre um *endpoint*, do gestor de inventário, que retorna a lista de itens presentes numa *venue*, recorrendo a *unit testing*.

```
@Test
public void getAll() {
    ValidatableResponse response = given() RequestSpecification
            .header( s: "Content-Type", MediaType.APPLICATION_JSON)
            .header( s: "Authorization", token)
            .when().get( s: "/resources") Response
            .then() ValidatableResponse
            .statusCode(200)
            .body( s: "payload.list.size()", is( value: 10),
                    ...objects: "category", is(Category.SUCCESSFUL.toString()));
}
```

Figura 8.1: Teste de integração sobre o *endpoint* que retorna a lista de recursos

Para que o processo de testagem fosse mais rápido, optou-se por após a construção de um *endpoint* no *backend* realizarem-se testes funcionais manuais recorrendo à plataforma *Swagger* [80]. Na Figura 8.2 encontra-se um exemplo de um teste a

um *endpoint* com recurso à ferramenta *Swagger*, neste caso à listagem de todos os artistas.

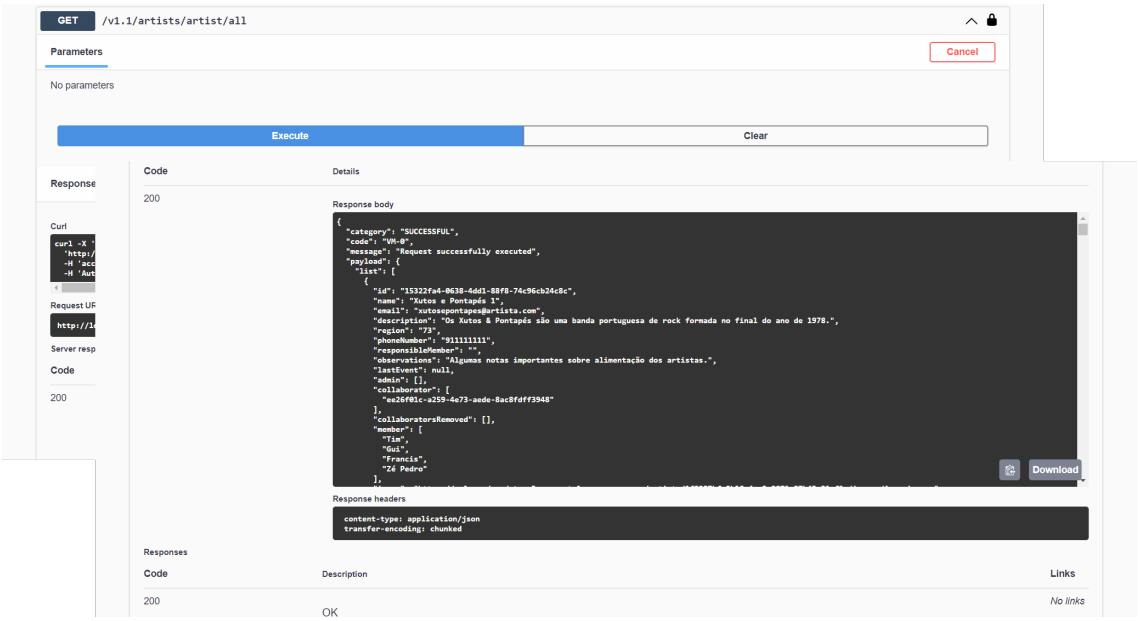

Figura 8.2: Teste de um *endpoint* com recurso à ferramenta *Swagger*

No final do desenvolvimento durante o *Sprint* #9, foram realizados testes de integração que consistem em *smoke tests* do *frontend* com o *backend* da plataforma. Por forma a testar a plataforma recorrendo a *smoke tests*, foi construída uma tabela com as seguintes colunas:

- **US** - ID da *User Story (US)* sobre o qual está a ser realizado o teste.
- **ID** - ID do teste.
- **Cenário** - Cenário do teste.
- **Descrição** - Descrição do teste.
- **Passos** - Passos executados durante o teste.
- **Resultado Esperado** - Resultado esperado após a execução dos passos.
- **Resultado Obtido** - Resultado obtido após a execução dos passos.
- **Resultado** - Se o teste passou ou falhou.

De seguida é apresentado um excerto das tabelas de testes que podem ser consultadas no Apêndice C.

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
1	1	Adicionar um novo utilizador à *venue*	Adicionar um novo utilizador	- Editar uma *venue* - Selecionar que quer adicionar um novo colaborador - Inserir campos necessários - Adicionar novo colaborador	Convite enviado para o novo utilizador	Convite enviado para o novo utilizador	**Passou**
2	2	Reenviar convite a um novo utilizador para a *venue*	Reenviar novamente convite a um utilizador removido	- Editar uma *venue* - Selecionar que quer adicionar um novo colaborador - Selecionar *dropdown* para adicionar novamente utilizador removido - Campos preenchidos automaticamente após seleção - Adicionar novo colaborador - Guardar as alterações efetuadas	Utilizador novamente adicionado ao espaço	Utilizador novamente adicionado ao espaço	**Passou**
3	3	Listar utilizadores com acesso à *venue*	Listar utilizadores com acesso à *venue*	- Selecionar uma *venue* - Ver todos os utilizadores da *venue*	Ver a lista de utilizadores com acesso à *venue*	Ver a lista de utilizadores com acesso à *venue*	**Passou**
4	4	Remover utilizador da *venue*	Remover utilizador da *venue*	- Editar uma *venue* - Ver todos os utilizadores da *venue* - Selecionar um utilizador para ser removido - Confirmar remoção do utilizador - Guardar as alterações efetuadas	*Venue* com lista de utilizadores exceto o removido	*Venue* com lista de utilizadores exceto o removido	**Passou**

Tabela 8.1: Excerto das tabelas de testes

8.2 Testes não funcionais

Nesta Seccção são descritos os testes executados para garantir que os requisitos não funcionais definidos na secção 5.4 são cumpridos. Os testes foram realizados durante o desenvolvimento, sendo que durante o *Sprint* #9 foram executados novamente, por forma a garantir a qualidade do sistema.

Escabilidade

No *Sprint* #9 foram realizados testes para avaliar a escabilidade e *performance* do sistema. Os testes consistiram no envio de pedidos com recurso à ferramenta

Apache JMeter [9]. Nos testes aumentaram-se o número de pedidos e o número de utilizadores sobre a mesma funcionalidade, de forma a verificar como o sistema reage e se adapta em alturas de maior *stress*.

De seguida encontra-se um exemplo de um teste realizado sobre a *API* do novo serviço do Sala-Z 2.1, *Artist Management*, sobre o *endpoint* que retorna todos os artistas da plataforma. Através da análise da Tabela 8.2 verifica-se que o tempo médio e máximo da resposta são inferiores a 3 segundos, sendo o máximo atingido 2,058 segundos. Uma vez que estes valores se aproximam dos 3 segundos, tempo máximo de resposta definido no cenário de atributo de qualidade da escabilidade na Secção 5.4, no futuro, deve encontrar-se soluções para melhorar a *performance* do sistema.

Número de utilizadores	Número de pedidos	Min (ms)	Médio (ms)	Max (ms)
10	10	57	82	259
100	20	58	126	675
200	30	58	198	941
300	40	57	308	1195
400	50	59	374	2058

Tabela 8.2: Tempos de resposta dos testes realizados ao microserviço *Artist Management*

Portabilidade

Durante o desenvolvimento optou-se por este requisito não funcional não ser assegurado, devido ao curto espaço de tempo para desenvolvimento. A aluna e orientador da empresa decidiram que o foco do estágio seria o desenvolvimento das funcionalidades para ecrãs *desktop*, no entanto a empresa pretende implementar responsividade e testes sobre este requisito não funcional no futuro.

Segurança

Para garantir o cenário de atributo de qualidade de segurança descrito na Subsecção 5.4.4 foi executado um plano de testes apresentados nas Tabelas 8.3, 8.4, 8.5 e 8.6. Os planos foram construídos com base nos diferentes perfis de utilizadores presentes na plataforma: administrador do Sala-Z, utilizador da *venue*, utilizador do artista e utilizador não autenticado, sendo que o utilizador da *venue* e do artista englobam tanto perfis de administradores como de colaboradores. Para cada tipo de perfil, foi estabelecido um plano de testes com base nas rotas que estes utilizadores não têm permissão de acesso, por exemplo, um administrador do Sala-Z autenticado, não tem permissão para aceder ao portal do artista ou ao portal da *venue*. O portal do artista consiste na primeira página que um utilizador do artista acede após o *login*, já o portal da *venue* é a primeira página acedida por um utilizador da *venue*.

ID	Cenário	Passos	Resultado Esperado	Resultado Obtido	Resultado
75	Utilizador não autenticado tenta aceder ao *backoffice*	- Utilizador não autenticado tenta aceder a uma rota do *backoffice*	Utilizador é redirecionado para o ecrã de *login* da plataforma	Utilizador é redirecionado para o ecrã de *login* da plataforma	**Passou**
76	Utilizador não autenticado tenta entrar no portal do utilizador do artista	- Utilizador não autenticado tenta aceder a uma rota do portal do utilizador do artista	Utilizador é redirecionado para o ecrã de *login* da plataforma	Utilizador é redirecionado para o ecrã de *login* da plataforma	**Passou**
77	Utilizador não autenticado tenta entrar no portal do utilizador da *venue*	- Utilizador não autenticado tenta aceder a uma rota do portal do utilizador da *venue*	Utilizador é redirecionado para o ecrã de *login* da plataforma	Utilizador é redirecionado para o ecrã de *login* da plataforma	**Passou**

Tabela 8.3: Testes não funcionais de segurança por um utilizador não autenticado

ID	Cenário	Passos	Resultado Esperado	Resultado Obtido	Resultado
78	Utilizador de uma *venue* tenta entrar no *backoffice*	- Utilizador faz *login* na plataforma como utilizador da *venue* - Utilizador da *venue* tenta aceder a uma rota do *backoffice*	É apresentado o portal do utilizador da *venue*	É apresentado o portal do utilizador da *venue*	**Passou**
79	Utilizador de uma venue tenta entrar no portal do utilizador do artista	- Utilizador faz *login* na plataforma como utilizador da *venue* - Utilizador da *venue* tenta aceder a uma rota do portal do utilizador do artista	É apresentado o portal do utilizador da *venue*	É apresentado o portal do utilizador da *venue*	**Passou**

Tabela 8.4: Testes não funcionais de segurança por um utilizador da *venue*

ID	Cenário	Passos	Resultado Esperado	Resultado Obtido	Resultado
80	Utilizador do artista tenta entrar no *backoffice*	- Utilizador faz *login* na plataforma como utilizador do artista - Utilizador da artista tenta aceder a uma rota do *backoffice*	É apresentado o portal do utilizador da artista	É apresentado o portal do utilizador da artista	**Passou**
81	Utilizador do artista tenta entrar no portal do utilizador da *venue*	- Utilizador faz *login* na plataforma como utilizador do artista - Utilizador do artista tenta aceder a uma rota do portal do utilizador da *venue*	É apresentado o portal do utilizador do artista	É apresentado o portal do utilizador do artista	**Passou**

Tabela 8.5: Testes não funcionais de segurança por um utilizador do artista

ID	Cenário	Passos	Resultado Esperado	Resultado Obtido	Resultado
82	Administrador do Sala-Z tenta entrar no portal do utilizador do artista	- Utilizador faz *login* na plataforma como administrador do Sala-Z - Administrador Sala-Z tenta aceder a uma rota do portal do utilizador do artista	É apresentado o *backoffice*	É apresentado o *backoffice*	**Passou**
83	Administrador do Sala-Z tenta entrar no portal do utilizador da *venue*	- Utilizador faz *login* na plataforma como administrador do Sala-Z - Administrador Sala-Z tenta aceder a uma rota do portal do utilizador da *venue*	É apresentado o *backoffice*	É apresentado o *backoffice*	**Passou**

Tabela 8.6: Testes não funcionais de segurança por um administrador do Sala-Z

A fim de testar a plataforma Sala-Z 2.1, foi realizado um teste recorrendo à ferramenta *OWASP Zed Attack Proxy (ZAP)* [93]. A ferramenta fornece um teste automático que gera relatórios que permitem identificar problemas de segurança com detalhes sobre as vulnerabilidades, através de testes de penetração ao *website* [42]. O teste consistiu na realização de um teste automático sobre o endereço da plataforma onde foram detetados 7 tipos de alertas, Figura 8.3. Os alertas são apresentados com um risco associado, bandeira que é apresentada no tipo de alerta, Figura 8.3. Podemos verificar que os riscos encontrados com o *automated scan* são médios (bandeira laranja), baixos (bandeira amarela) e informativos (bandeira azul) [45], não existindo nenhum risco alto.

Devido ao curto espaço de tempo para a realização dos testes não funcionais durante o *Sprint* #9, não foi possível corrigir os alertas. No entanto, a plataforma deverá assegurar que estes sejam corrigidos tornando a plataforma mais segura e menos suscetível a potenciais ataques no futuro.

De seguida, serão analisados tipos de alertas encontrados e potenciais soluções para a sua correção:

- ***Content Security Policy (CSP)***: É uma *layer* de segurança que ajuda a prever ataques, incluindo *Cross Site Scripting* e *Data Injection* [79]. Estes ataques são utilizados, por exemplo, para roubo de dados ou distribuição de *malware* [79]. O *CSP* fornece um conjunto de *headers* que permite que os donos do *website* declarem conteúdo que os *browsers* podem carregar na página [79].

  Solução: Configurar o servidor *web* para retornar o *Header Content-Security-Policy HTTP* para controlar os recursos do *browser* que podem ser permitidos carregar na página [79].

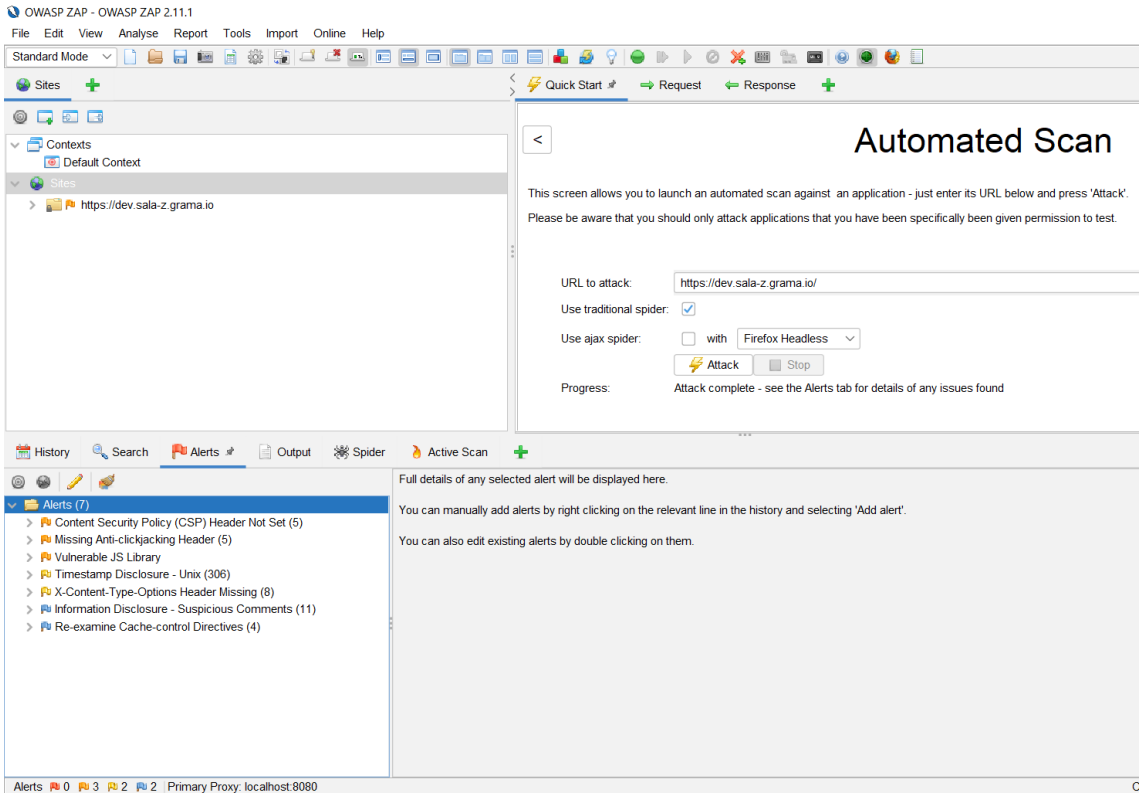

Figura 8.3: *OWASP ZAP - Automated Scan* sobre o Sala-Z 2.1

- ***Missing Anti-Clicking Jacking Header***: a resposta não inclui nem o *Content-Security-Policy* com diretivas *frame-ancestors* ou *X-Frame-Options* que protejam contra ataques *ClickJacking* [53].

