INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Plataforma de gestão partilhada de cursos de divulgação científica

Susana Mendes Cardoso

Licenciado

Projecto Final para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Prof. Doutora Cátia Vaz
Prof. Doutora Paula Louro

Júri:

Presidente: Prof. Doutor Nuno Datia

Vogais: Prof. Doutora Cátia Vaz
Prof. Doutor José Simão

Setembro, 2023

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Plataforma de gestão partilhada de cursos de divulgação científica

Susana Mendes Cardoso

Licenciado

Projecto Final para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Prof. Doutora Cátia Vaz
Prof. Doutora Paula Louro

Júri:

Presidente: Prof. Doutor Nuno Datia

Vogais: Prof. Doutora Cátia Vaz
Prof. Doutor José Simão

Setembro, 2023

I never am really satisfied that I understand anything; because, understand it well as I may, my comprehension can only be an infinitesimal fraction of all I want to understand about the many connections and relations which occur to me, how the matter in question was first thought of or arrived at…

Ada Lovelace

Agradecimentos

Às orientadoras, pela sua disponibilidade e compreensão neste longo processo. Ao Instituto Superior de Engenharia de Lisboa, que se tornou na minha segunda casa, por todos estes anos de formação, pela *Praxe*, pelos colegas e amigos. Aos meus pais, por me terem dado tudo, pelo apoio incondicional e serem uma fonte de inspiração de perseverança. À minha irmã, Inês, por ter seguido um caminho tão diferente do meu. Ao André, por estar sempre lá.

Resumo

As instituições académicas organizam ao longo do ano lectivo vários eventos com distintos objectivos, deste a captação de potênciais alunos à partilha de conhecimentos na comunidade. Estes eventos são habitualmente organizados pelo corpo docente dessas mesmas instituições. O ISEL não é excepção, e organiza também os seus próprios eventos, tais como o ISEL Energy Week, o Programa de Ocupação Científica de Jovens nas Férias e o ISEL ALIVE.

Até agora as equipas de organização destes eventos utilizam múltiplas ferramentas para organizar e promover os mesmos. Esta forma de trabalhar torna-se complexa devido à dispersão de informação entre as várias ferramentas.

A solução mais imediata seria a utilização de uma plataforma de gestão de eventos disponível no mercado, o que provavelmente tornaria o processo mais eficiente e centralizado numa única aplicação. Estas plataformas suportam vários tipos de eventos (físicos, virtuais e híbridos), tem soluções integradas de divulgação de eventos e inscrições. Contudo, esta pode não ser uma solução viável, pois estas plataformas não são inteiramente adequadas a eventos desta natureza e acarretam custos monetários elevados para a organização, sendo esta última razão por si só suficiente para inviabilizar a sua utilização, dado que são eventos académicos.

Com o apoio da equipa do ISEL ALIVE na identificação das dificuldades sentidas nestas tarefas, pretende-se com este projecto o planeamento e implementação de uma aplicação de gestão de eventos direcionada a qualquer evento académico cujas necessidades não conseguem ser acomodadas nas plataformas existentes no mercado, por forma a melhorar o seu fluxo de trabalho, centralizando a informação numa única aplicação por um custo reduzido.

Palavras-chave: Palavras-chave: Gestão de eventos, BPMN, API, Spring, React

Abstract

Academic institutions organize various events throughout the academic year with different objectives, from attracting potential students to sharing knowledge in the community. These events are usually organized by the teaching staff of these same institutions. ISEL is no exception, and also organizes its own events, such as the ISEL Energy Week, the Youth Scientific Occupation Program on Holidays and ISEL ALIVE.

Until now, the teams organizing these events use multiple tools to organize and promote them. This way of working becomes complex due to the dispersion of information between the various tools.

The most immediate solution would be to use an event management platform available on the market, which would probably make the process more efficient and centralized in a single application. These platforms support various types of events (physical, virtual and hybrid), have integrated solutions for promoting events and registrations. However, this may not be a viable solution, as these platforms are not entirely suitable for events of this nature and entail high monetary costs for the organization, the latter reason alone being sufficient to make their use unfeasible, given that they are academic events.

With the support of the ISEL ALIVE team in identifying the difficulties experienced in these tasks, this project aims to plan and implement an event management application aimed at any academic event whose needs cannot be accommodated on existing platforms on the market, in order to improve there workflow, centralizing information in a single application at a reduced cost.

Keywords: Event management, BPMN, API, Spring, React

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API *Application Programming Interfaces*. 16, 29, 31, 34

BPM *Business Process Management*. 2, 13, 14, 23
BPMN *Business Process Model and Notation*. 14, 23, i

CD *Continuous Deployment*. 19
CI *Continuous Integration*. 19
CRM *Customer Relationship Management*. 7

FTP *File Transfer Protocol*. 29, 30

UML *Unified Modeling Language*. 14

1

Introdução

O papel das instituições académicas na sociedade não se limita ao processo tradicional de transição dos alunos do ensino secundário para o ensino superior. Envolve também a promoção de partilha de conhecimento na comunidade, assim como a captação de potenciais alunos que se encontrem em percursos menos comuns, e poderem consequentemente necessitar de uma abordagem diferente para serem integrados neste nível de ensino. Para os fins mencionados, é comum nestas instituições a realização de diversos eventos de carácter público.

No ISEL são organizados vários eventos ao longo do ano lectivo, distintos entre si tanto na estrutura como na organização, habitualmente organizados por membros do corpo docente do instituto, entre outros, o ISEL Energy Week, o Programa de Ocupação Científica de Jovens nas Férias e o ISEL ALIVE. O ISEL Energy Week é um evento direcionado à comunidade académica a partir do 10º ano e a parceiros sociais com o intuito de divulgar o trabalho e instalações de serviços energéticos. Neste são realizadas visitas a instalações eléctricas, actividades e desafios temáticos. O Programa de Ocupação Científica de Jovens nas Férias (OCJF), promovido pela Ciência Viva — Agência Nacional para a Cultura Científica e Tecnológica, em que participam várias instituições de ensino superior, que organizam estágios destinados a alunos de vários níveis académicos, proporcionam uma oportunidade de aproximação à realidade da investigação científica e tecnológica. O ISEL ALIVE é um curso de Verão destinado a alunos do ensino secundário, oferecendo aos jovens uma ferramenta auxiliar no seu processo de decisão relativo ao curso superior que pretendem ingressar, apresentando as diferentes

oportunidades formativas oferecidas. Os participantes interagem com as várias áreas departamentais através de actividades lúdicas que lhes permitem familiarizar-se com os cursos leccionados em cada departamento e com o campus.

Até ao momento, têm sido utilizadas várias ferramentas (aplicações tradicionais de processamento de texto e folhas de cálculo) para organizar e promover estes eventos, porém, têm sido observadas algumas limitações nestas ferramentas face às necessidades dos organizadores, nomeadamente a dispersão de informação, a dificuldade de cruzamento de dados entre aplicações e na partilha de informação entre múltiplos intervenientes. Dadas essas limitações, considerou-se oportuno o desenvolvimento de um projecto para a construção de uma plataforma gratuita que permita às entidades organizadoras destes eventos centralizar a sua gestão. O mercado dispõe de diversas aplicações para este fim, no entanto, estas podem acarretar custos monetários elevados e/ou serem desadequadas às necessidades específicas destes eventos [1–3].

1.1 Objectivos

Este projecto visa a construção de uma aplicação de gestão de eventos, que permita aos grupos de trabalho do ISEL optimizarem os seus processos de organização e gestão desses mesmos eventos. Para tal, a equipa do ISEL ALIVE disponibilizou-se para apoiar na identificação das necessidades do evento que organiza. Estas serão utilizadas para definir as especificações e testar a implementação da solução proposta. Contudo, a solução tem de permitir acomodar os restantes eventos académicos na mesma plataforma.

1.2 Estrutura do documento

O presente documento encontra-se organizado em sete capítulos: trabalho relacionado, requisitos, conceitos e tecnologias, solução proposta, implementação, avaliação e conclusão. No capítulo 2 é apresentado o tipo de eventos feitos no ISEL e o estado da arte nas aplicações de gestão de eventos já existentes. No capítulo 3 são apresentados vários casos de uso de projecto, com foco nos requisitos funcionais e não funcionais que devem ser cumpridos. No capítulo 4 são apresentadas as tecnologias escolhidas para o projecto: uma introdução teórica do BPM, uma breve análise às possíveis linguagem e *framework*s, a ambientes de execução e gestão de dados. O capítulo 5 expõe a arquitectura lógica e fisíca a seguir. No capítulo 6 é apresentada a implementação do projecto.

No capítulo 7 são apresentados os critérios de avaliação do projecto. No capítulo 8 apresenta-se as conclusões e indica algumas direcções para trabalho futuro.

2

Trabalho Relacionado

Neste capítulo serão apresentados os cursos de divulgação cientifica, os resultados da pesquisa por aplicações de gestão de eventos e uma análise geral das aplicações consideradas mais relevantes.

2.1 Cursos de divulgação científica

Os cursos de divulgação científica organizados no ISEL ao longo do ano lectivo, são habitualmente organizados por membros do corpo docente do instituto. Alguns exemplos deste tipo de eventos são o ISEL Energy Week, o Programa de Ocupação Científica de Jovens nas Férias e o ISEL ALIVE. O ISEL Energy Week é um evento direcionado à comunidade académica a partir do 10º ano e a parceiros sociais com o intuito de divulgar o trabalho e instalações de serviços energéticos. Neste são realizadas visitas a instalações eléctricas, actividades e desafios temáticos. O Programa de Ocupação Científica de Jovens nas Férias (OCJF), promovido pela Ciência Viva — Agência Nacional para a Cultura Científica e Tecnológica, em que participam várias instituições de ensino superior, que organizam estágios destinados a alunos de vários níveis académicos, proporcionam uma oportunidade de aproximação à realidade da investigação científica e tecnológica. O ISEL ALIVE é um curso de Verão destinado a alunos do ensino secundário, oferecendo aos jovens uma ferramenta auxiliar no seu processo de decisão relativo ao curso superior que pretendem ingressar, apresentando as diferentes oportunidades formativas oferecidas. Os participantes interagem com as várias áreas

departamentais através de actividades lúdicas que lhes permitem familiarizar-se com os cursos leccionados em cada departamento e com o campus.

2.2 Plataformas de gestão de Eventos

Dadas as dificuldades apresentadas pelas equipas da organização de eventos na procura de uma plataforma que satisfaça as necessidades específicas de gestão, põe-se a questão recorrente neste tipo de abordagens: existe alguma aplicação que os resolva? Torna-se então inevitável, antes de executar qualquer tipo de desenvolvimento, a pesquisa no mercado por aplicações de gestão de eventos, e uma posterior análise da sua adequação à realidade deste evento em particular.

O mercado dispõe de uma variedade de plataformas de gestão de eventos, habitualmente adaptadas a casos de utilização específicos. Embora todas tenham o mesmo propósito, existem sempre adaptações feitas a pensar em determinados utilizadores ou categorias de eventos. Nesta secção serão analisadas algumas funcionalidades de três plataformas (Bizzabo, Splash, Socio) para mostrar o que o mercado oferece neste momento. Foram escolhidas estas plataformas por apresentarem melhor avaliação em plataformas agregadoras [4] [5]. O estudo destas plataformas será dividido em antes, durante e após o evento.

2.2.1 Antes do Evento

Existem determinadas tarefas e acções que devem ser executadas antes do evento. Estas plataformas disponibilizam a elaboração do programa/sessões do evento, elaboração um formulário de inscrição através de um site ou de envio de email e pagamentos *online*.

As plataformas Bizzabo, Splash e Socio estão adaptadas para a realização de eventos presenciais, virtuais e híbridos [1] [2] [3]. Todas as plataformas disponibilizam a criação do site do evento com algum nível de personalização, oferecem um formulário de inscrição que pode ser adaptado pelos organizadores [6] [7] [8]. Relativamente ao envio de email apenas a plataforma Splash permite uma personalização do mesmo [8]. Quanto aos pagamentos, as plataformas Splash e Socio tem uma integração obrigatória com a plataforma Stripe para a gestão dos pagamentos [9] [10].

2.2.2 Durante o Evento

Durante o evento, existem essencialmente duas tarefas a realizar: acreditação (*check-in*) e controlo de entradas e saídas. Em todas as plataformas isto é possível utilizando uma aplicação móvel ou a leitura de códigos de barras, ou ainda o uso de *QR-Codes*. O controlo de entradas e saídas permite um controlo rigoroso de todos os serviços e das presenças em cada sessão, por forma a certificar que o participante esteve nas sessões.