  Solução: A *origin* deve estar propriamente especificada no *header Access-Control-Allow-Origin* [53]. Apenas *websites* confiáveis precisam que este recurso seja bem especificado neste *header*, com suporte para o protocolo mais seguro [53].

- ***Vulnerable JS Library***: uma vulnerabilidade numa biblioteca *javascript* pode ser utilizada para ataques *Cross-Site Scripting*, *Cross-Site Request Forgery* e *Buffer Overflow* [68].

  Solução: Manter a versão da biblioteca *javascript* sempre atualizada e remover bibliotecas que não são utilizadas são exemplos de soluções que previnem o *website* contra esta vulnerabilidade [68].

- ***Timestamp Disclosure Unix***: é uma forma de saber o *timestamp* pelo servidor que pode ser usado para recuperar informações confidenciais, como por exemplo, quando usado como um *salt* ou um *token* durante a autenticação ou encriptação [89].

  Solução: O código da aplicação deverá ser modificado para que este não retorne informações sobre o *timestamp* do servidor [89].

- ***X-Content-Type-Options Header Missing***: o *header Anti-MIME-Sniffing*, *X-Content-Type-Options* não foi definido como *nosniff*. Assim, permite que versões antigas de *browsers* como *Internet Explore* e *Chrome* realizem *MIME-sniffing* para que as respostas apresentem outro conteúdo do apresentado pela plataforma [70].

  Solução: Assegurar que a plataforma define o *Content-Type header* de forma apropriada e o *header X-Content-Type-Options* para "*nosniff*"em todas as páginas *web* [70].

- ***Information Disclosure - Suspicious Comments***: as respostas apresentam comentários suspeitos que podem ajudar o atacante [55].

  Solução: Remover toda a informação que possa ajudar o atacante [55].

- ***Re-examine Cache - control Directives***: o *header cache-control* não foi definido apropriadamente ou está em falta, permitindo que o *browser* e *proxies* armazenem conteúdo em *cache* [60]. Conteúdos estáticos como *css*, *js* e imagens podem ser intencionais mas deve-se verificar e assegurar que não é armazenado em *cache* informação sensível [60].

  Solução: Assegurar que para conteúdo sensível, o *header cache-control HTTP* é definido como "*no-cache*, *no-store*, *must-revalidate*"[60].

Após a análise dos alertas e soluções, foi verificado se as vulnerabilidades presentes no documento *OWASP Top 10* são asseguradas. O *OWASP Top 10* é um documento que contém os riscos de segurança mais críticos para as aplicações *web* [81]. Por exemplo, analisando a vulnerabilidade *A1 Access Broken Link* verifica-se que da lista de alertas que apresentam riscos para esta vulnerabilidade temos o *Information Disclosure - Suspicious Comments* [75]. Assim sendo, uma das prioridades no futuro da plataforma passa pela correção destes alertas e se as vulnerabilidades presentes no documento não estão presentes na plataforma.

Capítulo 9

Conclusão

A realização do presente estágio curricular permitiu ao aluno adquirir novos conhecimentos, com a procura de novas soluções a nível arquitetural e durante o decorrer do desenvolvimento, na descoberta de novas tecnologias e aplicação dos conhecimentos adquiridos durante o Mestrado em Engenharia Informática. O estágio proporcionou a integração do aluno numa empresa, contactando de perto com uma equipa e profissionais da área, preparando-se melhor para que no futuro entre no mundo do trabalho.

Numa fase inicial, o estágio começou pela exploração da plataforma com uma análise atual sobre as funcionalidades que esta disponibiliza, a sua arquitetura e interface.

No levantamento do estado de arte, inicialmente, analisaram-se concorrentes diretos, indiretos e potenciais permitindo ao aluno e à empresa saber o que distingue a plataforma dos restantes concorrentes. De seguida, analisaram-se soluções para a adição das novas funcionalidades, se a abordagem a seguir seria uma arquitetura monolítica ou baseada em microsserviços, optando pela segunda. Posteriormente, analisaram-se os padrões associados a uma arquitetura baseada em microsserviços, referindo os que são utilizados na última versão da plataforma, e tomadas de decisão sobre aqueles que serão utilizados na nova arquitetura. Seguidamente, comparam-se diferentes tecnologias tanto a nível de *frontend* como *backend*, na adição dos novos serviços, com a escolha de *Quarkus* no *backend* e *Vue* no *frontend*.

No primeiro semestre foi selecionada a metodologia de desenvolvimento de *software* que melhor se adequa ao projeto e realizado o planeamento das tarefas durante o ano curricular, com base na metodologia adotada. Por fim, foi realizado o levantamento dos riscos associados ao desenvolvimento e a criação de planos de mitigação que permite criar estratégias previamente, caso os riscos levantados ocorram.

De seguida, foram levantados os requisitos funcionais do projeto com base nas *user stories* criadas parcialmente pelo aluno, e priorizados seguindo o método de MoScoW [20]. Para além dos requisitos funcionais, foram definidos também os requisitos não funcionais do sistema construindo cenários de atributos de quali-

dade.

Após o levantamento dos requisitos, desenhou-se a arquitetura do projeto com base nos conhecimentos adquiridos previamente ao longo do semestre, seguindo uma arquitetura baseada em microsserviços. Seguidamente, construíu-se um diagrama entidade-relacionamento que permite uma melhor visualização das entidades e relações estabelecidas entre si.

O segundo semestre passou pela implementação dos requisitos funcionais levantados, tendo em conta a priorização estabelecida na construção das tarefas para cada *sprint*. No final da implementação durante um *sprint* foram realizados testes de aceitação e o sistema foi avaliado, concluindo se este cumpre os requisitos funcionais e não funcionais estabelecidos. No final do desenvolvimento foram analisados se o riscos levantados ocorreram e descritos os riscos encontrados ao longo do desenvolvimento bem como os planos de mitigação utilizados.

Por fim, a realização do presente estágio curricular permitiu ao aluno adquirir novas competências, ferramentas de trabalho e conhecimentos. A integração deste na empresa permitiu conhecer novos métodos de trabalho e a dinâmica de trabalhar numa equipa. Em suma, a realização do estágio tanto do ponto de vista do aluno como da empresa revelou-se um sucesso.

Referências

[1] 10 best web development frameworks. https://hackr.io/blog/web-development-frameworks.

[2] Advantages and disadvantages of scrum methodology. https://www.projectpractical.com/advantages-and-disadvantages-of-scrum-methodology/.

[3] Agile manifesto. https://www.scrumportugal.pt/agile-manifesto/.

[4] Amazon api gateway. https://aws.amazon.com/pt/api-gateway/.

[5] Amazon s3. https://aws.amazon.com/s3/.

[6] Amazon simple notification service. https://aws.amazon.com/sns/.

[7] Amazon simple queue service. https://aws.amazon.com/sqs/.

[8] Angular - repositório github. https://github.com/angular/. Último acesso: 4-1-2022.

[9] Apache jmeter. https://jmeter.apache.org/usermanual/properties_reference.html.

[10] Backends for frontends pattern. https://docs.microsoft.com/en-us/azure/architecture/patterns/backends-for-frontends.

[11] Computação em nuvem com a aws. https://aws.amazon.com/pt/what-is-aws/?nc1=f_cc.

[12] Discord. https://discord.com/.

[13] Estilo de arquitetura de microsserviços. https://docs.microsoft.com/pt-pt/azure/architecture/guide/architecture-styles/microservices.

[14] Expressjs - repositório github. https://github.com/expressjs/express. Último acesso: 4-1-2022.

[15] Fibonacci scale in agile estimation. https://www.wrike.com/blog/fibonacci-scale-in-agile-estimation/.

[16] How fast should a website load? https://www.bluecorona.com/blog/how-fast-should-website-be//.

[17] Iso/iec 25010. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?start=6.

[18] Linear. https://linear.app/.

[19] Microsserviços. https://aws.amazon.com/pt/microservices/.

[20] Moscow prioritization. https://www.productplan.com/glossary/moscow-prioritization/.

[21] Pattern: Access token. https://microservices.io/patterns/security/access-token.html.

[22] Pattern: Api gateway / backends for frontends. https://microservices.io/patterns/apigateway.html.

[23] Pattern: Messaging. https://microservices.io/patterns/communication-style/messaging.html.

[24] Pattern: Microservice architecture. https://microservices.io/patterns/microservices.html.

[25] Pattern: Multiple service instances per host. https://microservices.io/patterns/deployment/multiple-services-per-host.html.

[26] Pattern: Remote procedure invocation (rpi). https://microservices.io/patterns/communication-style/rpi.html.

[27] Pattern: Service instance per vm. https://microservices.io/patterns/deployment/service-per-vm.html.

[28] Pattern: Single service instance per host. https://microservices.io/patterns/microservices.html.

[29] Pros and cons of using spring boot. https://scand.com/company/blog/pros-and-cons-of-using-spring-boot/.

[30] Quarkus - repositório github. https://github.com/quarkusio/quarkus. Último acesso: 4-1-2022.

[31] React - repositório github. https://github.com/facebook/react. Último acesso: 4-1-2022.

[32] Scrum – exemplo prático. https://tecnologia.culturamix.com/dicas/scrum-exemplo-pratico.

[33] Service decomposition patterns in microservices. https://anjireddy-kata.medium.com/service-decomposition-patterns-in-microservices-649cfec42dc5.

[34] Slack. https://slack.com/intl/pt-pt/.

[35] Software development process. https://courses.lumenlearning.com/zeliite115/chapter/reading-software-development-process/.

[36] Spring boot - repositório github. https://github.com/spring-projects/spring-boot. Último acesso: 4-1-2022.

[37] Spring boot vs quarkus. https://javasterling.com/spring-boot/spring-boot-vs-quarkus/.

[38] Vue.js - repositório github. https://github.com/vuejs/vue. Último acesso: 4-1-2022.

[39] What are the three scrum roles? https://www.visual-paradigm.com/scrum/what-are-the-three-scrum-roles/.

[40] What is an epic? https://www.zoho.com/sprints/what-are-epics.html.

[41] What is quarkus? https://www.redhat.com/en/topics/cloud-native-apps/what-is-quarkus.

[42] OWASP ZAP: a powerful tool to discover Websites vulnerabilities. https://www.guruadvisor.net/en/50-sicurezza/845-owasp-zap-a-powerful-tool-to-discover-websites-vulnerabilities.

[43] Viral Agenda. https://www.viralagenda.com/pt/coimbra/concerts.

[44] Omar Al-Debagy and Peter Martinek. A comparative review of microservices and monolithic architectures. In *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 000149–000154. IEEE, 2018.

[45] Alerts. zaproxy.org/docs/desktop/start/features/alerts/.

[46] Nikolay Ashanin. https://medium.com/@nvashanin/quality-attributes-in-software-architecture-3844ea482732. https://medium.com/@nvashanin/quality-attributes-in-software-architecture-3844ea482732.

[47] Elastic Load Balancing. https://aws.amazon.com/pt/elasticloadbalancing/.

[48] Bandcamp. https://bandcamp.com.

[49] Spring Boot. https://spring.io/projects/spring-boot.

[50] Mara Calvello. Scrum meetings: Types, benefits, and tips to get it done right. https://fellow.app/blog/meetings/scrum-meetings-types-benefits-and-tips/?utm_source=google&utm_medium=cpc&utm_campaign=Dynamic_Campaign_Iberia&utm_term=-&utm_content=529656184702&adgroupid=124079959157&gclid=CjwKCAiAh_GNBhAHEiwAjOh3ZIbWi7kc3lRWEyBq6WjraQFhB_JviSpIXU31Dmoik_CVag9w2f2jRxoC3p4QAvD_BwE#daily.

[51] Chandana. Scrum project management: Advantages and disadvantages. https://www.simplilearn.com/scrum-project-management-article.

[52] Fred Churchville. 5 core components of microservices architecture. https://searchapparchitecture.techtarget.com/feature/5-core-components-of-microservices-architecture.

[53] Missing Anti clickjacking Header. https://www.zaproxy.org/docs/alerts/10020-1/.

[54] Amazon Cognito. https://aws.amazon.com/cognito/.

[55] Information Disclosure Suspicious Comments. https://www.zaproxy.org/docs/alerts/10027/.

[56] Hiren Dhaduk. Best frontend frameworks of 2022 for web development. https://www.simform.com/blog/best-frontend-frameworks/.

[57] Discogs. https://www.discogs.com.

[58] Recursos do Amazon EC2. https://aws.amazon.com/pt/ec2/features/?trk=ec2_landing.

[59] Amazon EC2. https://aws.amazon.com/ec2/.

[60] Re examine Cache-control Directives. https://www.zaproxy.org/docs/alerts/10015/.

[61] Express. https://expressjs.com/.

[62] Gigplanner. https://gigplanner.com/#features.

[63] Gigwell. https://www.gigwell.com.

[64] Manishaben Jaiswal. Software architecture and software design. *International Research Journal of Engineering and Technology (IRJET) e-ISSN*, pages 2395–0056, 2019.

[65] JUNIT. https://junit.org/junit5/.

[66] Tejas Kaneriya. Advantages & disadvantages of node.js : Why to use node.js? https://www.simform.com/blog/nodejs-advantages-disadvantages/#section13.

[67] Dawid Karczewski. What are the best frontend frameworks to use in 2021? https://www.ideamotive.co/blog/best-frontend-frameworks.

[68] Vulnerable Javascript Library. https://beaglesecurity.com/blog/vulnerability/vulnerable-javascript-library.html.

[69] Indusree Mavuru. Traditional vs. agile software development methodologies. https://www.kpipartners.com/blog/traditional-vs-agile-software-development-methodologies.

[70] X-Content-Type-Options Header Missing. https://www.zaproxy.org/docs/alerts/10021/.

[71] C4 model. https://c4model.com/.

[72] SDLC Waterfall Model. https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm.

[73] Muzeek. https://muzeek.com.

[74] Bianca Minetto Napoleão. Matriz de riscos (matriz de probabilidade e impacto). https://ferramentasdaqualidade.org/matriz-de-riscos-matriz-de-probabilidade-e-impacto/.

[75] Alert Tag: OWASP_2021_A01. https://www.zaproxy.org/alerttags/owasp_2021_a01/.

[76] Prism.fm. https://prism.fm/product/.

[77] React. https://reactjs.org/docs/getting-started.html.

[78] Plataforma Sala-Z. https://sala-z.grama.io/. Último acesso: 17-1-2022.

[79] Content Security Policy (CSP) Header Not Set. https://www.zaproxy.org/docs/alerts/10038/.

[80] Plataforma Swagger. https://swagger.io/. Último acesso: 27-6-2022.

[81] OWASP Top Ten. https://www.zaproxy.org/alerttags/owasp_2021_a01/.

[82] Smoke Tests. https://www.techtarget.com/searchsoftwarequality/definition/smoke-testing. Último acesso: 27-6-2022.

[83] Exemplo *board* Trello. https://blog.trello.com/br/como-ser-lider-equipe-trabalho.

[84] Theatron. https://www.theatron.eu/about.

[85] Dave Todaro. What is a user story? https://ascendle.com/ideas/what-is-a-user-story/.

[86] JSON Web Token. https://jwt.io/introduction.

[87] Trello. https://trello.com/tour.

[88] Stack Overflow Trends. https://insights.stackoverflow.com/trends?tags=reactjs%2Cangular%2Cvue.js%2Cember.js%2Cbackbone.js. Último acesso: 10-1-2022.