Como estas plataformas permitem eventos virtuais ou híbridos, todas disponibilizam a possibilidade de se fazer transmissão dos seus eventos, no caso do Socio este tem uma solução integrada [11], no caso do Splash e Bizzabo estas permitem a integração com plataformas de *streaming* como o YouTube e o Zoom [12][13]. As plataformas possuem uma aplicação móvel que permite a acreditação dos eventos presenciais ou híbridos e permitem também visualizar a agenda do evento e fazer *networking* entre participantes.[14][15][16]

2.2.3 Depois do Evento

Após o término do evento, por forma a ter-se toda a informação necessária para tomar decisões sobre o próximo evento, é fundamental o desenvolvimento de relatórios de satisfação e formular relatórios de estatísticas sobre o mesmo. Caso tenham existido sessões para certificação, é neste momento que devem ser emitidos os devidos certificados para os participantes.

Todas as plataformas recolhem dados estatísticos ao longo do evento e possuem ou integram tecnologias *Customer Relationship Management* (CRM) [17] [18] [19]. Um sistema CRM providência a uma organização uma forma de gerir as interacções com os seus clientes. Não foi encontrada qualquer informação relativa à emissão de certificados em nenhuma das plataformas.

2.3 Trabalhos académicos

Para além de avaliar as plataformas existentes no mercado, validou se tambem no meio académico se existiria alguma solução proposta que pudesse ser utilizada para a gestão dos cursos de divulgação científica. Os trabalhos validados foram o *Cardinal Connect* [20], o *Event Management Solution* [21] e o *College Activity Management System* [22]. Estas propostas possuem paralelos na organização dos seus eventos com os cursos de divulgação científica principalmente na divulgação dos eventos, contudo existe uma

divergencia, nomeadamente nas inscrições aos eventos. A *Cardinal Connect* e a *Event Management Solution* possuem uma solução de inscrição nos eventos que divulgam, no entanto, estas não possuem a noção de candidatura: uma inscrição que pode ser aceite ou rejeitada por parte da organização. A proposta do *College Activity Management System* não possui uma solução relativa a inscrições. Ao existir uma diferenciação entre inscrição e candidatura faz com que nenhuma destas soluções propostas seja viável para os cursos de divulgação científica apresentados.

3

Requisitos

Este capítulo apresenta os requisitos funcionais e não funcionais, recolhidos junto da equipa do evento ISEL ALIVE, para uma ferramenta de apoio à organização do mesmo, assim como os casos de uso pretendidos.

3.1 Requisitos Funcionais

Os requisitos funcionais definem o que o sistema deve fazer e como se deve comportar ou reagir. Os requisitos funcionais identificados são os seguintes:

- A plataforma deve suportar vários tipos de utilizadores:
  - administrador
  - gestor de evento
  - orador
  - participante
- A plataforma deve suportar o registo de utilizadores.
- Pretende-se a possibilidade de criar, editar e eliminar um evento.
- Deve existir um formulário de inscrição associado a um evento.

- O formulário deve ser dinâmico, permitindo várias edições (estado rascunho) tornado-se estático aquando da sua publicação.
- O formulário deve ser de acesso privado (necessidade de autenticação).
- Um evento deve ser composto por actividades.
- A cada actividade está associado um orador/orientador (formador).
- Caso exista actividade exterior, esta deve registar a hora de saída e de chegada dos participantes.
- Deve existir um registo de presenças dos participantes durante um evento.
- Se o participante for menor de idade deve ser indicado o contacto de email do encarregado de educação.
- Caso o participante não esteja presente deve-se notificar o encarregado de educação.
- Devem ser geradas notificações sempre que existir mudança de estado da candidatura e do evento.
- Deve existir um formulário para a submissão do comprovativo de pagamento.
- Deve ser emitido um certificado após um evento.
- Deve ser possível gerir os modelos de email.

3.2 Requisitos Não Funcionais

Requisitos não funcionais são qualificações dos requisitos funcionais ou do produto geral. Foram identificados os seguintes:

- Não deve lidar com pagamentos.
- Manutenção de histórico de edições anteriores.
- Armazenamento de ficheiros separado (FTP).
- Reutilizável para outros eventos
- Implementação compatível com as máquinas do ISEL.
- Testes automatizados
- Deploy/build automatizado

3.3 Casos de Uso

Um caso de uso é uma descrição, na perspetiva do utilizador, do comportamento do sistema perante uma interacção. Os casos indicados permitem identificar quem são os utilizadores do sistema, quais os seus objectivos e as acções que conseguem realizar.

O administrador terá acesso a todas as funcionalidades existentes no sistema, nomeadamente gerir os utilizadores e gerir os eventos da plataforma. Na Figura 3.1 são ilustradas as funcionalidades atribuídas ao papel de Administrador do evento. O gestor de evento terá acesso apenas a funcionalidades ligadas ao evento. Para a gestão do evento este pode criar, editar, eliminar um evento; Criar ou editar um formulario de inscrição; É ainda da sua responsabilidade gerir candidaturas e gerir os modelos de email. Na Figura 3.2 são apresentadas as funcionalidades atribuídas ao papel de Gestor do evento. O orador apenas poderá interagir com a plataforma para editar as actividades a que está associado. Na Figura 3.3 ilustra-se as funcionalidades atribuídas ao papel de Orador. O participante interage com a plataforma para preencher a sua candidatura e confirmar o pagamento. Na Figura 3.4 ilustra-se as funcionalidades atribuídas ao papel de Participante do evento.

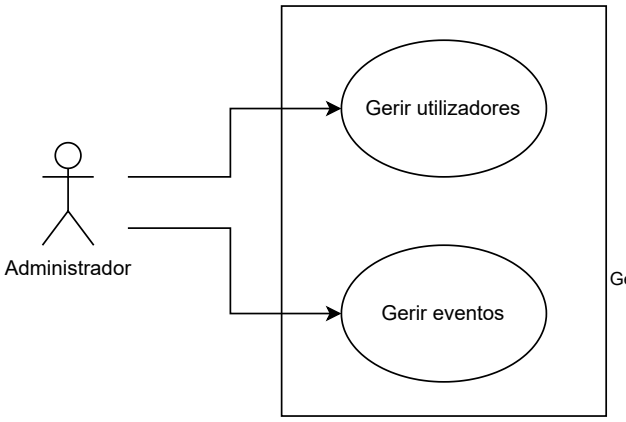

Figura 3.1: Caso de uso — Administrador.

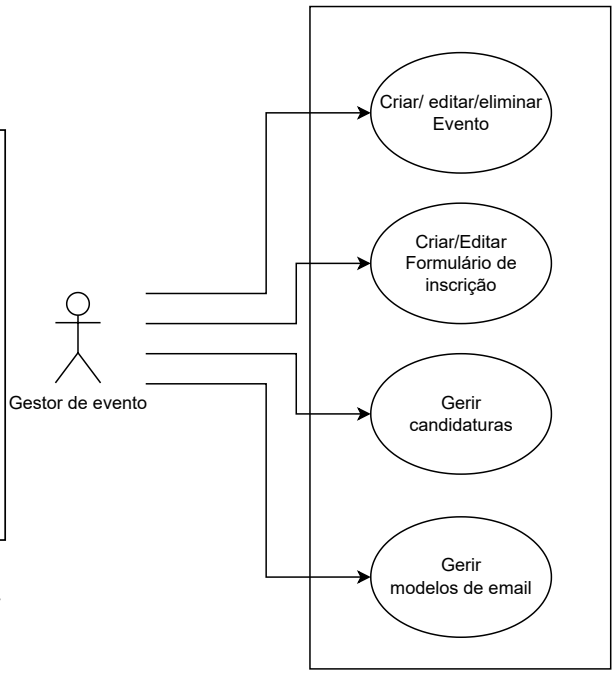

Figura 3.2: Caso de uso — Gestor.

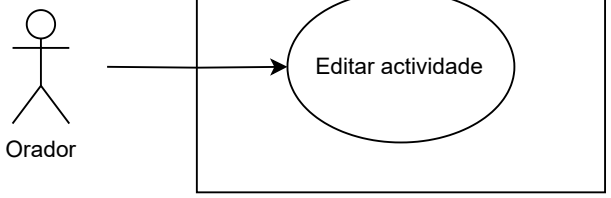

Figura 3.3: Caso de uso — Orador.

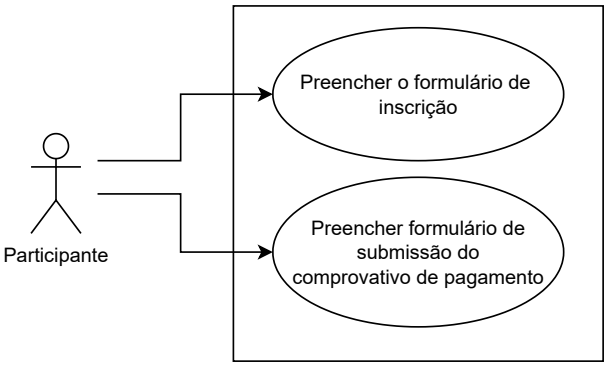

Figura 3.4: Caso de uso — Participante.

4

Conceitos e tecnologias

Este capítulo apresenta um conjunto de conceitos e tecnologias que irão ter um papel no desenvolvimento da plataforma que se pretende construir.

4.1 Business Process Management

Um dos aspectos centrais de um negócio, seja esse a gestão de eventos ou outro qualquer, independentemente da sua complexidade, ou das ferramentas que necessitem, são os seus processos de trabalho. Um processo de negócio é composto por uma colecção de sub processos ou de actividades que produzem um serviço ou produto específico.

A modelação destes processos de negócio é importante, pois produz um meio de distribuição e análise, que consequentemente possibilita a sua identificação e resolução de problemas, assim como a sua optimização. Para os definir, o *Business Process Management* (BPM) propõe um modelo de formalização representado como um conjunto coordenado de sub processos/actividades paralelas e/ou sequenciais, que estão conectadas para atingir um objectivo comum.

O BPM serve de apoio à gestão, análise e desenho de processos de negócio. Este pode ser considerado uma extensão dos sistemas orientados ao projecto. Estes definem métodos para automatizar um processo, no todo ou em parte, durante o qual documentos, informações ou tarefas são passados de um participante para outro através de uma acção, de acordo com um conjunto de procedimentos.

A utilização de um BPM é vantajoso para um projecto, visto que este e os seus respectivos processos são delineados de uma forma mais padronizada e detalhada, permitindo uma análise e melhoria contínua.

No conjunto de ferramentas que o BPM dispõe, surge o *Business Process Model and Notation* (BPMN) [23] como uma ferramenta ideal para colmatar a necessidade descrita anteriormente. Desenvolvido por S.Williams, este define um conjunto de regras para uma representação gráfica de um processo de negócio, fornecendo uma notação capaz de representar processos de qualquer complexidade, que seja intuitiva para qualquer colaborador, independentemente da sua capacidade técnica.

4.1.1 *Business Process Modeling Notation*

O *Business Process Model and Notation* (BPMN) [23] é um padrão para modelar processos de negócio que fornece uma notação gráfica com base numa técnica de fluxograma semelhante aos diagramas de actividade da *Unified Modeling Language* (UML) [24]. O propósito deste padrão de modelação é apoiar a gestão de processos de negócio, tanto para utilizadores técnicos quanto para utilizadores de negócio, fornecendo uma notação que seja intuitiva, mas capaz de representar semanticamente processos complexos. Este pode ser utilizado para apoiar o(s) objectivo(s) de todas as entidades envolvidas no projecto, adoptando uma linguagem comum para descrever processos, mitigando lacunas de comunicação que podem surgir entre a concepção e a implementação de processos de negócio. Em suma, usando esta notação, é possível descrever um processo de negócio completo. Para tal, a notação fornece diversos símbolos, que permitem que o processo seja representado graficamente. A especificação BPMN apresenta vários símbolos base divididos em seis categorias: objectos de fluxo, objectos de conecção, dados, pistas, artefactos [25]. Estes elementos base estão representados na Figura 4.1.

Objetos de fluxo são os principais elementos gráficos para definir o comportamento de um processo de negócio: eventos, atividades e gateway. Para ligar os objectos de fluxo usa-se os objectos de conecção compostos por fluxos de sequência, fluxos de mensagens e associações. Para a representação de dados pode se utilizar o elemento objeto de dados ou variações do mesmo. Por forma a agrupar os componentes anteriores existem dois elementos a piscina e a pista. Caso seja necessário adicionar mais informação sobre o processo pode se usar artefactos compostos por elementos de grupo ou anotações de texto. Alguns destes símbolos possui variações dos mesmos. Os símbolos de evento, tarefa e de controlo de fluxo possuem variações que são apresentadas nas figuras 4.3, 4.2 e 4.4.

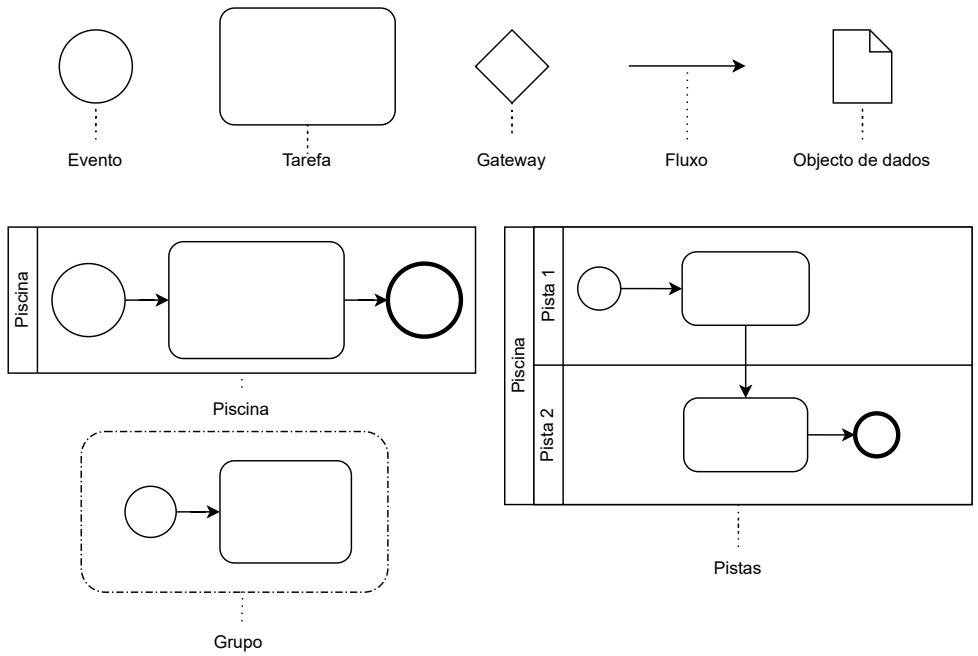

Figura 4.1: BPMN notação — Geral.

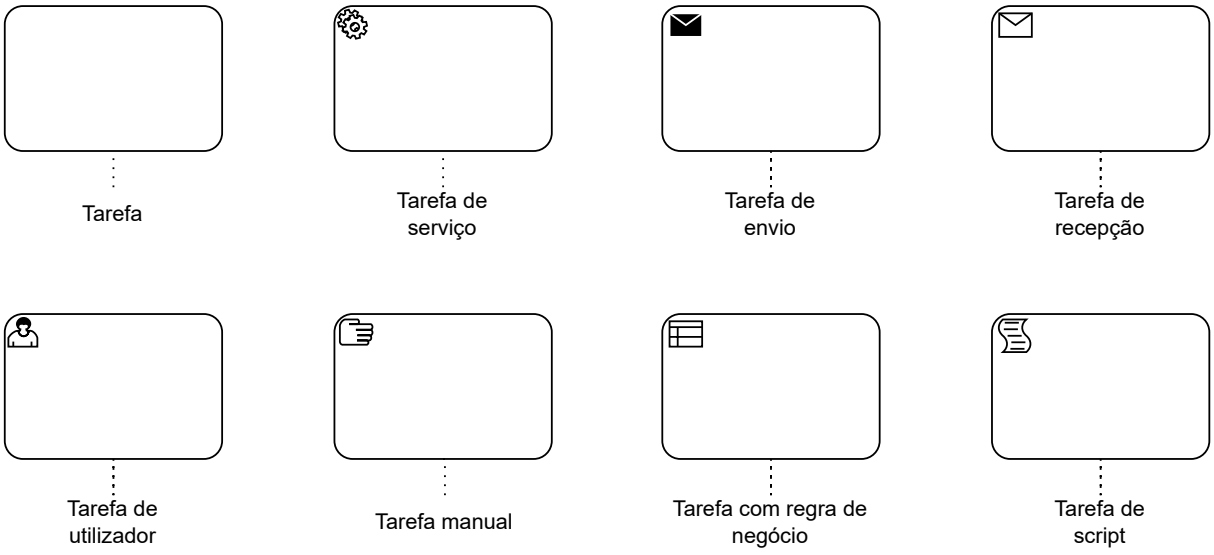

Figura 4.2: BPMN notação — Tarefas.

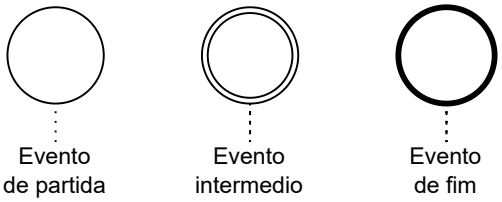

Figura 4.3: BPMN notação — Eventos.

Figura 4.4: BPMN notação — Controlo de fluxo.

4.2 Frameworks

Para o desenvolvimento de uma aplicação, foi necessário avaliar quais as ferramentas oferecidas pelo mercado e como estas, em conjunto com os conhecimentos adquiridos no mestrado, podem ser utilizadas para obter o melhor resultado. Numa visão de mais alto nível a aplicação pode ser dividida em duas camadas: a camada de serviços e a camada de apresentação. Para cada uma destas camadas, existem várias opções de desenvolvimento, pelo que nesta secção serão analisadas algumas dessas opções apresentando as vantagens e desvantagens identificadas em cada uma.

4.2.1 Camada de serviços

Na camada de serviços, pretende-se disponibilizar acesso a dados e isolar processos de negócio da camada aplicacional, através do uso de *Application Programming Interfaces* (API). Para tal existem duas abordagens possíveis: utilizar uma abordagem funcional, tipicamente utilizada em linguagens como *JavaScript*, ou então orientada a objectos, recorrendo a linguagens como Java ou C#. Independentemente da escolha, existem *frameworks* — conjuntos de abstracções que facilitam tarefas comuns de desenvolvimento de *software*, como autenticação, roteamento de pedidos, *middlewares* — que podem ser utilizadas para acelerar a concretização de projectos.

No caso da linguagem *JavaScript*, uma alternativa possível e de larga adopção pelo mercado é o Node.js. O Node.js é um ambiente de execução que permite a criação de aplicações escaláveis e de alto desempenho em *JavaScript*, tanto ao nível do servidor quanto do cliente. Este permite que uma equipa de desenvolvimento utilize a mesma linguagem de programação a todos os níveis de projecto, o que contribui para uma redução da complexidade do mesmo (comparativamente à escolha de tecnologias distintas para cliente e servidor), possibilitando uma redução do esforço e valências necessárias. O Node.js foi projetado para ser escalável, o que significa que as APIs construídas

podem facilmente lidar com grandes quantidades de tráfego e pedidos simultâneos. Esta plataforma possui também uma comunidade activa que cria várias bibliotecas em código aberto e promove o desenvolvimento. Esta abertura pode, contudo, ter os seus aspectos negativos tais como a possível falta de compatibilidade entre bibliotecas de terceiros e a plataforma ou necessidade de algum tipo de adaptação, aumentando a complexidade do desenvolvimento [26]. Um exemplo dessas bibliotecas abertas é a *framework* Express.js, baseada em Node.js, possui uma estrutura clara e organizada para a criação de rotas, *middlewares* e gestão de erros, no entanto, esta *framework* não inclui recursos como autenticação ou validações de entrada o que implica a utilização de outras bibliotecas de terceiros, já descrito como uma possível uma desvantagem [27] [28].

Do outro lado do espectro temos o Spring Boot [29] [30], uma *framework* de desenvolvimento de aplicações Java que permite o desenvolvimento de APIs RESTful. Esta *framework* suporta o uso de todas as linguagens JVM (Java, Kotlin e Goovy) e possui configurações pré-preparadas para usos comuns, e gestão automatizada das várias versões de bibliotecas e dependências necessárias, o que acelera o processo de desenvolvimento. O Spring Boot inclui por omissão uma estrutura integrada de segurança contra ameaças mais comuns, tais como injeção de SQL e *cross-site scripting* (XSS). Também possui uma documentação abrangente e regularmente atualizada que ajuda a compreender e usar diferentes recursos da *framework* com facilidade.

Para desenvolver esta camada foi escolhida a *framework* Spring Boot.

4.2.2 Camada de apresentação

Actualmente existem inúmeras *frameworks* para o desenvolvimento de interfaces de utilizadores, que aplicam diversos paradigmas de organização de código e estratégias de apresentação de conteúdo que foram sendo desenvolvidas nos últimos anos, resultantes da experiência acumulada do mercado, resultante da larga procura por *software* nas últimas décadas. Tendo em conta a vastidão de *frameworks* que existem no mercado e dados os requisitos do projecto que serão apresentados em secções futuras, esta subseção vai se focar apenas em *frameworks* para o desenvolvimento de aplicações *web*, mais concretamente duas das *frameworks JavaScript* mais conhecidas neste âmbito são o React e o Angular.

A *framework* React [31] teve a sua origem num projecto interno do Facebook e foi transformado em *open source* e em Maio de 2013. É uma biblioteca *JavaScript* para construir interfaces de utilizador, sendo a sua estratégia principal a definição de interfaces de utilizador sob a forma de vários fragmentos de código independentes segundo padrões

pré-estabelecidos como um meio de reduzir a complexidade das mesmas. Possui uma lógica de código simples e pode resolver facilmente problemas de compatibilidade entre navegadores. React permite que a interface do utilizador seja combinada diretamente com vários componentes, resultando em páginas *web* interativas. Introduz também a sintaxe JSX [32, 33], o que torna a reutilização de componentes mais simples e garante que a estrutura interna dos mesmos seja clara.

A *framework* Angular [34] foi lançada em setembro de 2016 e teve origem como uma nova e melhorada versão da *framework* AngularJS, criada em 2009 pela Google. O Angular é usado principalmente para criar páginas dinâmicas, com mais foco na construção de aplicações CRUD (Create, Retrieve, Update, Delete). A *framework* caracteriza-se pela modularidade, ligação de dados bidirecional automatizada, marcação semântica e injeção de dependências. [35].

Para desenvolver esta camada foi escolhida a *framework* React.

4.3 *Continuous Integration/Continuous Deployment* (CI/CD)

Integração contínua e entrega contínua são práticas comuns no desenvolvimento de *software* que permitem melhorar a qualidade e uma maior eficiência na entrega. Na integração contínua, pressupõe-se a utilização de um sistema de controlo de versões que, com cada alteração, deve acionar um processo automático que consiste numa preparação do código para a entrega final, execução de quaisquer verificações de qualidade e/ou desempenho e produção dos artefactos necessários. Este processo deve ser, idealmente, determinista, na medida em que deve produzir o mesmo resultado com a mesma base de código.

Este processo de integração contínua de código, aliado a um processo de teste automatizado que permita identificar possíveis regressões ou casos de utilização não previstos, tem como consequência a possibilidade de gerar novas versões do *software* que está a ser desenvolvido, com os incrementos de funcionalidade resultantes da última iteração sobre o mesmo. Este lançamento constante de novas versões (leia-se, entrega contínua), permite que os stakeholders do projecto possam observar e opinar acerca das funcionalidades assim que prontas, e tal como os testes automatizados, identificar problemas com antecedência, reduzindo o risco de não cumprimento dos requisitos do projecto.

A entrega contínua é então a consequência da integração contínua: dado um sistema

que consegue integrar desenvolvimentos de um projecto ao longo do tempo (idealmente de forma automatizada) e produzir os artefactos necessários para a sua execução, torna-se então possível também a implantação dessas novas versões que incluem as mais recentes capacidades. Possibilita também a opção de automatizar o processo de entrega para garantir que as alterações ao código base possam ser colocadas em operação de uma forma mais expedita. Isto pressupõe que existe sempre uma versão estável do código, algo que pode ser garantido no processo de integração contínua. O uso desta prática permite a deteção e resolução de erros antecipadamente.

Existem várias plataformas de sistemas de controlo de versões que oferecem versões integradas de CI/CD, nomeadamente o *Bitbucket* [36], *GitHub* [37] ou *GitLab* [38].

4.4 Ambiente de execução

Durante o processo de desenvolvimento de uma aplicação, seria ideal que o ambiente de produção e o ambiente das máquinas de desenvolvimento tivessem exactamente o mesmo sistema operativo e as mesmas versões das bibliotecas internas por forma a evitar discrepâncias que possam trazer inconsistências no desenvolvimento. Esta forma de trabalhar pode ser complicada de atingir e gerir visto que a topologia da rede ou as políticas de segurança podem ser diferentes entre os vários ambientes. Uma solução para estes problemas poderia ser o uso de máquinas virtuais, porém a utilização das mesmas num cenário real pode trazer várias dificuldades adicionais, nomeadamente na carga computacional necessária executar um segundo sistema operativo, no licenciamento desses mesmos sistemas operativos, e na manutenção geral de estado de todas essas máquinas virtuais — podem começar todas no mesmo estado, mas ao longo do tempo irão haver desvios que vao necessitar de ser identificados e corrigidos. A necessidade de uma ferramenta que "funcionasse como uma máquina virtual, mas mais leve e simples de gerir", levou ao desenvolvimento de sistemas de contentores (*containers*). Um contentor contem a aplicação com as respetivas dependências, bibliotecas, binários e ficheiros de configuração, necessárias para a sua execução. A utilização destes contentores permitem um isolamento entre a aplicação e a infraestrutura do sistema "hospedeiro", mas também isolamento entre outros contentores alojados no mesmo sistema. Com esta camada de isolamento, a aplicação torna-se agnóstica relativamente ao sistema hospedeiro, desde que o hospedeiro suporte a sistemas de gestão de contentores. A utilização desta solução pode ser mais económica ao diminuir os custos operacionais associados ao desenvolvimento de *software*.