[89] Timestamp Disclosure Unix. https://scanrepeat.com/web-security-knowledge-base/timestamp-disclosure---unix.

[90] All Venues. https://www.allmusicvenues.com/.

[91] Mario Villamizar, Oscar Garcés, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *2015 10th Computing Colombian Conference (10CCC)*, pages 583–590. IEEE, 2015.

[92] Muhammad Waseem, Peng Liang, Gastón Márquez, and Amleto Di Salle. Testing microservices architecture-based applications: A systematic mapping study. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*, pages 119–128. IEEE, 2020.

[93] OWASP ZAP. `https://www.zaproxy.org/`.

Apêndices

Apêndice A

User Stories

Módulo: Gestão de *Venues*

1. Painel de Administração

   ES-1: **Como um** administrador da *venue* autenticado, **eu quero** gerir as contas de utilizadores com acesso à minha *venue* **para que** eu possa gerir cada utilizador (administrador e colaboradores de *venue*) que pode aceder e gerir a informação da minha *venue* na plataforma do Sala-Z.

   US-1: Enviar convite a um novo utilizador para a *venue*

   (a) **Como um** administrador da *venue* autenticado, **eu quero** enviar um convite a um novo utilizador **para que** este possa gerir a informação relativa à minha *venue*. Quero inserir:

   - Voltar a adicionar utilizador removido;
   - Nome*;
   - Apelido*;
   - Email*;

   (* campos obrigatórios)

   US-2: Reenviar convite a um novo utilizador para a *venue*

   (a) **Como um** administrador da *venue* autenticado, **eu quero** reenviar um convite a um novo utilizador **para que** o convite seja novamente enviado. Quero inserir:

   - Voltar a adicionar utilizador removido;
   - Nome*;
   - Apelido*;
   - Email*;

   (* campos obrigatórios)

US-3: Listar utilizadores com acesso à *venue*

(a) **Como um** administrador da *venue* autenticado, **eu quero** listar utilizadores com acesso à minha *venue* **para que** possa ver os utilizadores com permissão de gestão da minha sala.

US-4: Remover utilizador da *venue*

(a) **Como um** administrador da *venue* autenticado, **eu quero** remover utilizador com acesso à minha *venue* **para que** possa remover a permissão de gestão da minha *venue* a um utilizador.

2. Gestão de inventário

ES-2: **Como um** utilizador da *venue* autenticado, **eu quero** criar todos os items do inventário disponíveis na minha *venue* **para que** facilmente saber os items disponíveis na minha *venue*.

US-5: Criar item no inventário

(a) **Como um** utilizador da *venue* autenticado, **eu quero** criar um item do inventário **para que** possa adicionar um item ao inventário da minha *venue*. Quero inserir:

- Nome do material*;
- Descrição;
- Ano de compra*;
- Evento associado;
- Estado*;
- Tags*;
- Fotografia;
- Anexos;

(* campos obrigatórios)

US-6: Criar item no inventário com um número de série

(a) **Como um** utilizador da *venue* autenticado, **eu quero** criar um item do inventário com um número de série **para que** possa adicionar um item ao inventário da minha *venue* através de um item já existente. Quero inserir:

- Número de série;
- Nome do material*;
- Descrição;
- Ano de compra*;
- Evento associado;
- Estado*;

- Tags*;
- Fotografia;
- Anexos;

(* campos obrigatórios)

ES-3: **Como um** utilizador da *venue* autenticado, **eu quero** ter acesso à lista items disponíveis no inventário da minha *venue* **para que** consiga facilmente aceder e gerir todos os meus items de inventário.

US-7: Listar itens da *venue*

(a) **Como um** utilizador da *venue* autenticado, **eu quero** ver a lista completa de items disponíveis na minha *venue* **para que** facilmente ver os items disponíveis. Quero ver:

- Nome do material;
- Tags;
- Número de anexos;
- Ano de compra;
- Evento associado;
- Estado;

US-8: Ordenar itens na lista de itens

(a) **Como um** utilizador da *venue* autenticado, **eu quero** ordenar a lista de items disponíveis na minha *venue* **para que** possa facilmente aceder aos items disponíveis. Quero ordenar:

- Nome do material;
- Evento associado;
- Estado;

US-9: Pesquisar item na lista de itens

(a) **Como um** utilizador da *venue* autenticado, **eu quero** procurar na minha lista de items disponíveis na minha *venue* **para que** possa facilmente aceder aos items disponíveis. Quero procurar:

- Nome do material;

US-10: Filtrar itens na lista de itens

(a) **Como um** utilizador da *venue* autenticado, **eu quero** filtrar a lista de items disponíveis na minha *venue* **para que** possa facilmente aceder aos items disponíveis. Quero filtrar:

- Nome da *venue*

- Evento associado;
- Estado;

US-11: Ver detalhes de um item

(a) **Como um** utilizador da *venue* autenticado, **eu quero** ver os detalhes do item disponível no inventário da minha *venue* **para que** veja facilmente a informação do item detalhado. Quero ver:

- Nome do material;
- Descrição;
- Ano de compra;
- Evento associado;
- Estado;
- Tags;
- Fotografia;
- Anexos;

ES-4: **Como um** utilizador da *venue* autenticado, **eu quero** gerir todos os items disponíveis no inventário da minha *venue* **para que** consiga facilmente entender o estado do meu inventário.

US-12: Editar item

(a) **Como um** utilizador da *venue* autenticado, **eu quero** editar um item disponível no inventário da minha *venue* **para que** possa alterar e atualizar os detalhes associados ao item. Quero editar:

- Nome do material*;
- Descrição;
- Ano de compra*;
- Evento associado;
- Estado*;
- Tags*;
- Fotografia;
- Anexos;

(* campos obrigatórios)

US-13: Duplicar item

(a) **Como um** utilizador da *venue* autenticado, **eu quero** duplicar um item disponível no inventário da minha *venue* **para que** possa criar facilmente items no inventário da minha *venue* através de um item já existente.

US-14: Eliminar item

(a) **Como um** utilizador da *venue* autenticado, **eu quero** eliminar um item disponível no inventário da minha *venue* **para que** possa remover items que já não existem na minha *venue*.

ES-5: **Como um** utilizador da *venue* autenticado, **eu quero** associar um item do inventário da minha *venue* aos meus eventos **para que** possa facilmente gerir os meus eventos e todos os necessários.

US-15: Associar um item a um evento

(a) **Como um** utilizador da *venue* autenticado, **eu quero** associar um item do inventário da minha *venue* aos meus eventos **para que** possa facilmente gerir os meus eventos e todos os items necessários. Quero inserir:
   - Nome do evento;

3. Gestão de artistas

ES-6: **Como um** utilizador da *venue* autenticado, **eu quero** criar artistas **para que** possa facilmente saber os artistas que convidei ou que estão na minha lista para futuros convites para atuarem na minha *venue*.

US-16: Criar artista com importação

(a) **Como um** utilizador da *venue* autenticado, **eu quero** criar um artista importando-o **para que** possa adicionar um artista à minha lista de artistas através de um artista já existente. Quero inserir:
   - Nome do artista para importação;
   - Nome*;
   - Descrição*
   - Região*;
   - Tipo de música*;
   - Link do Spotify;
   - Link do Bandcamp;
   - Link do Soundcloud;
   - Membros da banda;
   - Observações*;
   - Fotografia;
   - Rider*;
   - Anexos;
   - Telemóvel*;
   - Email*;
   - Website;

- Link do Facebook;
- Link do Twitter;
- Link do Instagram;
- Outro Link;
- Álbuns;
- Músicas;

(* campos obrigatórios)

US-17: Criar artista manualmente

(a) **Como um** utilizador da *venue* autenticado, **eu quero** criar um artista manualmente **para que** possa adicionar um artista à minha lista de artistas. Quero inserir:

- Nome*;
- Descrição*
- Região*;
- Tipo de música*;
- Link do Spotify;
- Link do Bandcamp;
- Link do Soundcloud;
- Membros da banda;
- Observações*;
- Fotografia;
- Rider*;
- Anexos;
- Telemóvel*;
- Email*;
- Website;
- Link do Facebook;
- Link do Twitter;
- Link do Instagram;
- Outro Link;
- Álbuns;
- Músicas;

(* campos obrigatórios)

ES-7: **Como um** utilizador da *venue* autenticado, **eu quero** ter acesso à lista de artistas criados previamente **para que** possa facilmente aceder e gerir todos os artistas.

US-18: Listar artistas

(a) **Como um** utilizador da *venue* autenticado, **eu quero** ver a lista completa de artistas **para que** possa ver facilmente os artistas. Quero ver:

- Fotografia;
- Nome;
- Região;
- Tipo de música;
- Número total de atuações;
- Colaborador;
- Número de álbuns e músicas;
- Última atuação;
- Adicionado aos favoritos;

US-19: Listar artistas favoritos

(a) **Como um** utilizador da *venue* autenticado, **eu quero** ver a lista completa de artistas favoritos **para que** possa ver facilmente os artistas. Quero ver:

- Nome;
- Região;
- Tipo de música;
- Número total de atuações;
- Colaborador;
- Número de álbuns e músicas;
- Última atuação;
- Adicionado aos favoritos;

US-20: Procurar artista na lista de artistas

(a) **Como um** utilizador da *venue* autenticado, **eu quero** procurar a lista completa de artistas **para que** possa facilmente aceder aos artistas. Quero procurar:

- Nome;

US-21: Filtrar lista de artistas

(a) **Como um** utilizador da *venue* autenticado, **eu quero** filtrar a lista completa de artistas **para que** possa facilmente aceder aos artistas. Quero filtrar:

- Nome;
- Região;
- Tipo de música;
- Última atuação;

US-22: Ver detalhes de um artista

(a) **Como um** utilizador da *venue* autenticado, **eu quero** ver os detalhes de um artista **para que** veja facilmente a informação do artista detalhada. Quero ver:

- Nome;
- Descrição;
- Região;
- Tipo de música;
- Link do Spotify;
- Link do Bandcamp;
- Link do Soundcloud;
- Membros da banda;
- Colaboradores;
- Observações;
- Fotografia;
- Rider;
- Anexos;
- Telemóvel;
- Email;
- Website;
- Link do Facebook;
- Link do Twitter;
- Link do Instagram;
- Outro Link;
- Álbuns;
- Músicas;

ES-8: **Como um** utilizador da *venue* autenticado, **eu quero** gerir todos os meus artistas **para que** possa facilmente aceder aos artistas que aturam ou possam atuar no futuro na minha *venue*.

US-23: Editar artista

(a) **Como um** utilizador da *venue* autenticado, **eu quero** editar um artista **para que** possa alterar e atualizar os detalhes associados ao artista. Quero editar:

- Nome*;
- Descrição*
- Região*;
- Tipo de música*;
- Link do Spotify;
- Link do Bandcamp;

- Link do Soundcloud;
- Membros da banda;
- Colaboradores;
- Observações*;
- Fotografia;
- Rider*;
- Anexos;
- Telemóvel*;
- Email*;
- Website;
- Link do Facebook;
- Link do Twitter;
- Link do Instagram;
- Outro Link;
- Álbuns;
- Músicas;

(* campos obrigatórios)

US-24: Eliminar artista

(a) **Como um** utilizador da *venue* autenticado, **eu quero** eliminar um artista **para que** possa retirar um artista que já não façam sentido existir na minha lista de artistas.

US-25: Marcar um artista como favorito

(a) **Como um** utilizador da *venue* autenticado, **eu quero** marcar um artista como favorito **para que** possa ser adicionado à lista de artistas favoritos.

ES-9: **Como um** utilizador da *venue* autenticado, **eu quero** aceder facilmente às métricas principais do meu artista **para que** possa facilmente saber o artista que teve mais atuações na minha *venue* e as métricas principais associadas aos eventos deles.

US-26: Ver as estatísticas associadas a artistas

(a) **Como um** utilizador da *venue* autenticado, **eu quero** seguir as métricas principais do meu artista pela *dashboard* para que possa facilmente saber as métricas associadas ao artista. Quero ver:

- Selecionar o artista;
- Audiência;
- Novos espaços de atuação;

- Presença em eventos;

4. *Dashboard*

   ES-10: **Como um** utilizador da *venue* autenticado, **eu quero** aceder à informação relevante da minha *venue* **para que** possa facilmente saber os eventos e tarefas associadas à minha *venue* e os resultados das suas métricas.

   US-27: Ver as estatísticas associadas à *venue*

   (a) **Como um** utilizador da *venue* autenticado, **eu quero** seguir as métricas e estado atualizados das tarefas associadas aos meus eventos **para que** possa facilmente ter uma visão global do estado das tarefas associado a determinado evento. Quero ver:

   - Calendário;
   - Próximos eventos;
   - Atividade recente;
   - Próximas tarefas;
   - Gráfico de presença em eventos;
   - Gráfico de receitas e despesas;
   - Resumo de atividade das tarefas por evento;
   - Resumo de atividade das notificações;
   - Lista de artistas;

5. Gestão de tarefas de um evento

   ES-11: **Como um** utilizador da *venue* autenticado, **eu quero** criar tarefas associadas a cada evento **para que** possa facilmente acompanhar o estado de cada um dos meus eventos.

   US-28: Rápida criação da tarefa

   (a) **Como um** utilizador da *venue* autenticado, **eu quero** criar rapidamente uma tarefa num evento **para que** possa rapidamente adicionar uma tarefa à lista de tarefas. Quero inserir:

   - Título da tarefa;
   - Descrição da tarefa;
   - Estado da tarefa;
   - Anexos;
   - Prazo de conclusão da tarefa;
   - Membro responsável pela tarefa;
   - Tags;

US-29: Criar tarefa com detalhes

(a) **Como um** utilizador da *venue* autenticado, **eu quero** criar detalhadamente uma tarefa num evento **para que** possa adicionar uma tarefa com mais detalhes à lista de tarefas. Quero inserir:

- Título da tarefa;
- Descrição da tarefa;
- Estado da tarefa;
- Anexos;
- Prazo de conclusão da tarefa;
- Hora de entrega da tarefa;
- Membro responsável pela tarefa;
- Tags;

ES-12: **Como um** utilizador da *venue* autenticado, **eu quero** ter acesso à lista de tarefas para cada um dos meus eventos **para que** possa ter acesso e gerir cada um dos meus eventos.

US-30: Listar tarefas

(a) **Como um** utilizador da *venue* autenticado, eu quero ver a lista completa de tarefas dos meus eventos para que possa ver as tarefas dos eventos. Quero ver:

- Título da tarefa;
- Estado da tarefa;
- Número de anexos;
- Prazo de conclusão da tarefa;
- Membro responsável pela tarefa;
- Tags;
- Número de comentários associados;
- Evento associado;

US-31: Ver detalhes de uma tarefa

(a) **Como um** utilizador da *venue* autenticado, **eu quero** ver os detalhes de uma tarefa **para que** veja facilmente a informação da tarefa detalhada. Quero ver:

- Título da tarefa;
- Descrição da tarefa;
- Estado da tarefa;
- Anexos;
- Prazo de conclusão da tarefa;
- Membro responsável pela tarefa;
- Tags;

- Atividade da tarefa;
- Evento associado;

US-32: Ordenar lista de tarefas

(a) **Como um** utilizador da *venue* autenticado, **eu quero** ordenar as tarefas na lista de tarefas **para que** possa facilmente aceder a tarefas. Quero ordenar:
   - Título da tarefa;
   - Prazo de conclusão da tarefa;
   - Membro responsável pela tarefa;
   - Evento associado;

US-33: Procura tarefa na lista de tarefas

(a) **Como um** utilizador da *venue* autenticado, **eu quero** procurar na lista de tarefas dos meus eventos **para que** possa facilmente aceder a tarefas. Quero procurar:
   - Título da tarefa;

US-34: Filtrar lista de tarefas

(a) **Como um** utilizador da *venue* autenticado, **eu quero** filtrar a lista completa de tarefas **para que** possa facilmente aceder a tarefas. Quero filtrar:
   - Prazo de conclusão da tarefa;
   - Membro responsável pela tarefa;
   - Tags;

ES-13: **Como um** utilizador da *venue* autenticado, **eu quero** gerir todas as tarefas associadas a cada evento **para que** facilmente perceber o estado de cada um dos eventos.