4.4.1 Docker

No mercado existem várias plataformas que oferecem este tipo de solução, sendo o Docker uma das mais utilizadas. Esta é uma plataforma de conteinerização que fornece várias ferramentas que simplificam o processo de gestão e execução de contentores [39]. Neste sistema, cada contentor executa uma imagem, definida por um ficheiro de texto denominado Dockerfile. Este contém instruções que especificam a imagem base, configurações do ambiente, instalação de dependências ou cópia de ficheiros para a imagem e define os comandos a serem executados quando o contentor é iniciado, permitindo assim execuções de aplicações inerentes consistentes e reproduzíveis. Estas servem de modelo para os contentores, permitindo várias instanciações com base na mesma imagem.

Um contentor Docker constitui um ambiente de execução isolado que opera sobre o sistema operacional do hospedeiro. Os contentores partilham o kernel do sistema operativo hospedeiro, o que os torna leves e eficientes em termos de recursos. Cada contentor tem seu próprio sistema de ficheiros, processos e rede, garantindo que as aplicações executadas dentro destes estejam isoladas uns dos outros e do sistema hospedeiro.

O Docker Registry é um sistema de armazenamento e distribuição de imagens Docker. É possível configurar registos privados ou usar o Docker Hub. Docker Hub é um repositório público de imagens que sendo um registo central para imagens do Docker onde se pode colocar imagens próprias e aceder a imagens desenvolvidas por terceiros.

Em sistemas mais complexos, pode existir a necessidade de executar múltiplas imagens distintas. Como já referido, o sistema Docker permite a execução de vários contentores em simultâneo, sejam baseados na mesma imagem ou distintas, contudo, pode se tornar trabalhoso ou difícil de gerir todas as diferentes instâncias. Para lidar com isso, foi desenvolvido o Docker Compose, uma ferramenta para definir e gerir aplicações com vários contentores. Este define uma configuração de vários contentores usando um ficheiro YAML e executar todos com um único comando. Isto simplifica o processo de gestão de aplicações com vários serviços interligados.

4.5 Gestão de dados

Armazenamento e gestão de dados são necessidades comuns de uma aplicação que tenha de manter qualquer tipo de estado a longo prazo. Existem várias estratégias como, por exemplo, utilizar uma base de dados relacional ou uma não relacional. São duas

categorias de sistemas de gestão de base de dados baseadas em paradigmas diferentes de modelação e armazenamento de dados e que atendem a propósitos diferentes.

Base de dados relacionais [40] baseiam-se num esquema rígido composto por tabelas e as relações entre estas. Este esquema é definido antes da inserção de dados e a sua alteração pode ser um processo moroso, pois requer um planeamento cuidado e possivelmente uma migração de dados. O armazenamento de dados é feito de uma forma estruturada em tabelas com esquemas predefinidos em que cada coluna corresponde a um atributo/característica e cada linha representa um registo nessa tabela. As relações entre as tabelas são estabelecidas por meio de chaves primárias e estrangeiras, permitindo a integridade e reforçando as restrições de dados. A manipulação de dados deve seguir as propriedades ACID (Atomicidade, Consistência, Isolamento, Durabilidade).

Base de dados não relacionais [40] baseiam-se num esquema flexível utilizado vários modelos de dados como baseado em documento, valor-chave, família de colunas ou grafo. Estas são projectadas para armazenar e recuperar grandes volumes de dados não estruturados ou semi-estruturados, permitindo também adicionar ou modificar campos sem afetar os registos existentes. A abordagem sem esquema ou esquema na leitura torna as base de dados NoSQL adequadas para lidar com dados que mudam ou evoluem rapidamente. Contrariamente a base de dados relacionais que utilizam SQL (Structured Query Language) como linguagem de definição de tabelas e manipulação de dados, as bases de dados NoSQL têm suas próprias linguagens de consulta, utilizam APIs ou ainda consultas baseadas em JSON, dependendo do modelo de dados que suportem.

5

Solução Proposta

Tal como referido no capítulo 2, actualmente não existe no mercado uma plataforma que corresponda às necessidades/requisitos apresentados. Tendo em conta que é necessário desenvolver uma aplicação que corresponda a essas necessidades é preciso definir os processos de negócio assim como a arquitetura lógica e física que devem ser seguidas.

5.1 Processos de negócio

Seria irrealista assumir que simplesmente introduzir uma aplicação no fluxo de trabalho da equipa irá resolver todas as dificuldades que esta sente. É também indispensável uma optimização das metodologias de organização de processos de trabalho da equipa, mais especificamente na forma como são estruturados, mantidos e distribuídos. Uma solução possível seria a utilização de uma solução de BPM [41], termo utilizado para descrever um método de sistematizar os processos de trabalho de um determinado negócio, com o objectivo de maximizar a capacidade de análise, automação e partilha, resultando numa optimização dos mesmos.

Antes de iniciar o desenvolvimento, foi efetuado o levantamento dos processos de negócio por forma a garantir que os requisitos levantados anteriormente estão alinhados com os mesmos. Para tal, recorreu-se ao uso do BPMN.

Foram definidos dois processos: o processo de candidatura e o processo de criação

de evento. Como os processos são bastante extensos, foram colocados excertos dos mesmos nesta secção e a versão completa está disponível nos Anexos (A.1 e A.2).

No fluxo de candidatura ao evento, existe a intervenção de três utilizadores: o candidato, o gestor de eventos e a tesouraria, no entanto, é apenas visível na Figura 5.1 a perspetiva do candidato. Dada a dimensão do diagrama foi subdividido em três fases: submissão, estados intermédios e conclusão.

Na Figura 5.1 é apresentada a primeira fase da candidatura de um participante, que corresponde ao preenchimento e submissão de um formulário de inscrição.

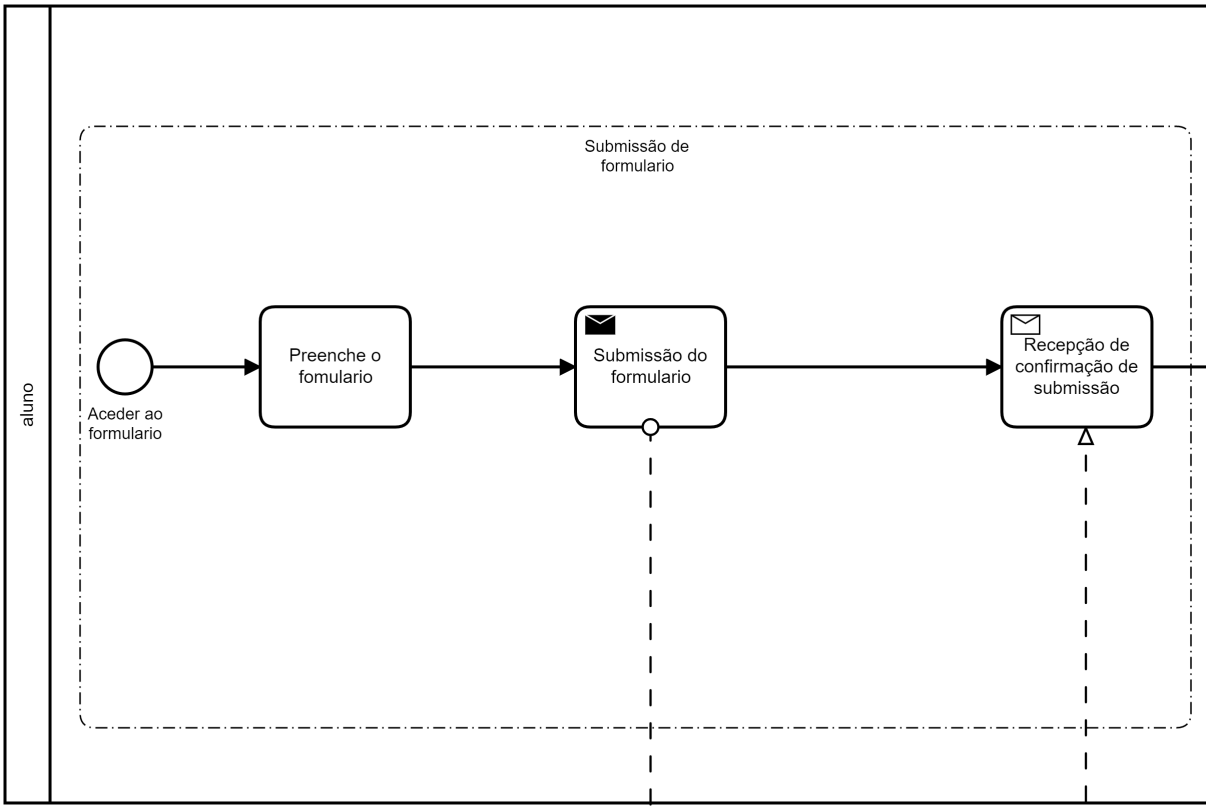

Figura 5.1: Processo de candidatura da perspetiva do candidato — Submissão.

Na Figura 5.2, é exibido o processo intermédio da candidatura, onde se corrige algum campo do formulário de inscrição e é indicado ao utilizador se a candidatura é aceite ou rejeitada.

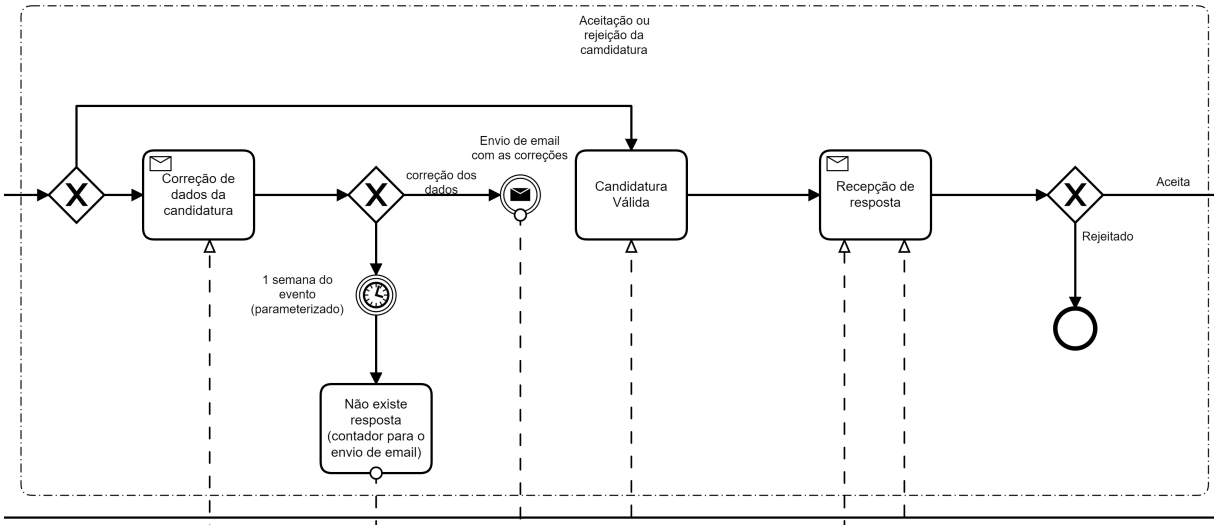

Figura 5.2: Processo de candidatura da perspetiva do candidato — Estados intermédios.

Por fim, na Figura 5.3 é mostrado o processo de conclusão da candidatura através do pagamento ou por desistência.