US-35: Editar tarefa

(a) **Como um** utilizador da *venue* autenticado, **eu quero** editar uma tarefa disponível na lista de tarefas **para que** possa alterar e atualizar os detalhes associados à tarefa. Quero editar:
   - Título da tarefa;
   - Descrição da tarefa;
   - Estado da tarefa;
   - Anexos;
   - Prazo de conclusão da tarefa;
   - Hora de entrega da tarefa;

- Membro responsável pela tarefa;
- Tags;

US-36: Duplicar tarefa

(a) **Como um** utilizador da *venue* autenticado, **eu quero** duplicar uma tarefa disponível na lista de tarefas **para que** possa criar facilmente tarefas na lista de tarefas da minha *venue* através de uma já existente.

US-37: Eliminar tarefa

(a) **Como um** utilizador da *venue* autenticado, **eu quero** eliminar uma tarefa disponível na lista de tarefas **para que** possa remover tarefas que já não existem na lista de tarefas do meu evento.

US-38: Ver a atividade associada à tarefa

(a) **Como um** utilizador da *venue* autenticado, **eu quero** ver a atividade e comentários das tarefas **para que** possa saber as alterações nas tarefas.

US-39: Adicionar comentário à tarefa

(a) **Como um** utilizador da *venue* autenticado, **eu quero** adicionar um comentário numa tarefa **para que** possa comunicar com outros utilizadores.

US-40: Editar comentário

(a) **Como um** utilizador da *venue* autenticado, **eu quero** editar um comentário numa tarefa **para que** possa alterar o conteúdo do meu comentário que escrevi previamente na tarefa.

US-41: Eliminar comentário

(a) **Como um** utilizador da *venue* autenticado, **eu quero** eliminar um comentário numa tarefa **para que** possa remover comentários que já não façam sentido existir.

6. Gestão de tarefas de uma *venue*/utilizador

ES-14: **Como um** utilizador da *venue* autenticado, **eu quero** ter acesso à lista de todas as tarefas associadas aos eventos da minha *venue* **para que** possa facilmente aceder e gerir cada um dos meus eventos.

US-42: Ver as estatísticas associadas a tarefas

(a) **Como um** utilizador da *venue* autenticado, **eu quero** visualizar as tarefas associadas à minha *venue* em uma *dashboard* **para que** possa visualizar métricas e gerir melhor as tarefas. Quero ver:

- Visão geral do estado das tarefas;
- Tarefas expiradas;
- Lista de tarefas:
  - Título da tarefa;
  - Estado da tarefa;
  - Prazo de conclusão da tarefa;
  - Membro responsável pela tarefa;
  - Título do evento;
- Calendário;
- Gráfico com as tarefas por mês e por estado;
- Tarefas concluída fora do prazo;

7. Comunicação entre *stakeholders*

   ES-15: **Como um** utilizador da *venue* autenticado, **eu quero** comunicar com outros *stakeholders* do evento **para que** possa facilmente acompanhar o estado do evento enviando mensagens a outros utilizadores da *venue* ou diretamente a artistas pela plataforma.

   US-43: Mencionar outros utilizadores da *venue* com comentários

   (a) **Como um** utilizador da *venue* autenticado, **eu quero** aceder à atividade do evento e adicionar um comentário **para que** possa comunicar com outros *stakeholders* da *venue* ou artista através desta.

8. Notificações

   ES-16: **Como um** utilizador da *venue* autenticado, **eu quero** receber notificações relevantes **para que** possa facilmente acompanhar qualquer mudança na minha *venue*, artistas, eventos ou tarefas associadas aos eventos.

   US-44: Ver notificações

   (a) **Como um** utilizador da *venue* autenticado, **eu quero** receber notificações **para que** possa facilmente visualizar comentários em que fui mencionado por outros *stakeholders*.

9. Configuração relatórios de eventos

   ES-17: **Como um** utilizador da *venue* autenticado, **eu quero** configurar relatórios de eventos relevantes **para que** para estar atualizado de todas as informações relacionadas com o evento.

US-45: Configurar relatórios de eventos

(a) **Como um** utilizador da *venue* autenticado, **eu quero** configurar as notificações relativas a eventos **para que** possa estar atualizado de todas as informações relacionadas com o evento. Quero configurar:

- Notificações diárias - Para eventos que ocorrem na próxima semana, receber email diário com as tarefas concluídas e por concluir. Apenas se houver tarefas por concluir;
- Notificações semanais - Após uma semana da criação dos eventos, receber email semanal com o estado das tarefas;
- Receber notificações no painel de controlo para conseguir ver as notificações mais recentes assim que entrar na plataforma;
- Barra de pesquisa - Receber notificações no icon de notificações da barra de pesquisa;

10. Gerir templates de eventos

ES-18: **Como um** utilizador da *venue* autenticado, **eu quero** criar e gerir templates de eventos **para que** possa facilmente criar novos eventos baseados nas minhas práticas usuais.

US-46: Criar template

(a) **Como um** utilizador da *venue* autenticado, **eu quero** criar um template de evento **para que** possa criar mais facilmente eventos posteriormente. Quero inserir:

- Nome do evento;
- Espaço;
- Descrição;
- Membros da banda;
- Tags;
- Estado do evento;
- Tipo de evento;
- Artista;
- Data de início;
- Data de fim;
- Hora;
- Sala;
- Audiência do evento;
- Lotação;
- Participantes;
- Anexos;
- Bilhetes;
- Gastos do evento;

- Receitas do evento;

US-47: Listar templates

(a) **Como um** utilizador da *venue* autenticado, eu quero ver a lista completa de templates de eventos para que possa ver todos os templates que criei. Quero ver:
   - Nome do evento;
   - Espaço;
   - Tipo de evento;
   - Número de tarefas;

US-48: Editar template

(a) **Como um** utilizador da *venue* autenticado, **eu quero** editar um template de evento **para que** possa alterar e atualizar os detalhes associados ao template. Quero editar:
   - Nome do evento;
   - Espaço;
   - Descrição;
   - Colaboradores;
   - Tags;
   - Estado do evento;
   - Tipo de evento;
   - Artista;
   - Data de início;
   - Data de fim;
   - Hora;
   - Sala;
   - Audiência do evento;
   - Lotação;
   - Participantes;
   - Anexos;
   - Bilhetes;
   - Gastos do evento;
   - Receitas do evento;
   - Tarefas associadas;

US-49: Eliminar template

(a) **Como um** utilizador da *venue* autenticado, **eu quero** eliminar um template de evento disponível na lista de templates de evento **para que** possa remover templates que já não existem na lista de templates de eventos.

ES-19: **Como um** utilizador da *venue* autenticado, **eu quero** criar facilmente eventos a partir de um template **para que** possa facilmente criar um novo evento baseado nas minhas práticas habituais.

US-50: Criar um evento a partir de um template

(a) **Como um** utilizador da *venue* autenticado, **eu quero** criar um evento a partir de um template **para que** facilmente criar um evento. Quero inserir:

- Selecionar o template;
- Nome do evento*;
- Artista*;
- Data*;
- Horas*;
- Espaço*;
- Tipo de evento*;
- Colaboradores;

(* campos obrigatórios)

ES-20: **Como um** utilizador da *venue* autenticado, **eu quero** criar facilmente templates a partir de eventos criados **para que** possa facilmente replicar um evento na plataforma.

US-51: Criar um template a partir de um evento

(a) **Como um** utilizador da *venue* autenticado, **eu quero** criar um template a partir de um evento **para que** possa facilmente criar eventos na plataforma. Quero inserir:

- Nome do template*;

(* campos obrigatórios)

11. Gerir eventos

ES-21: **Como um** utilizador da *venue* autenticado, **eu quero** duplicar um evento **para que** possa facilmente replicar um evento na plataforma.

US-52: Duplicar evento

(a) **Como um** utilizador da *venue* autenticado, **eu quero** duplicar um evento na lista de eventos **para que** possa criar facilmente eventos através de um evento já existente.

Módulo: Gestão de Artista

1. Painel de Administração

   ES-22: **Como um** administrador do artistas autenticado, **eu quero** gerir as contas de utilizadores com acesso ao meu artista **para que** eu possa gerir cada utilizador (administrador e colaboradores de artistas) que pode aceder e gerir a informação do meu artista na plataforma do Sala-Z.

   US-53: Enviar convite a um novo utilizador para o meu artista

   (a) **Como um** administrador do artistas autenticado, **eu quero** enviar um convite a um novo utilizador **para que** este possa gerir a informação relativa ao meu artista. Quero inserir:

   - Voltar a adicionar utilizador removido;
   - Nome*;
   - Apelido*;
   - Email*;

   (* campos obrigatórios)

   US-54: Reenviar convite a um novo utilizador para o meu artista

   (a) **Como um** administrador do artistas autenticado, **eu quero** reenviar um convite a um novo utilizador **para que** o convite seja novamente enviado. Quero inserir:

   - Voltar a adicionar utilizador removido;
   - Nome*;
   - Apelido*;
   - Email*;

   (* campos obrigatórios)

   US-55: Listar utilizadores com acesso ao meu artista

   (a) **Como um** administrador do artistas autenticado, **eu quero** listar utilizadores com acesso ao meu artista **para que** possa ver os utilizadores com permissão de gestão do meu artista.

   US-56: Remover utilizador do artista

   (a) **Como um** administrador do artistas autenticado, **eu quero** remover utilizador com acesso ao meu artista **para que** possa remover a permissão de gestão do meu artista a um utilizador.

2. Gestão de artistas

   ES-23: *Como um* utilizador de artistas autenticado, **eu quero** ver e editar os meus artistas na plataforma Sala-Z **para que** a plataforma possa exibir a informação atualizada sobre o meu artista.

   US-57: Listar artistas

   (a) **Como um** utilizador de artistas autenticado, **eu quero** ver a lista completa de artistas **para que** possa ver facilmente os artistas. Quero ver:
   - Fotografia;
   - Nome;
   - Região;
   - Tipo de música;
   - Número total de atuações;
   - Colaborador;
   - Número de álbuns e músicas;
   - Última atuação;
   - Adicionado aos favoritos;

   US-58: Filtrar lista de artistas

   (a) **Como um** utilizador de artistas autenticado, **eu quero** filtrar a lista completa dos meus artistas **para que** possa facilmente aceder aos artistas. Quero filtrar:
   - Nome;
   - Região;
   - Tipo de música;
   - Última atuação;

   US-59: Editar artista

   (a) **Como um** utilizador de artistas autenticado, **eu quero** editar um artista **para que** possa alterar e atualizar os detalhes associados ao artista. Quero editar:
   - Nome*;
   - Descrição*
   - Região*;
   - Tipo de música*;
   - Link do Spotify;
   - Link do Bandcamp;
   - Link do Soundcloud;
   - Membros da banda;
   - Colaboradores;

- Observações;
- Fotografia;
- Rider;
- Anexos;
- Telemóvel;
- Email;
- Website;
- Link do Facebook;
- Link do Twitter;
- Link do Instagram;
- Outro Link;
- Álbuns;
- Músicas;

(* campos obrigatórios)

US-60: Ver detalhes de um artista

(a) **Como um** utilizador de artistas autenticado, **eu quero** ver os detalhes de um artista **para que** veja facilmente a informação do artista detalhada. Quero ver:

- Nome;
- Descrição;
- Região;
- Tipo de música;
- Link do Spotify;
- Link do Bandcamp;
- Link do Soundcloud;
- Membros da banda;
- Colaboradores;
- Observações;
- Fotografia;
- Rider;
- Anexos;
- Telemóvel;
- Email;
- Website;
- Link do Facebook;
- Link do Twitter;
- Link do Instagram;
- Outro Link;
- Álbuns;
- Músicas;

US-61: Eliminar artista

(a) **Como um** utilizador de artistas autenticado, **eu quero** eliminar um artista **para que** possa retirar um artista que já não faça sentido existir na minha lista de artistas.

ES-24: **Como um** utilizador de artistas autenticado, **eu quero** criar artistas **para que** os meus artistas sejam apresentados na plataforma Sala-Z.

US-62: Criar artista com importação

(a) **Como um** utilizador de artistas autenticado, **eu quero** criar um artista importando-o **para que** possa adicionar um artista à minha lista de artistas através de um artista já existente. Quero inserir:

- Nome do artista para importação;
- Nome*;
- Descrição*
- Região*;
- Tipo de música*;
- Link do Spotify;
- Link do Bandcamp;
- Link do Soundcloud;
- Membros da banda;
- Observações*;
- Fotografia;
- Rider*;
- Anexos;
- Telemóvel*;
- Nome da pessoa responsável*;
- Email*;
- Website;
- Link do Facebook;
- Link do Twitter;
- Link do Instagram;
- Outro Link;
-

(* campos obrigatórios)

US-63: Criar artista manualmente

(a) **Como um** utilizador de artistas autenticado, **eu quero** criar um artista manualmente **para que** possa adicionar um artista à minha lista de artistas. Quero inserir:

- Nome*;
- Descrição*
- Região*;
- Tipo de música*;
- Link do Spotify;
- Link do Bandcamp;
- Link do Soundcloud;
- Membros da banda;
- Observações*;
- Fotografia;
- Rider*;
- Anexos;
- Telemóvel*;
- Nome da pessoa responsável*;
- Email*;
- Website;
- Link do Facebook;
- Link do Twitter;
- Link do Instagram;
- Outro Link;
-

(* campos obrigatórios)

ES-25: **Como um** utilizador de artistas autenticado, **eu quero** pedir a verificação do meu artista **para que** outro utilizador da plataforma Sala-Z possa ver o meu perfil como referenciado e artista verificado.

US-64: Verificar artista

(a) **Como um** utilizador de artistas autenticado, **eu quero** adicionar um selo de verificação a artista **para que** outro utilizador da plataforma Sala-Z possa ver o meu perfil como artista verificado. Quero inserir:

- Nome*;
- Email*
- Anexo;

(* campos obrigatórios)

3. *Dashboard*

ES-26: **Como um** utilizador de artistas autenticado, **eu quero** aceder à informação relevante do meu artista **para que** possa facilmente acompanhar a minha atividade, eventos e resultados das suas métricas.

US-65: Ver as estatísticas associadas a artistas

(a) **Como um** utilizador de artistas autenticado, **eu quero** seguir as métricas e atividade de eventos associados ao artista **para que** possa facilmente ter uma visão global das minhas métricas. Quero ver:

- Calendário com eventos;
- Audiência;
- Novos espaços de atuação;
- Presença em eventos;
- Presença em eventos;
- Receitas e despesas;
- Próximos eventos;
- Lista de artistas:
  - ID;
  - Artista;
  - Audiência;
  - Receita;
  - Última atuação;
  - Número de atuações

(* campos obrigatórios)

US-66: Filtrar lista de artistas na tabela

(a) **Como um** utilizador de artistas autenticado, **eu quero** filtrar os artistas na tabela **para que** possa facilmente aceder aos dados relativos a um artista especifico. Quero filtrar:

- Nome do artista.

US-67: Ordenar artistas na tabela

(a) **Como um** utilizador de artistas autenticado, **eu quero** ordenar a lista das métricas dos artistas **para que** possa facilmente visualizar a informação. Quero ordenar:

- ID;
- Artista;
- Audiência;
- Receita;
- Última atuação;
- Número de atuações;

4. Gerir eventos

   ES-27: **Como um** utilizador de artistas autenticado, **eu quero** ver e editar os meus eventos na plataforma Sala-Z **para que** a plataforma possa exibir a informação atualizada sobre os meus eventos.