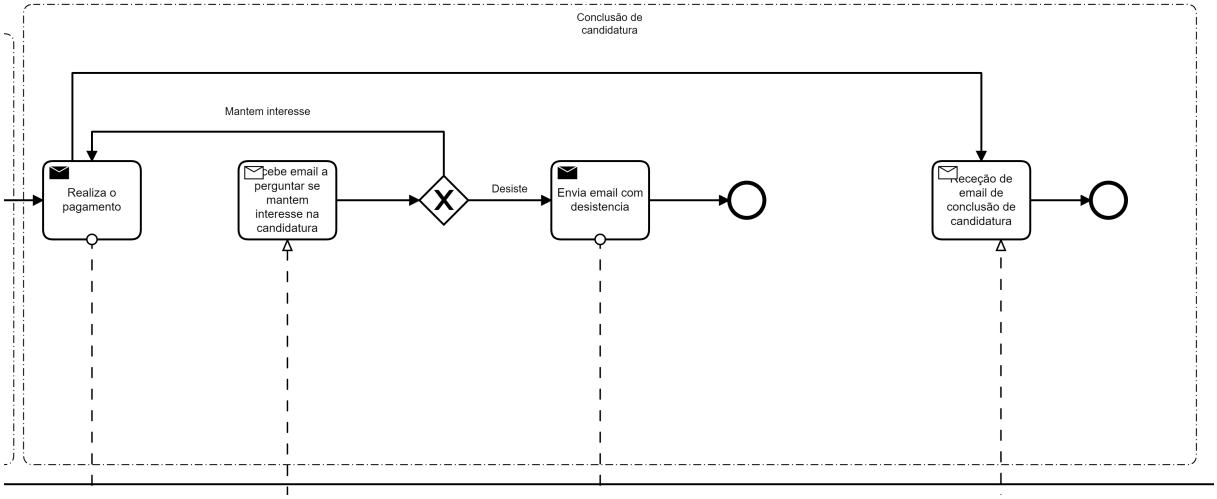

Figura 5.3: Processo de candidatura da perspetiva do candidato — Conclusão.

Após a definição do processo da candidatura foi definido o processo de gestão do evento apresentado na Figura 5.4. Neste intervêm três utilizadores: o administrador, o gestor de eventos e o monitor. A Figura 5.4 descreve apenas a perspetiva do gestor de eventos.

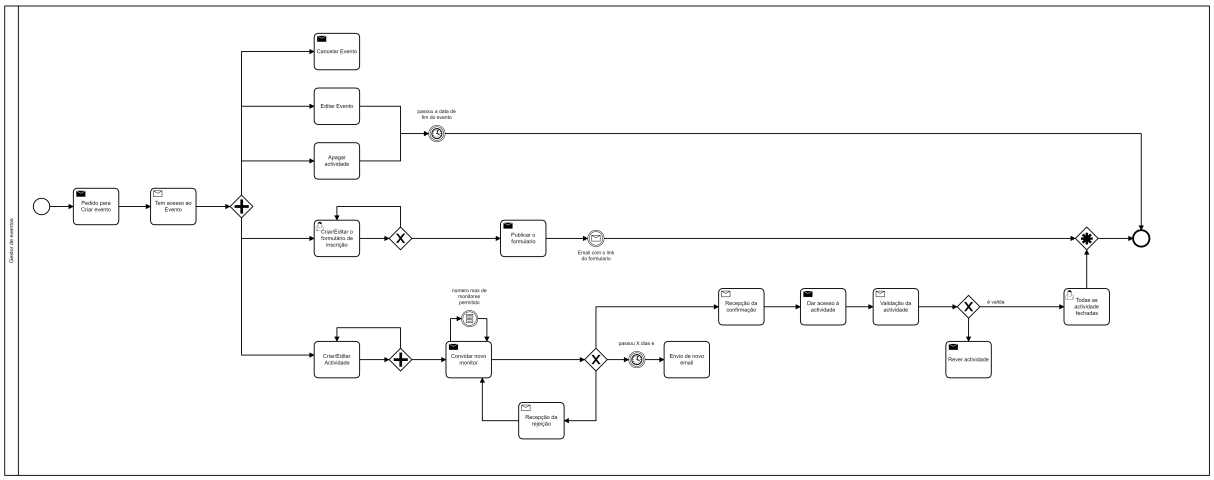

Figura 5.4: Processo de criação de um evento da perspetiva do gestor de eventos.

5.2 Arquitetura lógica

A arquitetura lógica define uma representação conceptual da estrutura e funcionalidade do sistema, independentemente do equipamento subjacente ou dos detalhes de implementação. Define os diferentes módulos, serviços e fluxos de dados no sistema e como estes interagem entre si para atingir os objetivos de negócio pretendidos e por forma a garantir a modularidade, fácil manutenção, flexibilidade e escalabilidade do sistema.

No desenho da arquitetura lógica, considerou-se adequado um sistema de camadas para satisfazer os requisitos propostos. Assim, a arquitetura lógica de mais alto nível do projeto foi definida por três camadas: a camada de apresentação, a camada de serviços e a camada de dados, conforme se mostra na Figura 5.5.

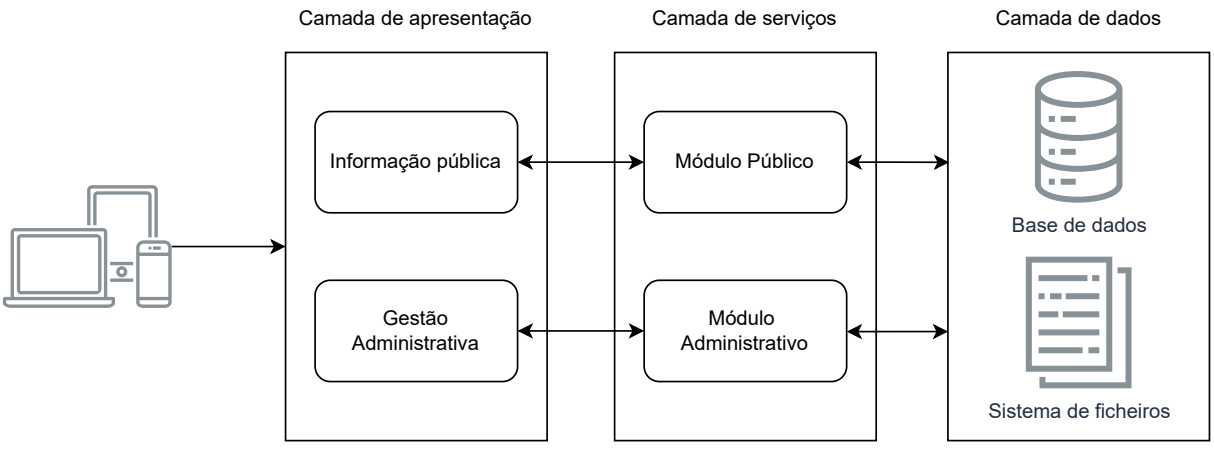

Figura 5.5: Arquitetura lógica.

A camada de apresentação é dividida em dois sub-módulos: um de acesso público e um de acesso reservado, destinado à gestão. O módulo público tem como função a apresentação dos eventos disponíveis, para apoiar a sua respectiva divulgação. O módulo de gestão serve para gerir não só a informação pública como também gerir os eventos e as candidaturas aos mesmos, e fornecer o formulário para a respetiva inscrição. Para compreender melhor esta camada foram desenvolvidos protótipos da interface de utilização que se encontram no Anexo B.

A camada de serviços é responsável por toda a lógica de negócio e por disponibilizar dados à camada de apresentação. Esta camada comunica diretamente com a camada de dados. A arquitetura interna desta camada baseia-se numa arquitetura orientada ao serviço, ou seja, cada serviço é uma unidade independente com um determinado conjunto de funcionalidades, desenhado para efetuar determinadas tarefas dentro de um âmbito comum. Esta abordagem traz benefícios comparativamente a uma abordagem monolítica para toda a aplicação, nomeadamente: desenvolvimento mais eficiente, maior fiabilidade e maior facilidade na depuração de erros. Permite também uma fácil manutenção, pois os serviços podem ser modificados e atualizados conforme necessário sem afetar outros serviços. A escalabilidade é outra das grandes vantagens, sendo possível redimensionar a disponibilidade os serviços de forma independente. Esta camada é dividida em dois serviços: um público que disponibiliza toda a informação relativa à divulgação dos eventos e um privado que disponibiliza e gere toda a informação relativa à gestão dos eventos assim como as candidaturas aos mesmos. Com esta divisão existe uma maior flexibilidade na gestão dos serviços como facilita a escalabilidade dos mesmos.

A camada de dados é responsável pela organização e armazenamento dos dados no sistema, definindo a estrutura e a relação dos dados no sistema. Esta camada encontra-se dividida em duas partes: uma responsável pelo modelo de dados e outra dedicada ao armazenamento de ficheiros. O modelo de dados deverá suportar os requisitos anteriormente definidos.

5.3 Arquitetura física

A arquitetura física refere-se aos aspectos tangíveis e concretos de um sistema, com foco nos equipamentos, infraestrutura de rede e componentes físicos envolvidos na execução do sistema. Deve ter em conta fatores como confiabilidade, escalabilidade, desempenho são considerações essenciais na arquitetura física.

Tal como referido anteriormente, uma das grandes vantagens do Docker é a possibilidade de poder executar um contentor independentemente do sistema operativo disponibilizado pela máquina hospedeira, podendo também correr múltiplos contentores nessa mesma máquina. Desta forma, a arquitetura física será composta por uma máquina (virtual ou física) que execute uma instância de Docker em que neste existem contentores para cada uma das partes do sistema, no mínimo um contentor para cada camada.

6

Implementação

Esta secção foca-se na implementação da solução proposta, concretizando a arquitetura lógica e física, definidas nas secções anteriores. O diagrama da arquitetura lógica pode ser actualizado para conter alguns dos detalhes da implementação, como ilustrado na Figura 6.1, concretizando a camada de apresentação como um *website* com páginas públicas e páginas admistrativas. A camada de serviços disponibiliza duas APIs que servem as respectivas páginas da camada de apresentação. A camada de dados é composta por uma base de dados e um sistema de ficheiros acedido por *File Transfer Protocol* (FTP). As subsecções seguintes apresentam uma visão mais detalhada de cada uma destas camadas. O código da implementação está disponível no seguinte link https://bitbucket.org/smccms/pgpe/src/master/ .

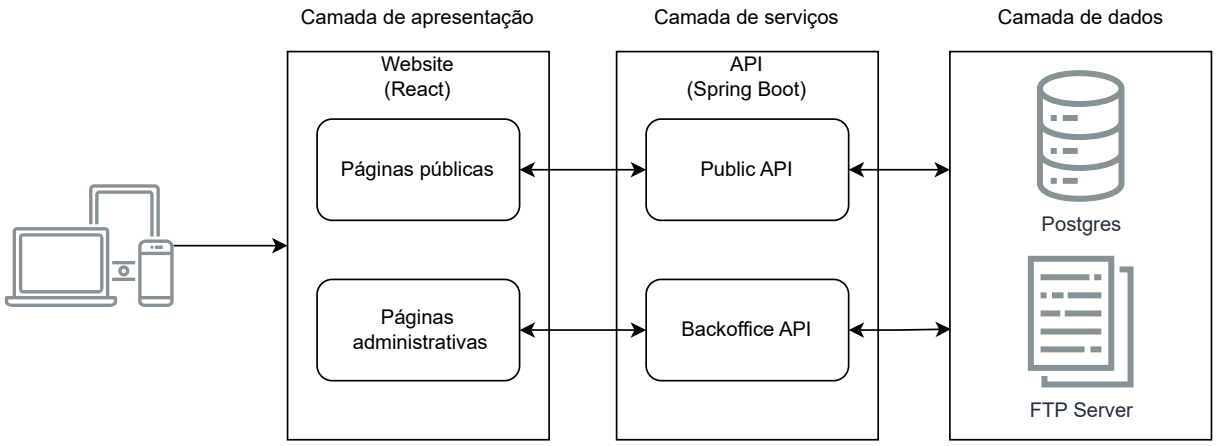

Figura 6.1: Arquitetura de implementação.

6.1 Camada de dados

Tal como definido na arquitetura lógica, a camada de dados contém dois sub-módulos: uma base de dados, que implementa o modelo de dados definido na Figura 6.2, e um servidor de FTP responsável pelo armazenamento de ficheiros. O modelo entidade associação com todas as propriedades encontra-se no Anexo D.

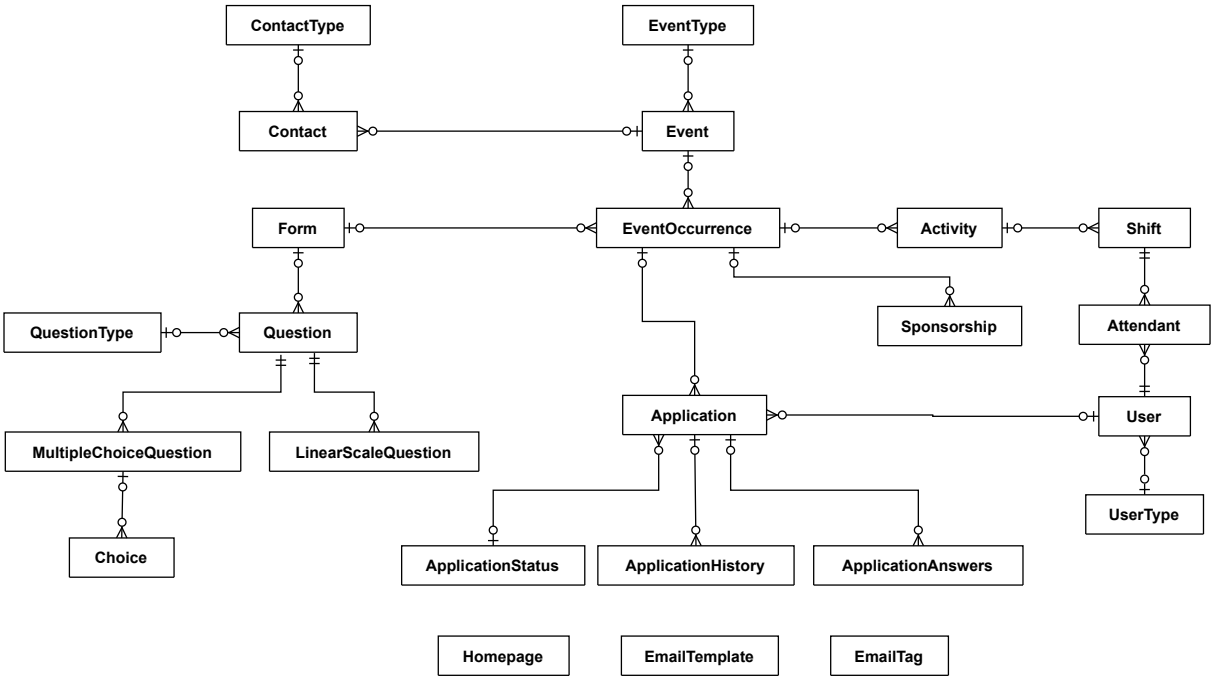

Figura 6.2: Modelo Entidade Associação.

No modelo entidade associação podemos ver várias entidades que podem ser agrupadas pelo seu contexto. No contexto de evento temos as entidades EventType, Event e EventOccurrence, a primeira define tipos de eventos, a segunda o evento e a última define a edição de um evento. No contexto de um formulário existem as seguintes entidades: Form define um formulário, Question define uma pergunta, QuestionType define o tipo de pergunta, LinearScaleQuestion define uma pergunta do tipo linear, MultipleChoiceQuestion define uma pergunta do tipo escolha múltipla e por fim Choice que define uma opção na escolha múltipla. No contexto de uma candidatura existem as seguintes entidades: Application define uma candidatura, ApplicationStatus define o estado de uma candidatura, ApplicationAnswers define as respostas da candidatura, ApplicationHistory define o historio da candidatura.

Para a implementação do modelo de dados recorreu-se à aplicação PostgreSQL [42], que oferece um sistema *open source* de base de dados relacional com uso da linguagem SQL, compatível com os principais sistemas operativos.

Por forma a melhorar o acesso a dados, foram criadas várias SQL *views* que permite encapsular a complexidade de ligações de algumas tabelas. Isto permite apenas um acesso a uma única *view* em vez do acesso a múltiplas tabelas, proporcionando à camada superior uma forma optimizada de aceder aos dados que irá precisar com regularidade. A seguinte lista apresenta os nomes das views definidas e a informação que disponibilizam:

- vw_applicationList — a lista de uma candidatura para os eventos
- vw_applicationForm — a lista de perguntas de uma candidatura
- vw_eventFormQuestion — a lista de perguntas de um formulário para um evento
- vw_eventFormSimpleQuestion — a lista de perguntas de um formulário para um evento
- vw_eventFormMultipleQuestion — a lista de perguntas de um formulário para um evento
- vw_eventFormLinearQuestion — a lista de perguntas de um formulário para um evento
- vw_formQuestion — a lista de perguntas de um formulário
- vw_formSimpleQuestion — a lista de perguntas simples de um formulário
- vw_formMultipleQuestion — a lista de perguntas de escolha múltipla de um formulário
- vw_formLinearQuestion — a lista de perguntas, lineares de um formulário

6.2 Camada de serviços

Tal como descrito na arquitetura lógica, a camada de serviços é composta por dois módulos: uma API pública e um API de gestão. Esta divisão deve se ao facto de cada API ceder dados a diferentes níveis de acesso à camada de apresentação permitindo uma gestão simplificada das diferentes responsabilidades.

Na implementação da camada de serviços, foi utilizada a *framework* Spring Boot. Tal como definido na arquitetura lógica foram criados dois projectos: um para servir as páginas públicas (publicAPI) e outro para servir as páginas de gestão(backofficeAPI).

6.2.1 APIs

Para a construção desta camada, foi necessária a instalação da linguagem Java e do gestor de pacotes Maven na máquina de desenvolvimento. Para se iniciar um projecto em Spring Boot é possível utilizar uma ferramenta online Spring Inicilizr [43] que permite gerar uma configuração inicial do projecto. Nesta ferramenta é possível escolher o tipo de gestão de dependências de projecto, neste caso Maven, a linguagem JVM pretendida, no caso Java e a versão da framework. É também possível configurar os metadados e as dependências do projecto. Com isto, é disponibilizado um zip com a base do projecto.

Para a implementação do projecto tirou-se partido das anotações de classe disponibilizadas pelo Spring Boot para registar os componentes no contexto da aplicação, nomeadamente:

- `@SpringBootApplication` — define a classe principal da aplicação.
- `@Component` — notação genérica que indica que a classe é um componente que será detetado automaticamente pelo spring. As anotações `@Repository`, `@Service`, `@Configuration` e `@Controller` são meta-anotações de `@Component`.
- `@Controller` — indica que a classe é controlador MVC, possui uma versão especializada `@RestController`.
- `@Service` — indica que o componente pertence a camada de serviços
- `@Repository` — indica que o componente pertence à camada de dados e que actua como um repositório da babe de dados
- `@Configuration` — indica que a classe contém a definição de um @Bean

A estrutura de classes do projecto foi definida a volta destas anotações, sendo todos os controladores anotados com `@RestController`, os serviços com `@Service` e repositórios com `@Repository`.

A arquitetura das classes presente na Figura 6.3, apoiada nestas anotações, foi então definida segundo o seguinte padrão: os controladores definem os endpoints relativos ao contexto dos mesmos, e estes por sua vez chamam um serviço que encapsula a lógica de negócio. O serviço faz uso dos vários repositórios para aceder e manipular os dados do sistema, recorrendo a tradutores que transformam objeto de transferência de dados (DTO) em entidades e vice-versa, para facilitar a comunicação entre as várias classes.

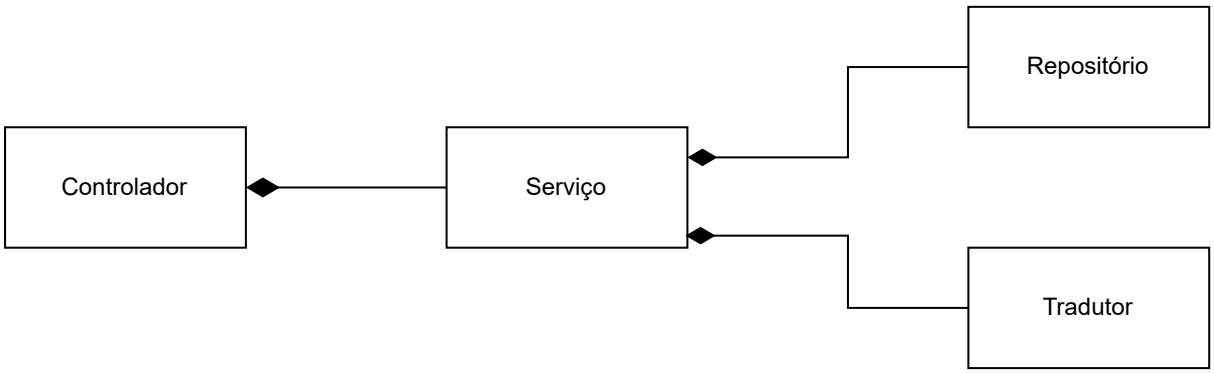

Figura 6.3: Arquitetura de classes.

Figura 6.4: Arquitetura de classes — Exemplo.

Para definir os dados aos quais os repositórios vão aceder foi necessário usar anotações nas respectivas classes, segundo o Spring Data JPA. Neste caso, as classes são um espelho das entidades presentes na base de dados, sendo identificadas com a anotação `@Entity`. A classe representa uma tabela e cada instância dessa classe representa uma linha na tabela de base de dados. Existem também outras anotações que permitem uma definição mais precisa do significado dos vários campos. A `@Id` permite indicar qual a propriedade da classe que corresponde à chave primária da

tabela. Caso a chave primaria seja um identificador gerado, deve usar se a anotação @GeneratedValue, podendo escolher uma das quatro formas de gerar o valor: `AUTO`, `TABLE`, `SEQUENCE` ou `IDENTITY`. Para o caso de não ser possível ou desejado espelhar a nomenclatura entre a base de dados e as respectivas classes/atributos é possível utilizar as anotações: `@Table` — para indicar o nome da tabela; `@Column` — para indicar o nome da coluna e, opcionalmente, outros detalhes como o tamanho, nullable e unique.

A API de gestão disponibiliza uma lista extensa de endpoints que se encontra no Anexo C. Numa visão de alto nível, existem os seguintes controladores que agrupam os endpoint em operações sobre o mesmo contexto:

- ApplicationController — possui operações sobre as candidaturas;
- EmailController — operações sobre os emails;
- EventController — operações sobre os eventos;
- EventOccurrenceController — operações sobre as edições de eventos;
- FormController — operações sobre os formulários;
- HomePageController — operações sobre a homepage;
- UserController — operações sobre os utilizadores;
- AuthController — operações de registo e autenticação de utilizadores.

Tendo em conta que a API de gestão permite o acesso e alteração de dados privados, existe a necessidade de garantir que o acesso aos mesmos apenas é concedido a utilizadores autorizados. Desta forma, a maioria dos endpoints exigem o envio de um *Authorization* token no HTTP *Header Authorization*. Para tal foi utilizado o JWT (Json web Token) [44]. Um JWT é uma representação compacta de uma série de *claims* — informações acerca de um determinado assunto, neste caso específico relativos à identificação do utilizador — juntamente com uma assinatura para validar sua autenticidade. Os endpoints /auth são os únicos cuja autorização não é necessária.

A API pública disponibiliza os endpoints apresentados na tabela 6.1.

Método HTTP	**Url**	**Descrição**
GET	`/event/{id}/regulation`	devolve o ficheiro que corresponde ao regulamento de um evento
GET	`/event/{id}`	devolve a informação de um evento e das suas edições
GET	`/eventoccurrence/{id}`	devolve a informação relativa a uma edição
GET	`/homepage`	devolve informações relativas à plataforma e uma lista dos eventos

Tabela 6.1: Endpoints disponibilizados na API Pública.

6.3 Camada de apresentação

Para a implementação da camada de apresentação foi utilizado a *framework* React. Para tal é necessário passar por um processo de instalação com os seguintes passos:

- instalar o NodeJS do site oficial consoante o tipo de sistema, tipicamente com este é também instalado o NPM
- instalar o React através do comando `npm install react`
- instalar o Mui material através
  `npm install @mui/material @emotion/react @emotion/styled`

A organização da diretoria `pages/` espelha os serviços disponíveis pela plataforma. Na diretoria `public/` encontram-se os componentes React que representam as páginas públicas da aplicação. Na diretoria `backoffice/` os componentes estão agrupados por diretorias consoante o contexto dos mesmos, por exemplo, a diretoria `applications/` contem os componentes necessários para efetuar as operações sobre as candidaturas, o mesmo aplica-se nas restantes diretorias dentro da diretoria `backoffice/`.

Na aplicação para se poder aceder as páginas de gestão é necessário fazer login por forma a obter um JWT Token irá servir para poder fazer os restantes pedidos da área de gestão.

6.4 Alojamento

Ao longo do processo de desenvolvimento foi-se tornando cada vez mais desafiante manter todos os componentes em execução em simultâneo num contexto de desenvolvimento, que envolvia várias aplicações em execução. Para facilitar o trabalho, tanto de desenvolvimento como o da disponibilização final do sistema, recorreu-se ao Docker para controlar a execução das várias partes do sistema.