   US-68: Listar eventos do artista

   (a) **Como um** utilizador de artistas autenticado, **eu quero** listar os eventos **para que** possa ver todos os eventos que estou associado como colaborador. Quero ver:

   - Data;
   - Nome;
   - Estado;
   - Artista;
   - Horas;
   - Espaço;

   US-69: Filtrar lista de eventos do artista

   (a) **Como um** utilizador de artistas autenticado, **eu quero** filtrar os eventos **para que** possa facilmente aceder a eventos. Quero filtrar:

   - Nome do artista;
   - Estado;
   - Tipo de evento;
   - Mês;
   - Espaço;

   US-70: Ver detalhes do evento

   (a) **Como um** utilizador de artistas autenticado, **eu quero** ver os detalhes do evento **para que** possa ver facilmente a informação do item detalhado. Quero ver:

   - Nome;
   - Artista;
   - Espaço;
   - Notas sobre a organização do evento;
   - Estado;
   - Tags;
   - Evento público;
   - Dia;
   - Repetição;
   - Estadia;
   - Horário;

- Duração do evento;
- Sala;
- Lotação da sala;
- Audiência;
- Cachet;
- Atividade;

5. Comunicação entre stakeholders

   ES-28: **Como um** utilizador de artistas autenticado, **eu quero** comunicar com outros *stakeholders* do evento **para que** possa facilmente acompanhar o estado do evento.

   US-71: Mencionar outros utilizadores do evento com comentários

   (a) **Como um** utilizador de artistas autenticado, **eu quero** aceder à atividade do evento **para que** possa comunicar com outros *stakeholders*.

6. Notificações

   ES-29: **Como um** utilizador de artistas autenticado, **eu quero** receber notificações relevantes **para que** possa facilmente acompanhar qualquer mudança na atividade do artista e eventos.

   US-72: Ver notificações

   (a) **Como um** utilizador de artistas autenticado, **eu quero** receber notificações **para que** possa facilmente visualizar comentários em que fui mencionado por outros *stakeholders* e toda a atividade associada a eventos e artistas.

7. Configurar relatórios de eventos

   ES-30: **Como um** utilizador de artistas autenticado, **eu quero** configurar relatórios de eventos relevantes **para que** para estar atualizado de todas as informações relacionadas com o evento.

   US-73: Configurar relatórios de eventos

   (a) **Como um** utilizador de artistas autenticado, **eu quero** configurar as notificações relativas a eventos **para que** possa estar atualizado de todas as informações relacionadas com o evento. Quero configurar:

   - Notificações frequentes - Para os meus eventos, quero receber um receber notificações por e-mail sobre informações dos eventos, nomeadamente comentários, atualizações;

- Receber notificações no painel de controlo para conseguir ver as notificações mais recentes assim que entrar na plataforma;
- Barra de pesquisa - Receber notificações no icon de notificações da barra de pesquisa;

Apêndice B

Funcionalidades Desenvolvidas

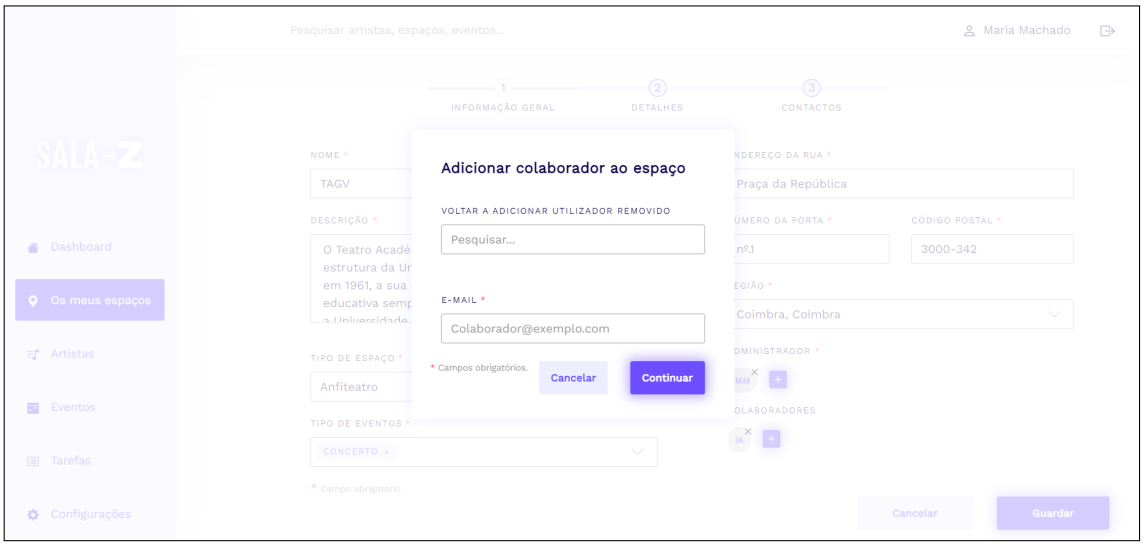

Figura B.1: Adicionar colaborador à *venue* (Plataforma Sala-Z 2.1)

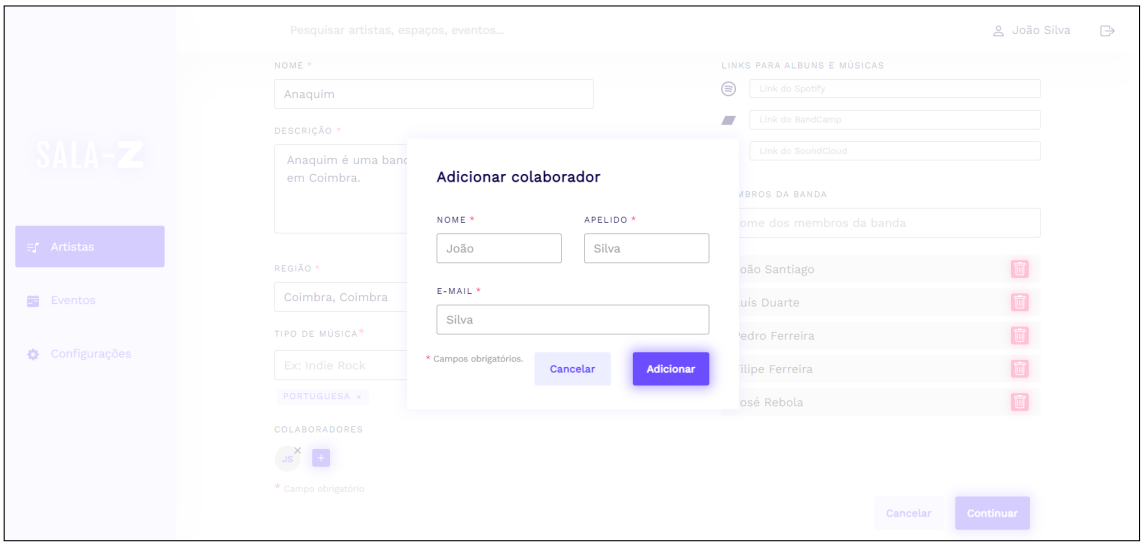

Figura B.2: Adicionar colaborador ao artista (Plataforma Sala-Z 2.1)

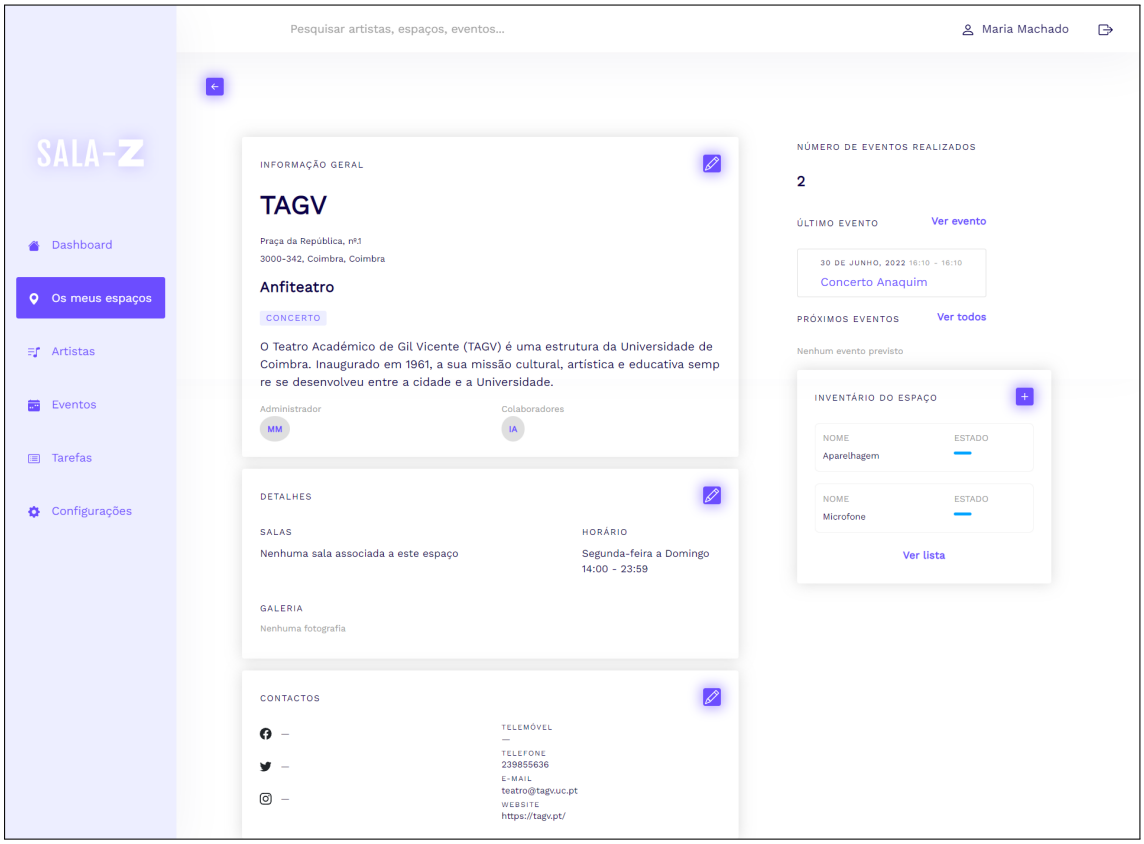

Figura B.3: Listagem de inventário na *venue* (Plataforma Sala-Z 2.1)

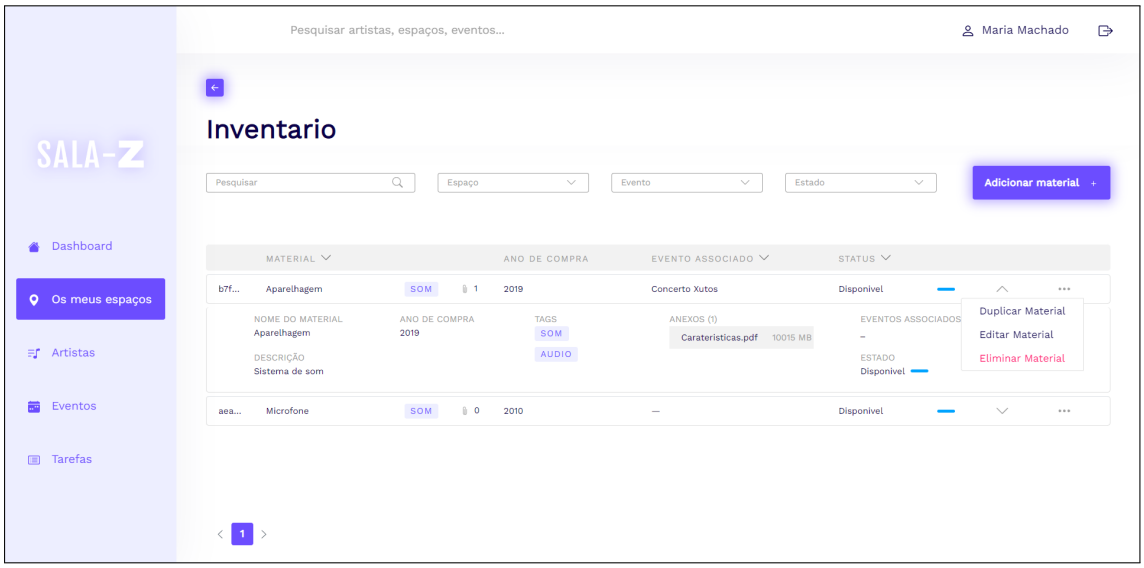

Figura B.4: Listagem de inventário (Plataforma Sala-Z 2.1)

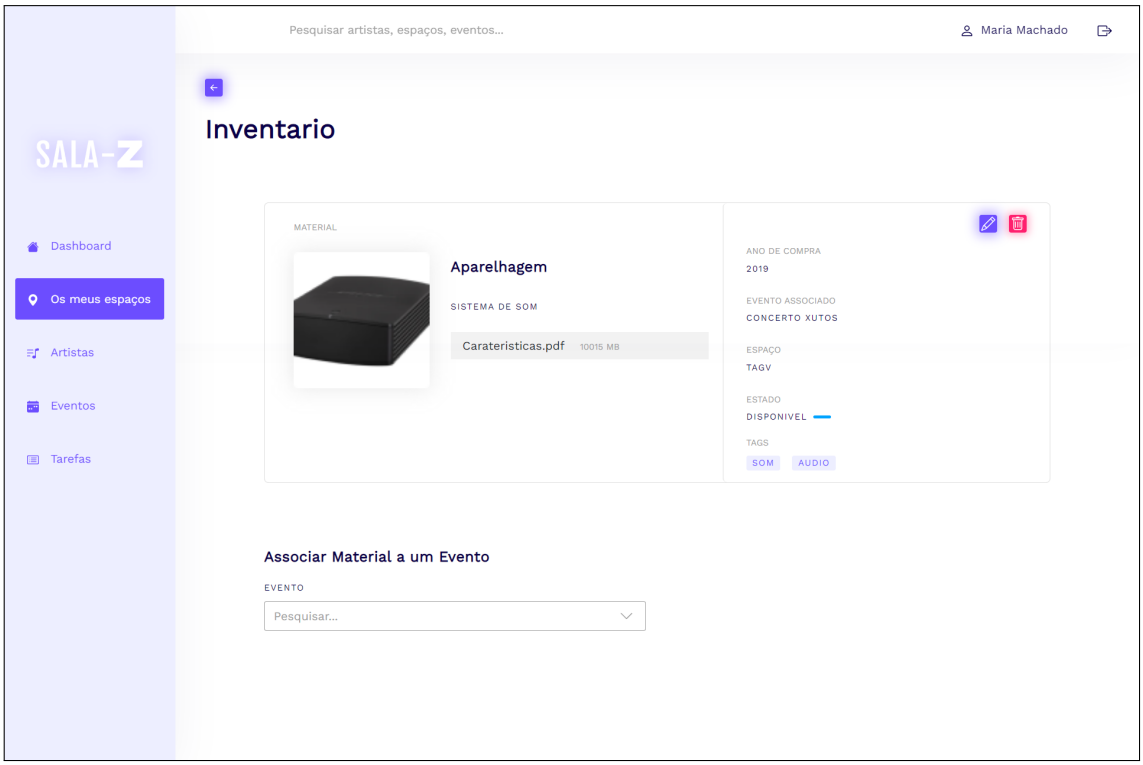

Figura B.5: Detalhes de um item do inventário (Plataforma Sala-Z 2.1)

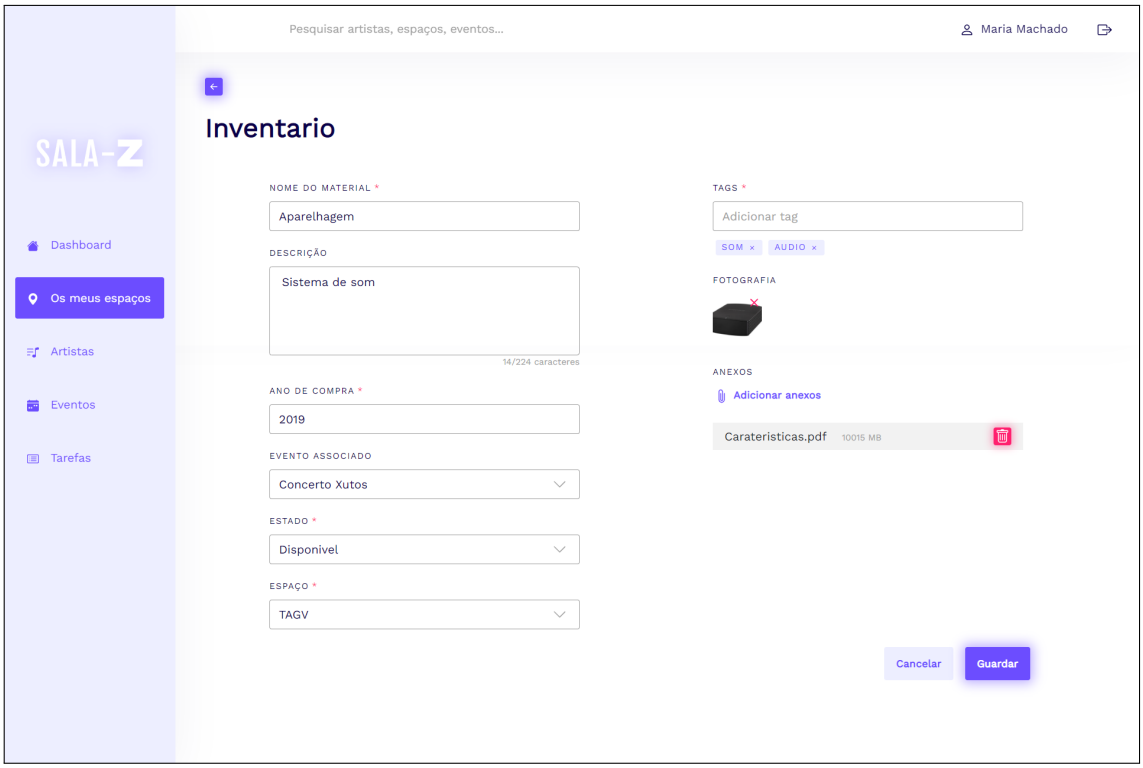

Figura B.6: Editar de um item do inventário (Plataforma Sala-Z 2.1)

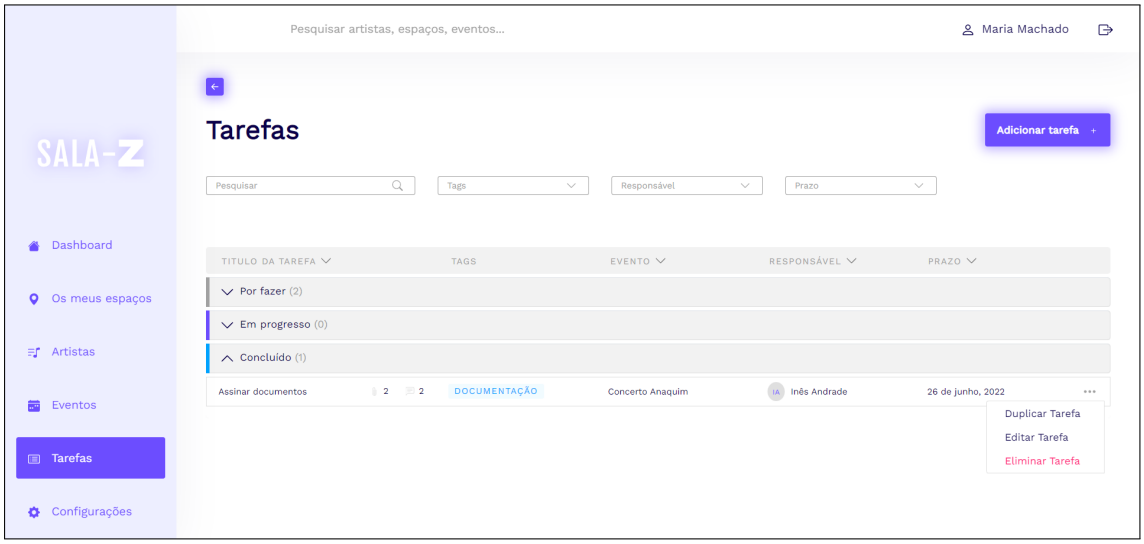

Figura B.7: Listagem de tarefas (Plataforma Sala-Z 2.1)

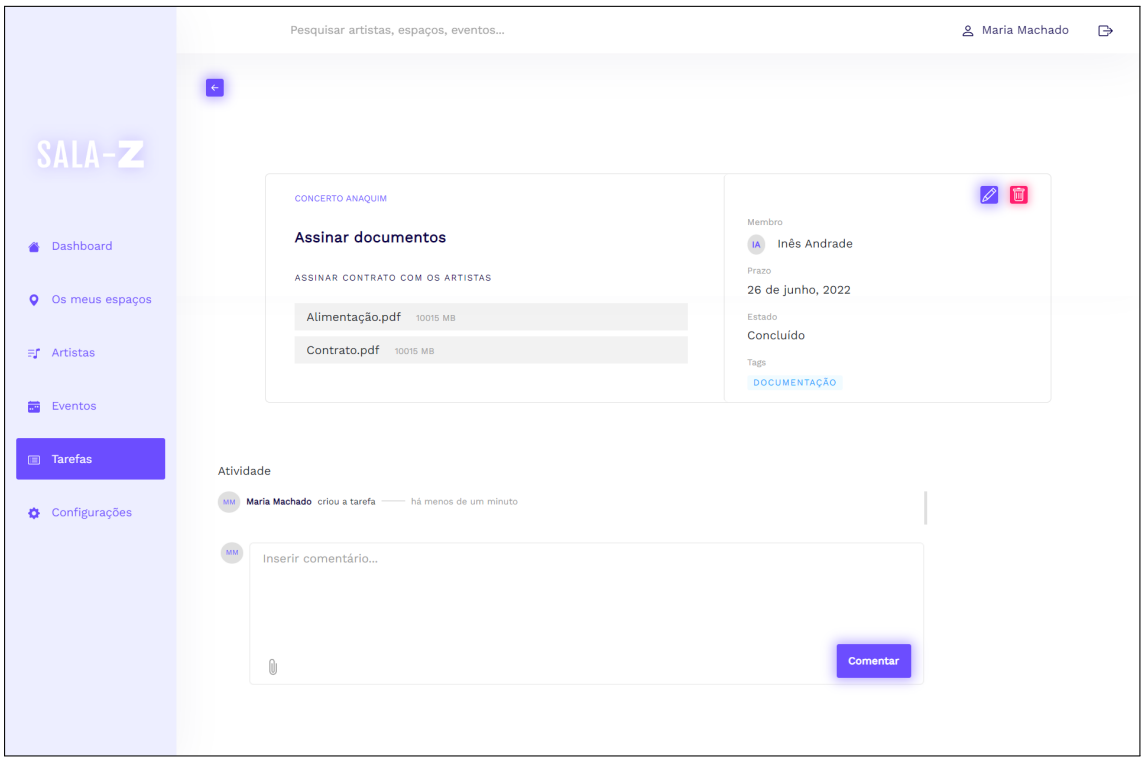

Figura B.8: Detalhes de uma tarefa (Plataforma Sala-Z 2.1)

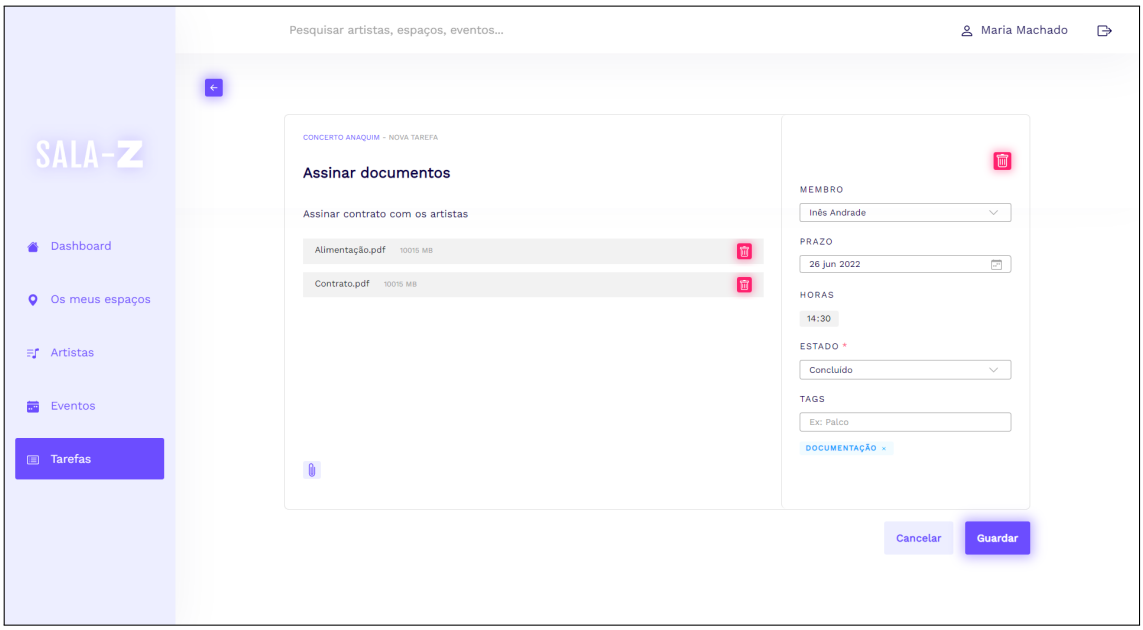

Figura B.9: Editar de uma tarefa (Plataforma Sala-Z 2.1)

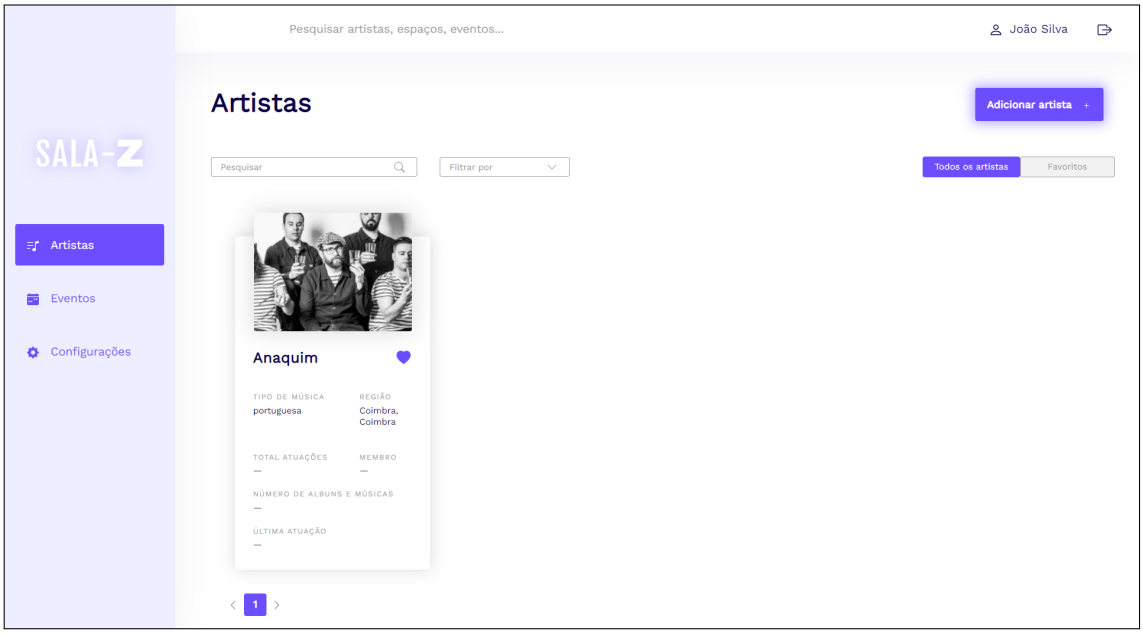

Figura B.10: Alterações de dados pessoais (Plataforma Sala-Z 2.0)

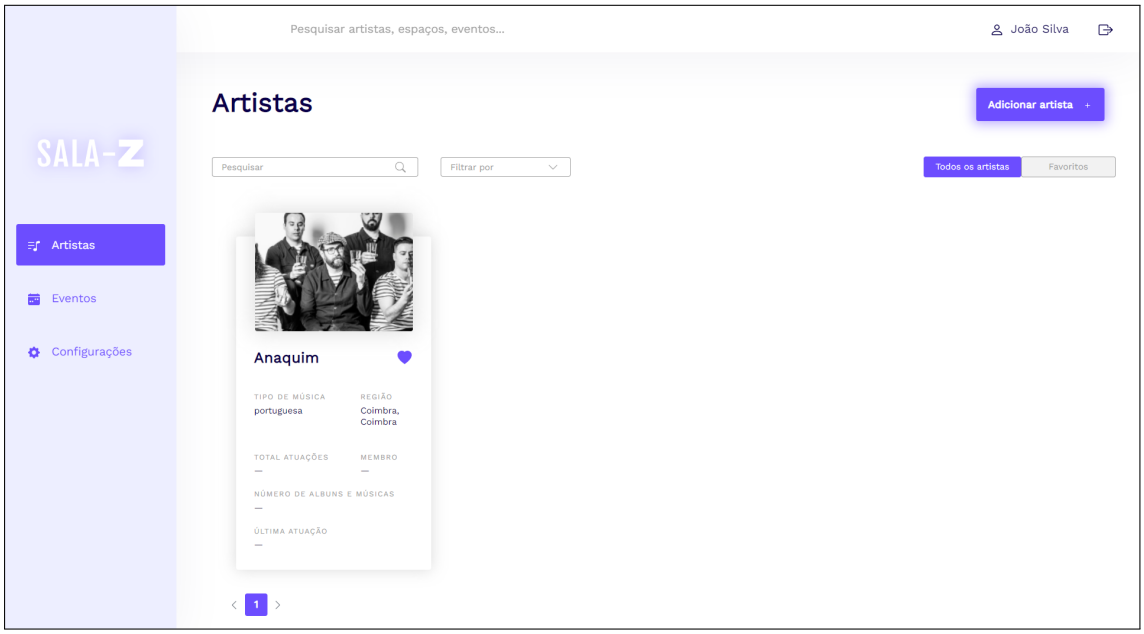

Figura B.11: Listagem de artistas (Plataforma Sala-Z 2.1)

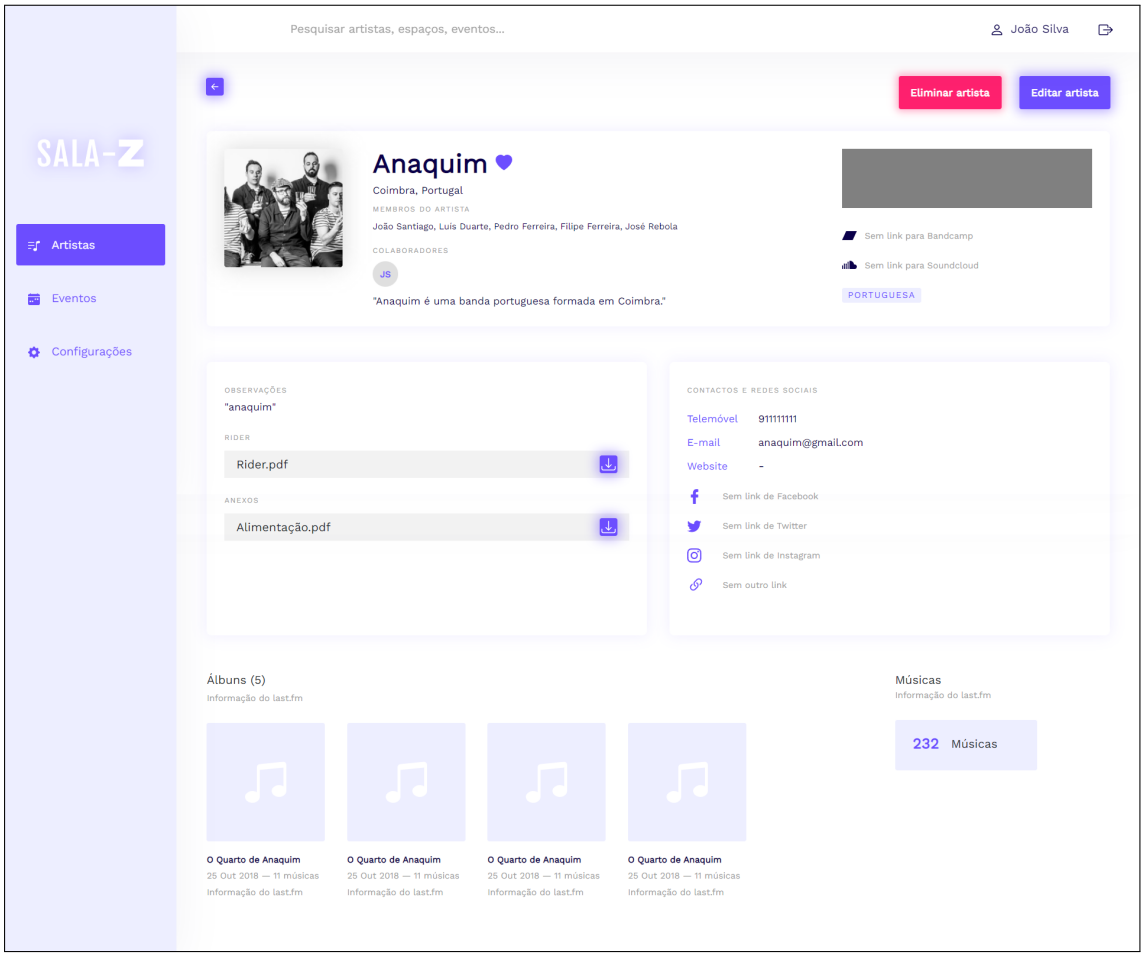

Figura B.12: Detalhes do artista (Plataforma Sala-Z 2.1)