Para executar o sistema utilizando o Docker, foi criado na directoria base do mesmo um ficheiro `Docker-compose.yml`. Neste ficheiro são definidos serviços, identificados por um nome definido pelo utilizador, que o Docker Engine irá utilizar para gerar contentores que irão ser executados segundo especificações de configuração definidas no mesmo ficheiro, entre as quais qual a imagem Docker a utilizar. Para este projecto, foram definidos cinco serviços. Dois serviços foram configurados recorrendo a imagens disponíveis no DockerHub [45]: db com a imagem `postgres:13.5` e o ftp-server com a imagem `stilliard/pure-ftpd`. Os restantes três serviços utilizam imagens construidas com base nos respectivos Dockerfiles: public-API, backoffice-API e frontend. Os dockerfiles da public-API e da backoffice-API são semelhantes. Ambos informam o Docker que:

- a construção da imagem tem como base a imagem `openjdk:16.0.1-jdk`
- define uma variável
- copiar o ficheiro `app.jar`
- corre o comando `java -jar /app.jar`

O dockerfile do frontend define duas fases, a primeira:

- a construção da imagem tem como base a imagem `node:16` com o nome builder
- define a diretoria de trabalho em `/app`
- copiar os ficheiros
- instala os node modules e dá build dos assets

a segunda:

- a construção da imagem tem como base a imagem `nginx:alpine`

- define a diretoria de trabalho em `/usr/share/nginx/html`
- remove os assets presentes nessa diretoria
- copiar o ficheiro `nginx.conf` para `/etc/nginx/nginx.conf`
- copiar os ficheiros da primeira fase para `/app/build`
- corre o nginx com as diretrizes globais e com o `deamon` desligado

Para garantir uma entrega contínua e tirando partido das ferramentas disponibilizadas pelo Bitbucket [46], o serviço utilizado para controlo de versões de git, foi montado um *pipeline* — uma ferramenta que permite definir *scripts* que podem executar tarefas automatizadas sobre código de um determinado repositório — que corre os testes unitários, faz *build* dos vários projectos e faz *deploy*. O *pipeline* pode ser executado manualmente via a interface *web* do Bitbucket ou então sempre que é criada uma tag com o nome `release` via git. O *pipeline* é composto por vários passos, dois dos quais são executados em paralelo. Primeiro são executados os testes unitários das APIs em paralelo. Depois é feito um *build* das APIs e do frontend, e o último passo é o carregamento dos artefactos gerados pelos builds para uma máquina remota.

Para instalar o ambiente de desenvolvimento foram seguidos os seguintes passos:

- Instalar o Docker Desktop para o sistema operativo utilizado: `https://docs.Docker.com/desktop/install/windows-install/`
- Aceder ao repositório do PGPE e fazer o clone para a máquina através do git;
- Para as APIs é necessário aceder as seguintes diretorias `src\backoffice-API` e `src\public-API` e correr os comandos `maven clean` e `maven package -dev` para gerar os executáveis `.jar`;
- Abrir uma linha de comandos na diretoria src do projeto e correr o seguinte comando: `Docker-compose -p pgpe up -d --build`

Com esta configuração, o projeto é executado na integra via Docker numa máquina qualquer e é possível observar a sua execução através do Docker Desktop.

Para montar no ambiente de produção devem ser seguidos os seguintes passos:i

- Ter o Docker instalado na máquina.
- Correr o seguinte comando: `sh server.sh <version>`

7

Avaliação

Ao longo do desenvolvimento deste projecto foi necessário efectuar testes de forma a validar o estado do mesmo face aos requisitos definidos. Neste sentido, foram feitos diversos tipos de testes aos artefactos produzidos, incluindo testes unitários, de integração e de usabilidade, e em diferentes momentos do projecto.

Os testes unitários são uma forma de validar partes individuais de um sistema de forma a mostrar que essa parte funciona conforme a especificação. Este tipo de teste permite identificar problemas mais cedo simplificando a integração, sendo feitos à medida que o projecto vai sendo desenvolvido. Para um projecto Spring Boot os testes devem estar presentes na pasta `src/test/java` e o esquema de *packages* deve ser um espelho do esquema de *packages* da `src/main/java`. Desta forma o contexto dos testes está divididos consoante os *packages*.

Desenvolver testes unitários para a *framework* Spring tem as suas particularidades, como visto anteriormente o Spring utiliza anotações para identificar os seus componentes, e existe um conjunto de anotações que permite produzir testes unitários.

- `@SpringBootTest` — permite criar um `ApplicationContext` para ser utilizado nos testes.
- `@WebMvcTest` — configura automaticamente a infraestrutura Spring MVC para os testes de unitários.
- `@MockBean` — cria um mock da classe.

- `@Test` — indica que o método é um teste.
- `@BeforeEach` — o método é executado antes de cada teste
- `@AfterEach` — o método é executado antes de cada teste
- `@ActiveProfiles` — indica qual o perfil activo, este deve corresponder a um application.properties com o mesmo nome
- `@DataJpaTest` — fornece algumas configurações padrão necessárias para testar a camada de dados
- `@AutoConfigureTestDatabase` — permite indicar que se vai utilizar uma auto configuração para a base de dados de teste
- `@WithMockUser` — indica que o metodo irá utilizar um utilizador simulado.

Para a API pública foram desenvolvidos 25 testes com uma cobertura de 100% das classes e com 72% das linhas cobertas e para a API privada foram feitos 153 testes com uma cobertura de 83% das classes e com 42% das linhas cobertas. Os tempos de execução dos testes estão disponíveis no Anexo E.

Os testes desenvolvidos para a camada de apresentação foram feitos com a aplicação Cypress. Para tal foi necessário executar o seguinte comando `npm install --save-dev cypress`. Com esta instalação o Cypress adiciona ao projecto a pasta `cypress`.

Para correr os testes é possivel adicionar ao package.json na secção scripts novos comandos que são depois executados via npm.

```
scripts:{
  "cypress:open": "cypress open",
  "cypress:run": "cypress run",
}
```

Para a camada de apresentação foram desenvolvidos 22 testes divididos entre o módulo público e o módulo de gestão com os tempos de execução apresentados na Tabela 7.1.

Ficheiro	Nº de testes	Duração
management/email_test.cy.js	2	1seg
management/eventOccurrence_test.cy.js	2	1seg
management/event_test.cy.js	2	1seg
management/form_test.cy.js	2	1seg
management/homepage_test.cy.js	1	795ms
management/users_test.cy.js	2	1seg
public/eventoccurrence_test.cy.js	1	1seg
public/event_test.cy.js	2	1seg
public/event_test.cy.js	2	1seg
public/homepage_test.cy.js	2	974ms
public/register_test.cy.js	4	6seg

Tabela 7.1: Tempos de execução dos testes da camada de apresentação.

Para validar um sistema como um todo, foram efectuados testes de usabilidade, compostos por um conjunto de acções que têm como objectivo validar a qualidade de uso da perspetiva do utilizador final. Estes testes envolvem um grupo de pessoas que opinaram sobre a qualidade da plataforma, desde as dificuldades encontradas às melhorias que são necessárias. Estes testes de usabilidade devem seguir os seguintes critérios: Interface Amigável, válida se os participantes que completam a tarefa que lhes foi concluída sem muitas dificuldades; Eficiência, quanto menos tempo for necessário para finalizar uma tarefa, melhor é o seu design; Satisfação, verifica se os participantes são capazes de terminar as tarefas sem que tenham reclamações; Erros, validar se durante o teste são encontrados erros gramaticais nos textos da aplicação ou anomalias de design. Estes testes foram feitos ao longo do projecto pela equipa do ISEL ALIVE.

8

Conclusão

Foi desenvolvido um sistema que, embora ainda não completamente pronto para um ambiente final, cumpre a maioria dos requisitos propostos, e deixa uma base sólida que pode ser trabalhada no futuro. A razão principal de alguns dos requisitos não se encontrarem completamente compridos deveu-se a uma estimativa optimista em termos de esforço de execução de certas tarefas, tais como criação e edição de formulário, que obrigou a um replaneamento do trabalho, traduzindo-se na necessidade de reduzir a prioridade de algumas funcionalidades em prol de garantir a estabilidade do sistema entregue dentro da calendarização disponível. Algumas funcionalidades, nomeadamente a visão de orador, a gestão de actividades exteriores e a emissão de certificados, não foram implementadas. O subsistema de envio de emails foi implementado de forma simplificada, enviando mensagens através de servidores Gmail, sendo necessária uma nova implementação que seja compativel com os serviços de email do ISEL. Todos os restantes requisitos foram cumpridos. O sistema final encontra-se todo encapsulado sob forma de um ficheiro docker-compose, o que facilita a execução de várias instâncias deste sistema, para os vários eventos que o ISEL organiza e outros futuros. Desta forma, o *Docker* revelou-se uma ferramenta fundamental para o sucesso deste projecto. Outro serviço que impulsionou o desenvolvimento foram os *Bitbucket pipelines*, que apesar de não aproveitado na sua totalidade, dadas limitações de acesso à máquina hospedeira final, permitiram automatizar a maioria do processo de geração de artefactos final do sistema, possibilitando acesso rápido as várias iterações do projecto ao longo do seu tempo de desenvolvimento. Em suma, o trabalho futuro deverá incluir a finalização dos requisitos em falta, as melhorias ao subsistema de email e à

pipeline de integração continua.

Referências

[1] (Nov. de 2022), URL: https://www.bizzabo.com/.

[2] (Nov. de 2022), URL: https://splashthat.com/.

[3] (Nov. de 2022), URL: https://socio.events/.

[4] (Nov. de 2022), URL: https://www.capterra.com/event-management-software/#top-20.

[5] (Nov. de 2022), URL: https://www.g2.com/categories/event-management-platforms?utf8=%E2%9C%93\&selected_view=grid.

[6] (Nov. de 2022), URL: https://www.bizzabo.com/event-registration.

[7] (Nov. de 2022), URL: https://socio.events/event-registration-software.

[8] (Nov. de 2022), URL: https://splashthat.com/custom-event-pages.

[9] (Set. de 2023), URL: https://support.splashthat.com/hc/en-us/articles/4409645021581-Getting-Paid-with-Stripe.

[10] (Nov. de 2022), URL: https://help.socio.events/en/articles/4235633-stripe-socio.

[11] (Nov. de 2022), URL: https://socio.events/virtual-event-platform.

[12] (Nov. de 2022), URL: https://splashthat.com/product/virtual.

[13] (Nov. de 2022), URL: https://www.bizzabo.com/virtual-experience.

[14] (Nov. de 2022), URL: https://socio.events/mobile-event-app.

[15] (Nov. de 2022), URL: https://splashthat.com/mobile-check-in.

[16] (Nov. de 2022), URL: https://www.bizzabo.com/event-app.

[17] (Nov. de 2022), URL: https://splashthat.com/event-roi.

[18] (Nov. de 2022), URL: https://socio.events/integrations.

[19] (Set. de 2023), URL: https://www.bizzabo.com/product/event-data-analytics.

[20] Eric B. Blancaflor, Gabriel Angelo B. Dela Cruz, Raya Shane C. Rabanal & Jerome Patrick S. Ramos, "Cardinal Connect: A Student Organization Events Management System", em *Proceedings of the 8th International Conference on Management of E-Commerce and e-Government*, sér. ICMECG '21, New York, NY, USA: Association for Computing Machinery, 2022, ISBN: 9781450390545. DOI: 10.1145/3483816.3483835. URL: https://doi.org/10.1145/3483816.3483835.

[21] Maria Rona L. Perez, Ace C. Lagman & Rossana T. Adao, "Event Management Solution Using Web Application Platform", em *Proceedings of the 2017 International Conference on Information Technology*, sér. ICIT '17, New York, NY, USA: Association for Computing Machinery, 2017, ISBN: 9781450363518. DOI: 10.1145/3176653.3176677. URL: https://doi.org/10.1145/3176653.3176677.

[22] M. Ashok Kumar, Ch. Mohan Srinivas, K. Vishnu Vardhan Reddy & K. Kiran Kumar, "College Activity Management System", em *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2018. DOI: 10.1109/ICCONS.2018.8662930.

[23] Thomas Allweyer, *BPMN 2.0: Introduction to the standard for business process modeling*. Books on Demand, 2016.

[24] (Dez. de 2022), URL: https://www.uml.org/.

[25] (Nov. de 2023), URL: https://www.omg.org/spec/BPMN/2.0.2/PDF.

[26] (Jan. de 2023), URL: https://nodejs.org/en/.

[27] (Jan. de 2023), URL: https://expressjs.com.

[28] (Jan. de 2023), URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction.

[29] Mark Heckler, *Spring Boot: Up and Running*. O'Reilly Media, Inc., 2021, ISBN: 9781492076988; 1492076988.

[30] (Jan. de 2023), URL: https://spring.io/web-applications/.

[31] (Jan. de 2023), URL: https://reactjs.org/docs/introducing-jsx.html.

[32] (Jan. de 2023), URL: https://reactjs.org/.

[33] (Jan. de 2023), URL: https://reactjs.org/docs/jsx-in-depth.html.

[34] (Jan. de 2023), URL: https://angular.io/guide/what-is-angular.

[35] Shyam Seshadri Brad Green, *AngularJS*. O'Reilly Media, 2013, ISBN: 1449344852; 9781449344856.