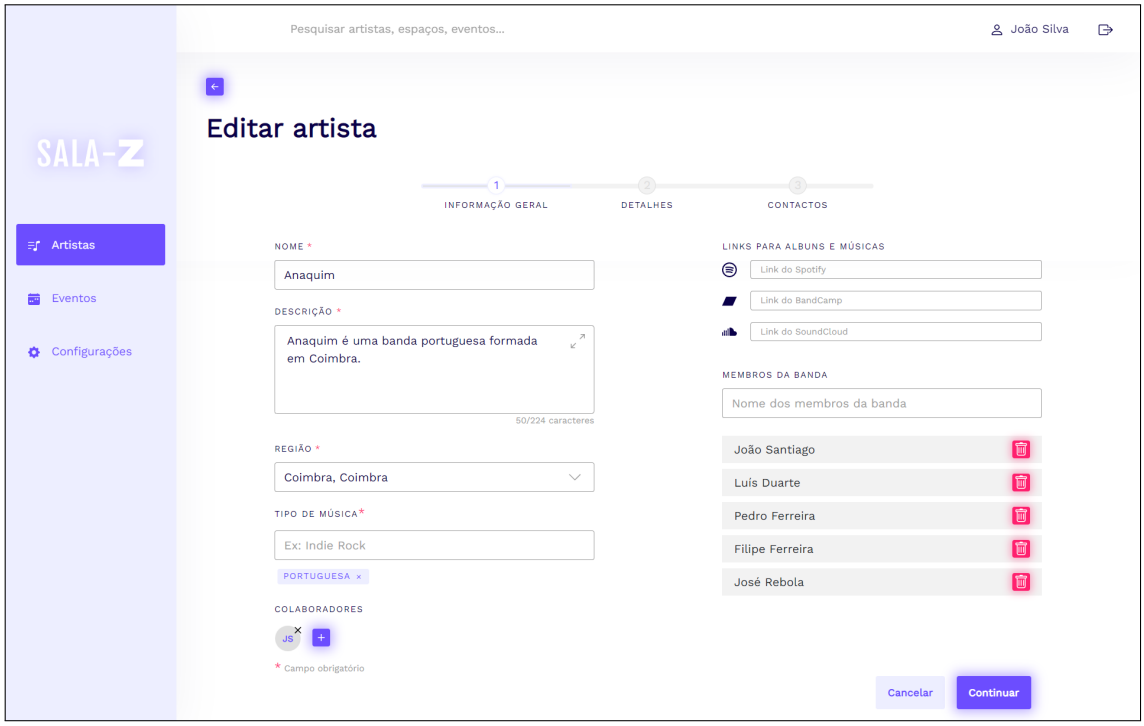

Figura B.13: Editar um artista (Plataforma Sala-Z 2.1)

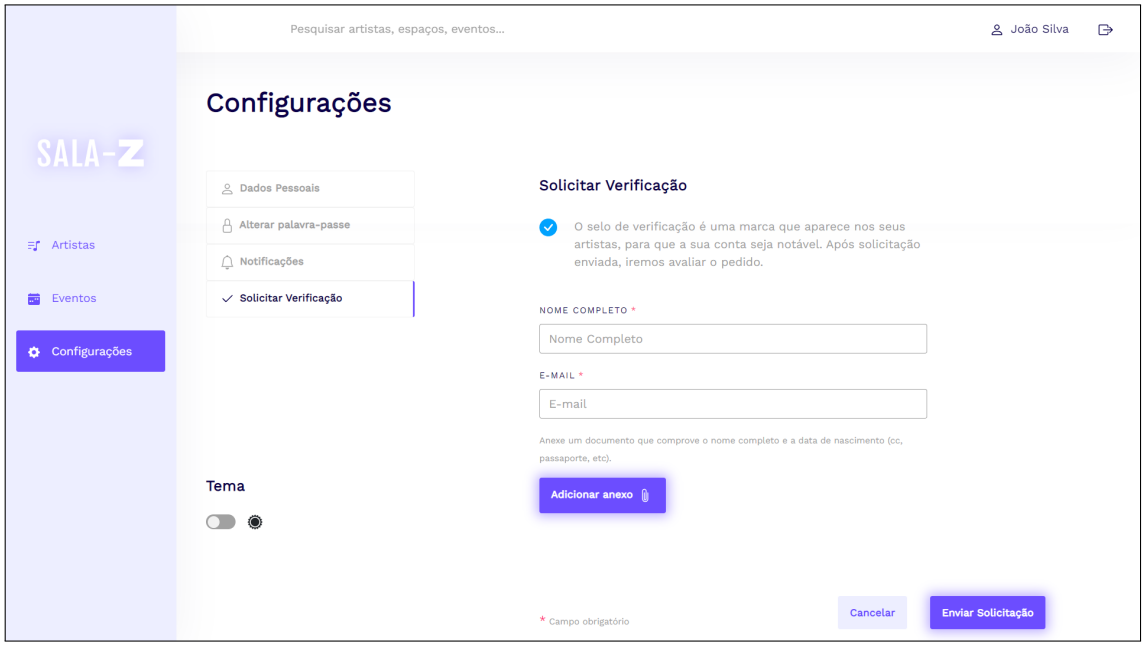

Figura B.14: Verificação de um perfil de gestor de artistas (Plataforma Sala-Z 2.1)

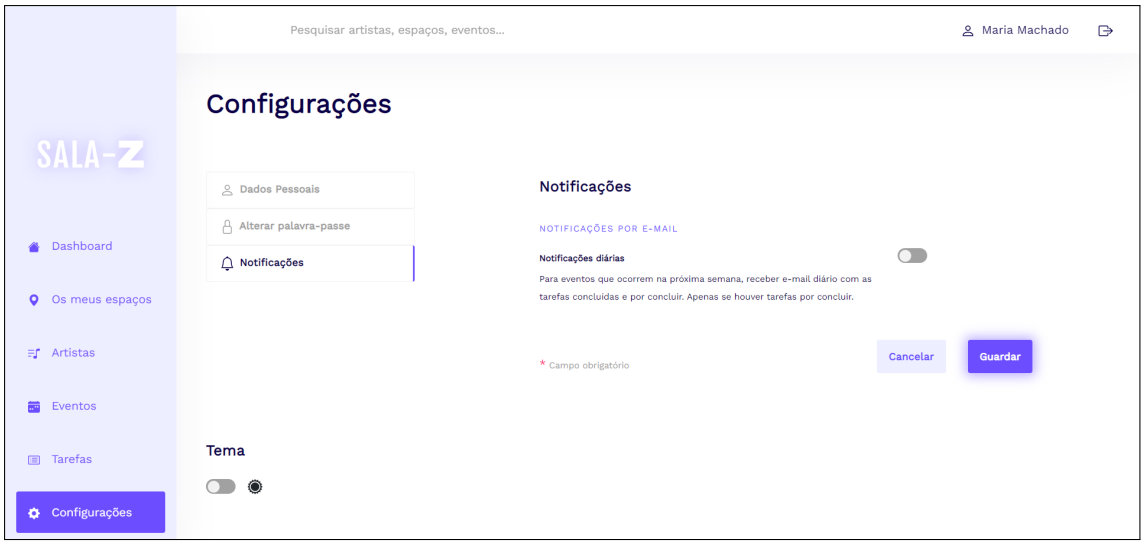

Figura B.15: Configuração de relatórios de um gestor de uma *venue* (Plataforma Sala-Z 2.1)

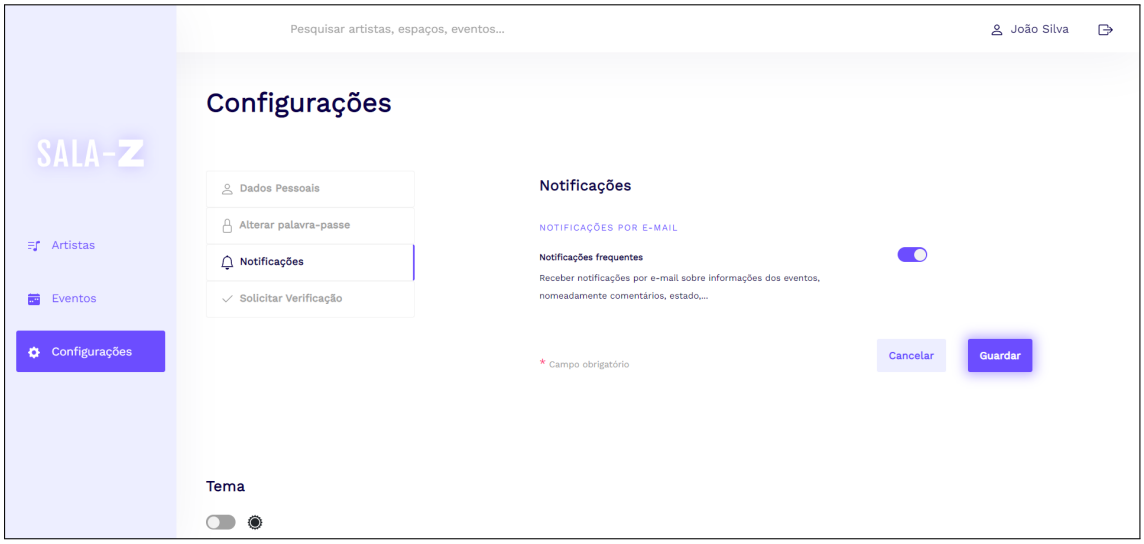

Figura B.16: Configuração de relatórios de um gestor de um artista (Plataforma Sala-Z 2.1)

Apêndice C

Plano de Testes

Módulo: Gestão de *Venues*

Tema: Painel de Administração

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
7 1	1	Adicionar um novo utilizador à *venue*	Adicionar um novo utilizador	- Editar uma *venue* - Selecionar que quer adicionar um novo colaborador - Inserir campos necessários - Adicionar novo colaborador	Convite enviado para o novo utilizador	Convite enviado para o novo utilizador	**Passou**
2	2	Reenviar convite a um novo utilizador para a *venue*	Reenviar novamente convite a um utilizador removido	- Editar uma *venue* - Selecionar que quer adicionar um novo colaborador - Selecionar *dropdown* para adicionar novamente utilizador removido - Campos preenchidos automaticamente após seleção - Adicionar novo colaborador - Guardar as alterações efetuadas	Utilizador novamente adicionado ao espaço	Utilizador novamente adicionado ao espaço	**Passou**
3	3	Listar utilizadores com acesso à *venue*	Listar utilizadores com acesso à *venue*	- Selecionar uma *venue* - Ver todos os utilizadores da *venue*	Ver a lista de utilizadores com acesso à *venue*	Ver a lista de utilizadores com acesso à *venue*	**Passou**
4	4	Remover utilizador da *venue*	Remover utilizador da *venue*	- Editar uma *venue* - Ver todos os utilizadores da *venue* - Selecionar um utilizador para ser removido - Confirmar remoção do utilizador - Guardar as alterações efetuadas	*Venue* com lista de utilizadores exceto o removido	*Venue* com lista de utilizadores exceto o removido	**Passou**

Tabela C.1: Gestão de *Venues* - Plano de Testes do painel de administração

Tema: Gestão de Inventário

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
5	5	Criar item no inventário	Verificar se o item é criado e adicionado corretamente à base de dados	- Escolher a *venue* - Criar um novo item - Preencher os campos necessários - Guardar novo item	Item adicionado com sucesso à *venue*	Item adicionado com sucesso à *venue*	**Passou**
6	6	Criar item no inventário com um número de série	Não desenvolvido				
7	7	Listar itens da *venue*	Verificar se a listagem de Itens associados à venue se encontra de acordo com a base de dados	- Escolher a *venue* - Ver a lista de inventários	Itens listados de acordo com a base de dados	Itens listados de acordo com a base de dados	**Passou**
8	8	Ordenar itens na lista de itens	Verificar se a ordenação da lista de Itens se encontra funcional	- Escolher a *venue* - Ver a lista de inventários - Ordenar Itens	Itens ordenados de acordo com as base de dados	Itens ordenados de acordo com as base de dados	**Passou**
9	9	Pesquisar item na lista de itens	Verificar se a pesquisa da lista de Itens se encontra funcional	- Escolher a *venue* - Ver a lista de inventários - Pesquisar por nome na lista Itens	Itens listados de acordo com a base de dados e pesquisa efetuada	Itens listados de acordo com a base de dados e pesquisa efetuada	**Passou**
10	10	Filtrar itens na lista de itens	Verificar se a filtragem da lista de Itens se encontra funcional	- Escolher a *venue* - Ver a lista de inventários - Filtrar Itens na lista	Itens listados de acordo com a base de dados e filtragem efetuada	Itens listados de acordo com a base de dados e filtragem efetuada	**Passou**
11	11	Ver detalhes de um item	Verificar se os detalhes do evento está de acordo com a informação na base de dados	- Escolher a *venue* - Ver a lista de inventários - Filtrar Itens na lista - Selecionar o item - Ver os detalhes do item	Detalhe do item apresentado conforme a base de dados	Detalhe do item apresentado conforme a base de dados	**Passou**
12	12	Editar item	Verificar se um item é editável e se a informação é guardada na base de dados	- Escolher a *venue* - Ver a lista de inventários - Filtrar Itens na lista - Selecionar o item - Ver os detalhes do item - Editar um item - Efetuar as alterações - Guardar	Edições do item apresentado conforme a base de dados	Edições do item apresentado conforme a base de dados	**Passou**
13	13	Duplicar item	Verificar se a duplicação de um item se encontra funcional	- Escolher a *venue* - Ver a lista de inventários - Filtrar Itens na lista - Selecionar o menu de um item - Duplicar item	Item é duplicado e adicionado à lista	Item é duplicado e adicionado à lista	**Passou**
14	14	Eliminar item	Verificar se a remoção de um item da lista de Itens se encontra funcional	- Escolher a *venue* - Ver a lista de inventários - Filtrar Itens na lista - Selecionar o menu de um item - Eliminar item	Item é eliminado e removido da lista	Item é eliminado e removido da lista	**Passou**
15	15	Associar um item a um evento	Verificar se a associação de um item à lista de eventos se encontra funcional	- Escolher a *venue* - Ver a lista de inventários - Filtrar Itens na lista - Selecionar o menu de um item - Ver os detalhes do item - Associar um evento	Item é associado ao evento	Item é associado ao evento	**Passou**

Tabela C.2: Gestão de *Venues* - Plano de Testes da gestão de inventário

Tema: Gestão de Artistas

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
16	16	Criar artista com importação	Verificar se o artista é importado e adicionado corretamente à base de dados	- Selecionar a lista de artistas - Adicionar novo artista - Pesquisar por nome de artista - Selecionar um artista - Preencher os campos necessários - Guardar alterações	Artista é criado com sucesso e adicionado à lista de artistas	Artista é criado com sucesso e adicionado à lista de artistas	**Passou**
17	17	Criar artista manualmente	Verificar se o artista é criado e adicionado corretamente à base de dados	- Selecionar a lista de artistas - Adicionar novo artista - Selecionar a criação manual - Preencher os campos necessários - Guardar alterações	Artista é criado com sucesso e adicionado à lista de artistas	Artista é criado com sucesso e adicionado à lista de artistas	**Passou**
18	18	Listar artistas	Verificar se a listagem de artistas associados à venue se encontra de acordo com a base de dados	- Selecionar a lista de artistas	Artistas listados de acordo com a base de dados e pesquisa efetuada	Artistas listados de acordo com a base de dados e pesquisa efetuada	**Passou**
19	19	Listar artistas favoritos	Verificar se a listagem de artistas favoritos associados à *venue* se encontra de acordo com a base de dados	- Selecionar a lista de artistas - Adicionar uma artista como favorito - Filtrar por favoritos	Artistas favoritos listados de acordo com a base de dados e pesquisa efetuada	Artistas favoritos listados de acordo com a base de dados e pesquisa efetuada	**Passou**
20	20	Procurar artista na lista de artistas	Verificar se a pesquisa da lista de artistas se encontra funcional	- Ver a lista de artistas - Pesquisar por nome na lista artistas	Artistas listados de acordo com a base de dados e pesquisa efetuada	Artistas listados de acordo com a base de dados e pesquisa efetuada	**Passou**
21	21	Filtrar lista de artistas	Verificar se a filtragem da lista de artistas se encontra funcional	- Ver a lista de artistas - Filtrar artistas na lista	Artistas listados de acordo com a base de dados e filtragem efetuada	Artistas listados de acordo com a base de dados e filtragem efetuada	**Passou**
22	22	Ver detalhes de um artista	Verificar se os detalhes do artista está de acordo com a informação na base de dados	- Ver a lista de artistas - Selecionar o artista - Ver os detalhes do artista	Detalhe do artista apresentado conforme a base de dados	Detalhe do artista apresentado conforme a base de dados	**Passou**
23	23	Editar artista	Verificar se um artista é editável e se a informação é guardada na base de dados	- Ver a lista de artistas - Selecionar o artista - Ver os detalhes do artista - Editar um artista - Efetuar as alterações - Guardar	Artistas listados de acordo com a base de dados e filtragem efetuada	Artistas listados de acordo com a base de dados e filtragem efetuada	**Passou**
24	24	Eliminar artista	Verificar se a remoção de um artista da lista de itens se encontra funcional	- Escolher um artista - Ver a lista de artistas - Selecionar o artista - Ver os detalhes do artista - Eliminar artista - Confirmar remoção	Artista é eliminado e removido da lista	Artista é eliminado e removido da lista	**Passou**

Tabela C.3: Gestão de *Venues* - Plano de testes da gestão de artistas

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
25	25	Marcar um artista como favorito	Marcar um artista como favorito	- Escolher um artista - Ver a lista de artistas - Selecionar o artista - Ver os detalhes do artista - Selecionar um artista como favorito	Item é eliminado e removido da lista	Item é eliminado e removido da lista	**Passou**
	26	Marcar um artista como favorito	Marcar um artista como favorito	- Ver a lista de artistas - Selecionar um artista como favorito	Item é eliminado e removido da lista	Item é eliminado e removido da lista	**Passou**
26	27	Ver as estatísticas associadas a artistas	Não desenvolvido				

Tabela C.4: Gestão de *Venues* - Plano de testes da gestão de artistas

Tema: *Dashboard*

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
27	28	Ver as estatísticas associadas à *venue*	Não desenvolvido				

Tabela C.5: Gestão de *Venues* - Plano de testes da *dashboard*

Tema: Gestão de tarefas de um evento

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
28	29	Rápida criação da tarefa	Não desenvolvido				
29	30	Criar tarefa com detalhes	Verificar se a tarefa é adicionada corretamente à base de dados	- Selecionar a lista de tarefas - Adicionar nova tarefa - Ver os detalhes da tarefas - Preencher os campos necessários - Guardar alterações	Tarefas listadas de acordo com a base de dados e pesquisa efetuada	Tarefas listadas de acordo com a base de dados e pesquisa efetuada	**Passou**
30	31	Listar tarefas	Verificar se a lista de tarefas é apresentada corretamente como a base de dados	- Ver a lista de tarefas	Tarefa é criada com sucesso e adicionada à lista de tarefas	Tarefa é criada com sucesso e adicionada à lista de tarefas	**Passou**
31	32	Ordenar lista de tarefas	Verificar se a ordenação da lista de tarefas se encontra funcional	- Ver a lista de tarefas - Ordenar tarefas	Tarefas ordenadas de acordo com as base de dados	Tarefas ordenadas de acordo com as base de dados	**Passou**
32	33	Ver detalhes de uma tarefa	Verificar se os detalhes da tarefa está de acordo com a informação na base de dados	- Ver a lista de tarefas - Selecionar a tarefa - Ver os detalhes da tarefa	Detalhe da tarefa apresentada conforme a base de dados	Detalhe da tarefa apresentada conforme a base de dados	**Passou**

Tabela C.6: Gestão de *Venues* - Plano de testes da gestão de tarefas de um evento

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
33	34	Procura tarefa na lista de tarefas	Verificar se a pesquisa da lista de tarefas se encontra funcional	- Ver a lista de tarefas - Pesquisar na lista tarefas	Tarefas listadas de acordo com a base de dados e pesquisa efetuada	Tarefas listadas de acordo com a base de dados e pesquisa efetuada	**Passou**
34	35	Filtrar lista de tarefas	Verificar se a filtragem da lista de tarefas se encontra funcional	- Ver a lista de tarefas - Filtrar tarefas na lista	Tarefas listadas de acordo com a base de dados e filtragem efetuada	Tarefas listadas de acordo com a base de dados e filtragem efetuada	**Passou**
35	36	Editar tarefa	Verificar se a filtragem da lista de tarefas se encontra funcional	- Ver a lista de tarefas - Filtrar tarefas na lista	Tarefas listadas de acordo com a base de dados e filtragem efetuada	Tarefas listadas de acordo com a base de dados e filtragem efetuada	**Passou**
36	37	Duplicar tarefa	Verificar se a duplicação de uma tarefa se encontra funcional	- Ver a lista de tarefas - Selecionar o menu de uma tarefa - Duplicar tarefa	Item é duplicado e adicionado à lista	Item é duplicado e adicionado à lista	**Passou**
37	38	Eliminar tarefa	Verificar se a remoção de uma tarefa da lista se encontra funcional	- Ver a lista de tarefas - Selecionar a tarefa - Ver os detalhes da tarefa - Eliminar tarefa - Confirmar remoção	Item é duplicado e adicionado à lista	Item é duplicado e adicionado à lista	**Passou**
38	39	Ver a atividade associada à tarefa	Verificar se a atividade da uma tarefa da lista se encontra de acordo com a base dados	- Ver a lista de tarefas - Selecionar a tarefa - Ver os detalhes da tarefa - Ver a atividade	Atividade encontra se de acordo com a base de dados	Atividade encontra se de acordo com a base de dados	**Passou**
39	40	Adicionar comentário à tarefa	Verificar se o comentário é adicionado à atividade da uma tarefa	- Ver a lista de tarefas - Selecionar a tarefa - Ver os detalhes da tarefa - Ver a atividade - Adicionar novo comentário	Comentário adicionado à atividade	Comentário adicionado à atividade	**Passou**
40	41	Editar comentário	Não desenvolvido				
41	42	Eliminar comentário	Não desenvolvido				

Tabela C.7: Gestão de *Venues* - Plano de testes da gestão de tarefas de um evento

Tema: Gestão de tarefas de uma *venue*/utilizador

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
42	43	Ver as estatísticas associadas a tarefas	Não desenvolvido				

Tabela C.8: Gestão de *Venues* - Plano de testes da gestão de tarefas da *sala*venue e utilizador

Tema: Comunicação entre *stakeholders*

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
43	44	Mencionar outros utilizadores da *venue* com comentários	Não desenvolvido				

Tabela C.9: Gestão de *Venues* - Plano de testes da comunicação entre *stakeholders*

Tema: Notificações

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
44	45	Ver notificações	Não desenvolvido				

Tabela C.10: Gestão de *Venues* - Plano de testes das notificaçõess

Tema: Configurar relatórios de eventos

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
45	46	Configurar relatórios de eventos	Notificações diárias	- Aceder às configurações - Ativar a receção de notificações diárias	Recebe diariamente no email o relatório	Recebe diariamente no email o relatório	**Passou**
			Notificações semanais	Não desenvolvido			
			Receber notificações no painel de controlo	Não desenvolvido			
			Barra de pesquisa	Não desenvolvido			

Tabela C.11: Gestão de *Venues* - Plano de testes da configuração de relatórios de eventos

Tema: Gerir templates de eventos

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
46	47	Criar template	Não desenvolvido				
47	48	Listar templates	Não desenvolvido				
48	49	Editar template	Não desenvolvido				
49	50	Eliminar template	Não desenvolvido				
50	51	Criar um evento a partir de um template	Não desenvolvido				
51	52	Criar um template a partir de um evento	Não desenvolvido				

Tabela C.12: Gestão de *Venues* - Plano de testes da gestão de templates de eventos

Tema: Gerir Eventos

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
52	53	Duplicar evento	Não desenvolvido				

Tabela C.13: Gestão de *Venues* - Plano de testes da gestão de eventos

Módulo: Gestão de Artistas

Tema: Painel de Administração

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
53	54	Enviar convite a um novo utilizador para o meu artista	Adicionar um novo utilizador	- Editar um artista - Selecionar que quer adicionar um novo colaborador - Inserir campos necessários - Adicionar novo colaborador	Convite enviado para o novo utilizador	Convite enviado para o novo utilizador	**Passou**
54	55	Reenviar convite a um novo utilizador para a gerir o meu artista	Reenviar novamente convite a um utilizador removido	- Editar um artista - Selecionar que quer adicionar um novo colaborador - Selecionar *dropdown* para adicionar novamente utilizador removido - Campos preenchidos automaticamente após seleção - Adicionar novo colaborador - Guardar as alterações efetuadas	Utilizador novamente adicionado ao artista	Utilizador novamente adicionado ao artista	**Passou**

Tabela C.14: Gestão de Artistas - Plano de testes do painel de administração

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
55	56	Listar utilizadores com acesso ao meu artista	Listar utilizadores com acesso ao meu artista	- Selecionar um artista - Ver todos os utilizadores do artista	Ver a lista de utilizadores com acesso ao artista	Ver a lista de utilizadores com acesso ao artista	**Passou**
56	57	Remover utilizador do artista	Remover utilizador do artista	- Editar um artista - Ver todos os utilizadores do artista - Selecionar um utilizador para ser removido - Confirmar remoção do utilizador - Guardar as alterações efetuadas	Artista com lista de utilizadores exceto o removido	Artista com lista de utilizadores exceto o removido	**Passou**

Tabela C.15: Gestão de Artistas - Plano de testes da gestão de artistas

Tema: Gestão de Artistas

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
57	58	Listar artistas	Verificar se a listagem de artistas associados ao artista se encontra de acordo com a base de dados	- Selecionar a lista de artistas	Artistas listados de acordo com a base de dados e pesquisa efetuada	Artistas listados de acordo com a base de dados e pesquisa efetuada	**Passou**
58	59	Filtrar lista de artistas	Verificar se a filtragem da lista de artistas se encontra funcional	- Ver a lista de artistas - Filtrar artistas na lista	Artistas listados de acordo com a base de dados e filtragem efetuada	Artistas listados de acordo com a base de dados e filtragem efetuada	**Passou**
59	60	Editar artista	Verificar se um artista é editável e se a informação é guardada na base de dados	- Ver a lista de artistas - Selecionar o artista - Ver os detalhes do artista - Editar um artista - Efetuar as alterações - Guardar	Artistas listados de acordo com a base de dados e filtragem efetuada	Artistas listados de acordo com a base de dados e filtragem efetuada	**Passou**
60	61	Ver detalhes de um artista	Verificar se os detalhes do artista está de acordo com a informação na base de dados	- Ver a lista de artistas - Selecionar o artista - Ver os detalhes do artista	Detalhe do artista apresentado conforme a base de dados	Detalhe do artista apresentado conforme a base de dados	**Passou**
61	62	Eliminar artista	Verificar se a remoção de um artista da lista de itens se encontra funcional	- Escolher um artista - Ver a lista de artistas - Selecionar o artista - Ver os detalhes do artista - Eliminar artista - Confirmar remoção	Artista é eliminado e removido da lista	Artista é eliminado e removido da lista	**Passou**
62	63	Criar artista com importação	Verificar se o artista é importado e adicionado corretamente à base de dados	- Selecionar a lista de artistas - Adicionar novo artista - Pesquisar por nome de artista - Selecionar um artista - Preencher os campos necessários - Guardar alterações	Artista é criado com sucesso e adicionado à lista de artistas	Artista é criado com sucesso e adicionado à lista de artistas	**Passou**

Tabela C.16: Gestão de Artistas - Plano de testes da gestão de artistas

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
63	64	Criar artista manualmente	Verificar se o artista é criado e adicionado corretamente à base de dados	- Selecionar a lista de artistas - Adicionar novo artista - Selecionar a criação manual - Preencher os campos necessários - Guardar alterações	Artista é criado com sucesso e adicionado à lista de artistas	Artista é criado com sucesso e adicionado à lista de artistas	**Passou**
64	65	Verificar artista	Verificar se a adição do selo de verificação ao perfil se encontra funcional	- Selecionar as configurações - Selecionar que pretende obter de verificação	É adicionado a verificação ao perfil do gestor do artista	É adicionado a verificação ao perfil do gestor do artista	**Passou**

Tabela C.17: Gestão de Artistas - Plano de testes da gestão de artistas

Tema: *Dashboard*

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
65	66	Ver as estatísticas associadas a artistas	Não desenvolvido				
66	67	Filtrar lista de artistas na tabela	Não desenvolvido				
67	68	Ordenar lista de artistas na tabela	Não desenvolvido				

Tabela C.18: Gestão de Artistas - Plano de testes da *dashboard*

Tema: Gestão de Eventos

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
68	69	Listar eventos do artista	Verificar se os eventos que estou associados estão listados na lista de eventos	- Aceder à lista de eventos - Ver a lista de eventos	Lista de eventos apresentada de acordo com a base de dados	Lista de eventos apresentada de acordo com a base de dados	**Passou**
69	70	Filtrar lista de eventos do artista	Verificar se os eventos que estou associados estão listados na lista de eventos	- Aceder à lista de eventos - Ver a lista de eventos - Filtrar a lista de eventos	Lista de eventos filtrada apresentada de acordo com a base de dados	Lista de eventos filtrada apresentada de acordo com a base de dados	**Passou**
70	71	Ver detalhes do evento	Verificar se os eventos que estou associados estão listados na lista de eventos	- Aceder à lista de eventos - Ver a lista de eventos - Visualizar detalhes	Apresentação dos detalhes, de acordo com a base de dados	Apresentação dos detalhes, de acordo com a base de dados	**Passou**

Tabela C.19: Gestão de Artistas - Plano de testes da gestão de eventos

Tema: Comunicação entre *stakeholders*

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
71	72	Mencionar outros utilizadores da evento com comentários	Não desenvolvido				

Tabela C.20: Gestão de Artistas - Plano de testes da comunicação entre *stakeholders*

Tema: Notificações

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
72	73	Ver notificações	Não desenvolvido				

Tabela C.21: Gestão de Artistas - Plano de testesd das notificações

Tema: Configurar relatórios de eventos

US	ID	Cenário	Descrição	Passos	Resultado Esperado	Resultado Obtido	Resultado
73	74	Configurar relatórios de eventos	Notificações frequentes	- Aceder às configurações - Ativar a receção de notificações frequentes	Recebe diariamente no email qualquer alterações nos meus eventos	Recebe frequente-mente no email qualquer alterações nos meus eventos	**Passou**
			Receber notificações no painel de controlo	Não desenvolvido			
			Barra de pesquisa	Não desenvolvido			

Tabela C.22: Gestão de Artistas - Plano de testes da configuração de relatórios de eventos