[36] (Ago. de 2023), URL: https://bitbucket.org/product/features/pipelines.

[37] (Ago. de 2023), URL: https://github.com/features/actions.

[38] (Ago. de 2023), URL: https://docs.gitlab.com/ee/ci/.

[39] James Turnbull, *The Docker Book*. 2016, ISBN: 9780988820203.

[40] Michael Kaufmann Andreas Meier, *SQL & NoSQL Databases (Models, Languages, Consistency Options and Architectures for Big Data Management)*. Springer Vieweg, 2019, ISBN: 9783658245481; 9783658245498.

[41] Stanley Williams, "Business process modeling improves administrative control", *The impact of information technology on management operation. — Princeton: Auerbach*, 1971.

[42] (Ago. de 2023), URL: https://www.postgresql.org/.

[43] (Set. de 2023), URL: https://start.spring.io/.

[44] Sebástian Peyrott, *JWT Handbook*. Auth0 Inc.

[45] (Ago. de 2023), URL: https://hub.docker.com/.

[46] (Ago. de 2023), URL: https://bitbucket.org/.

Diagramas *BPMN*

Neste anexo encontram-se os BPMN desenvolvidos durante o projecto.

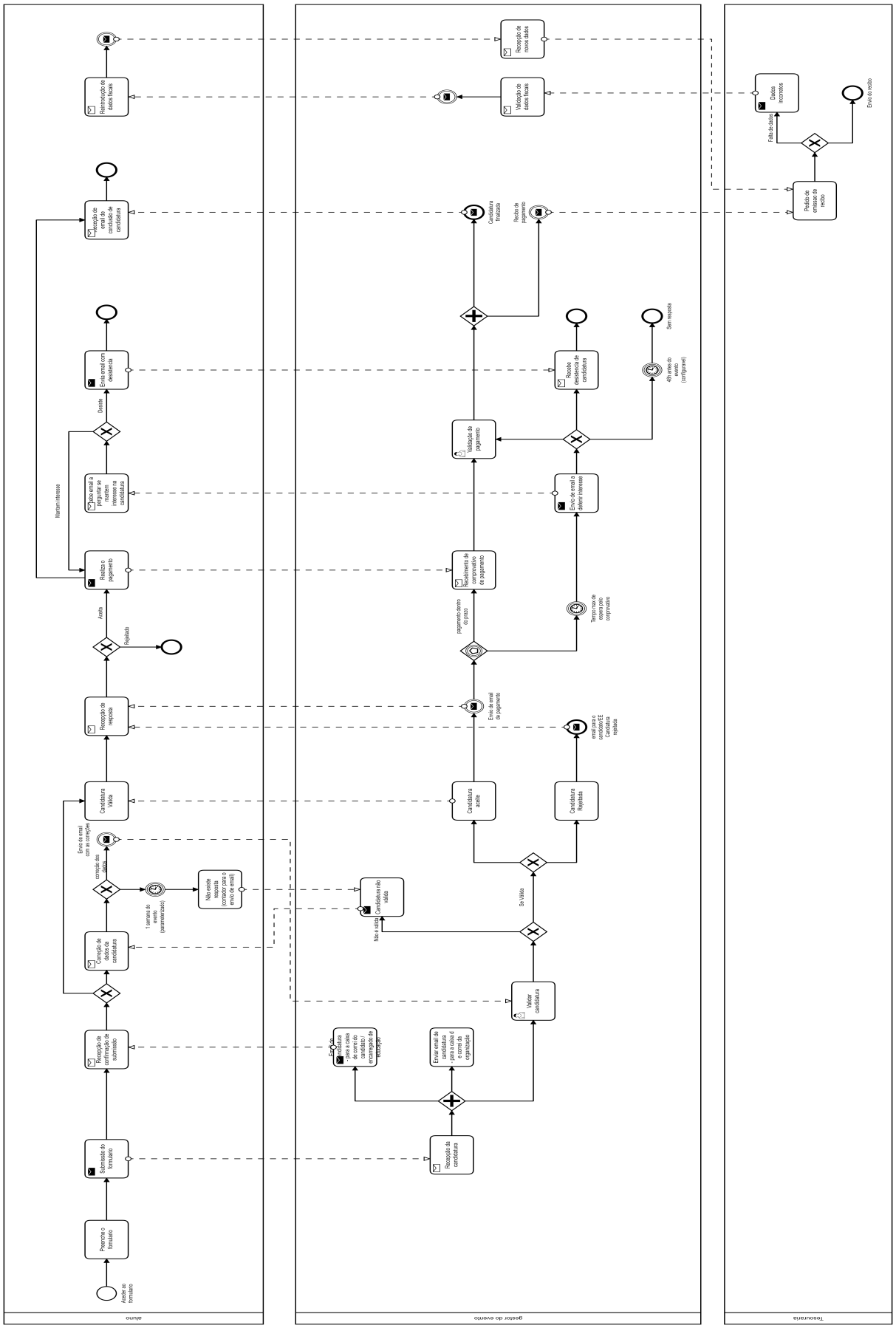

Figura A.1: Processo de candidatura.

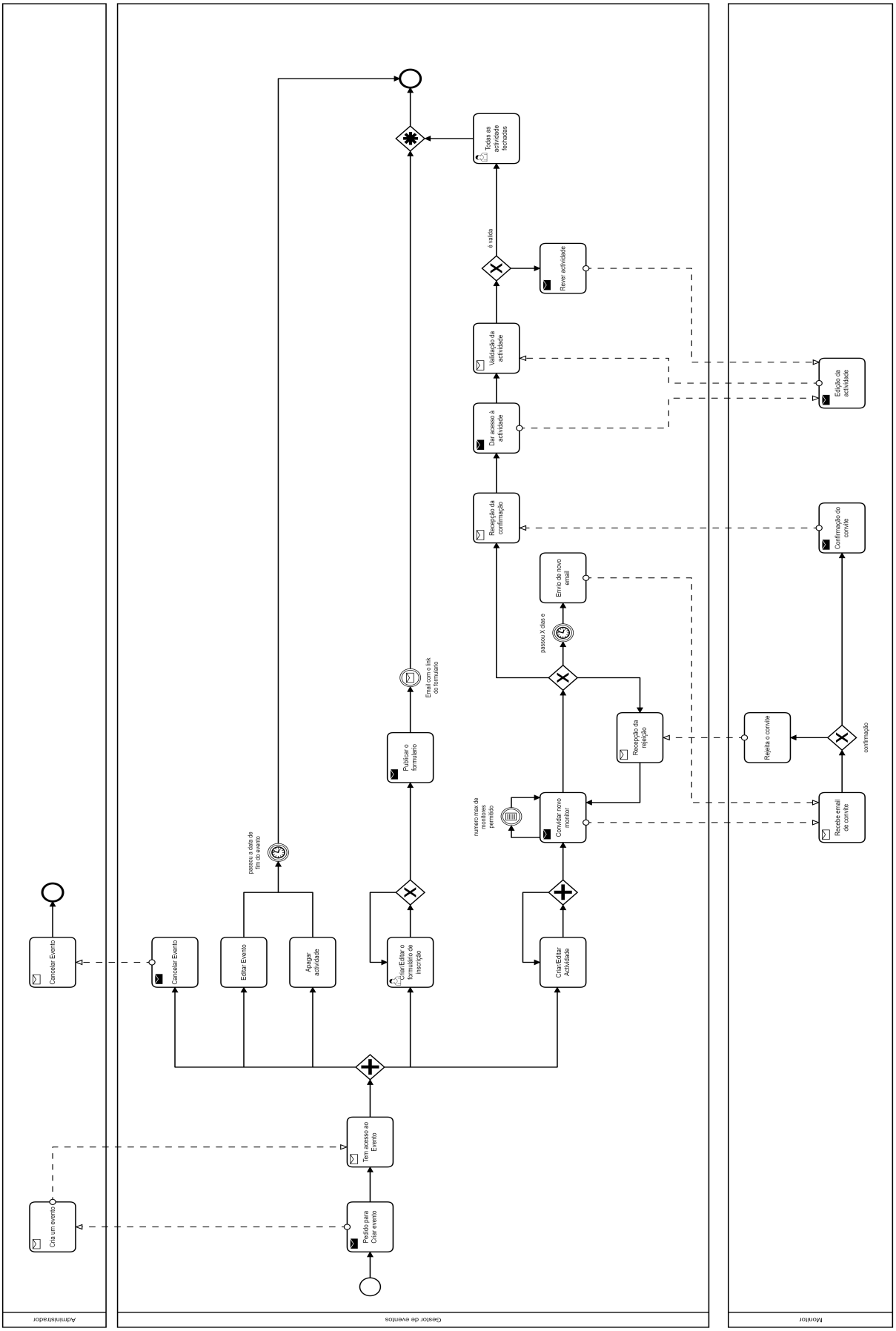

Figura A.2: Processo de criação de evento.

B

Maquetes da camada de apresentação

Neste anexo encontram-se as maquetes produzidas para o desenvolvimento deste projeto.

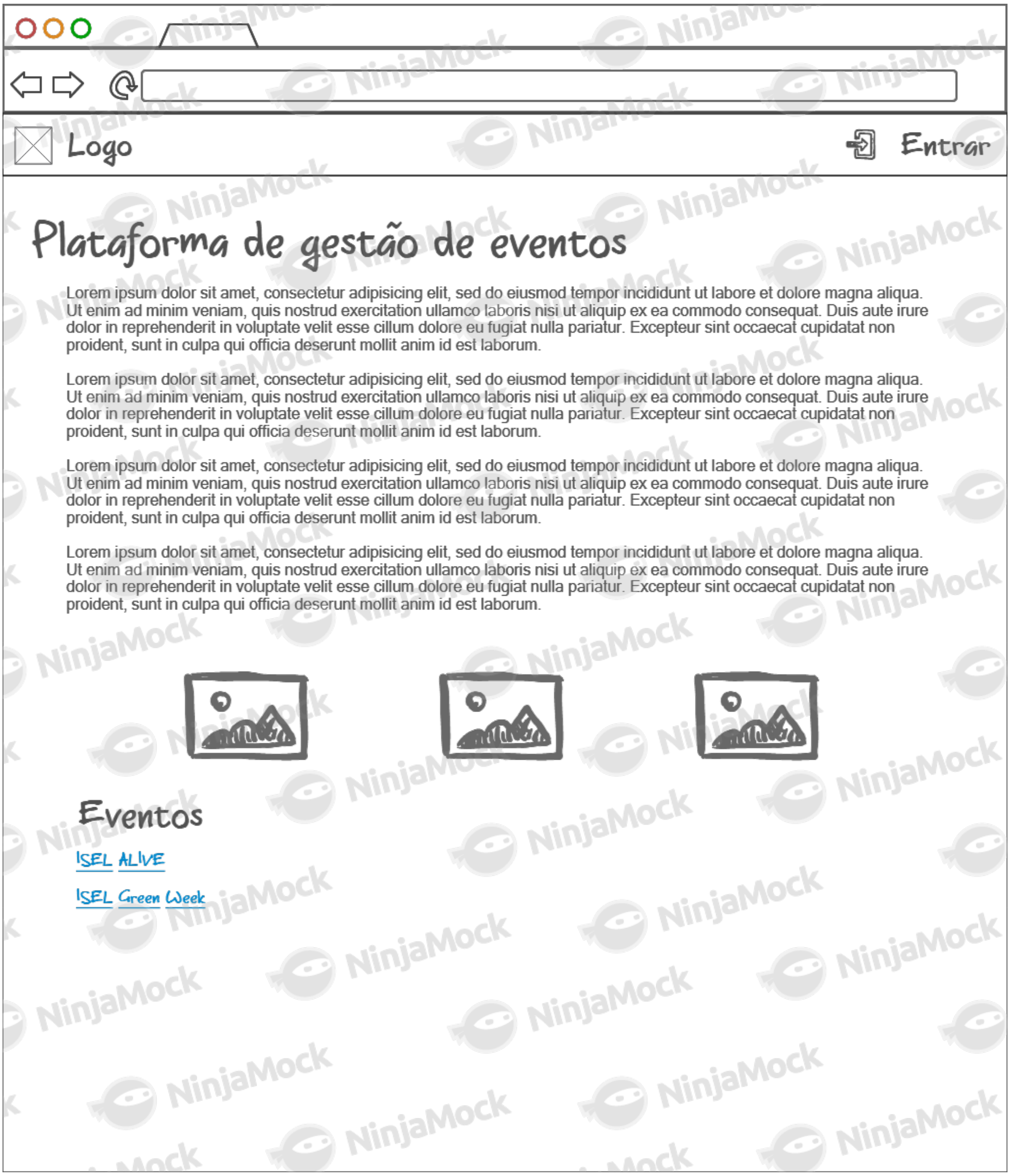

Figura B.1: Pagina Inicial.

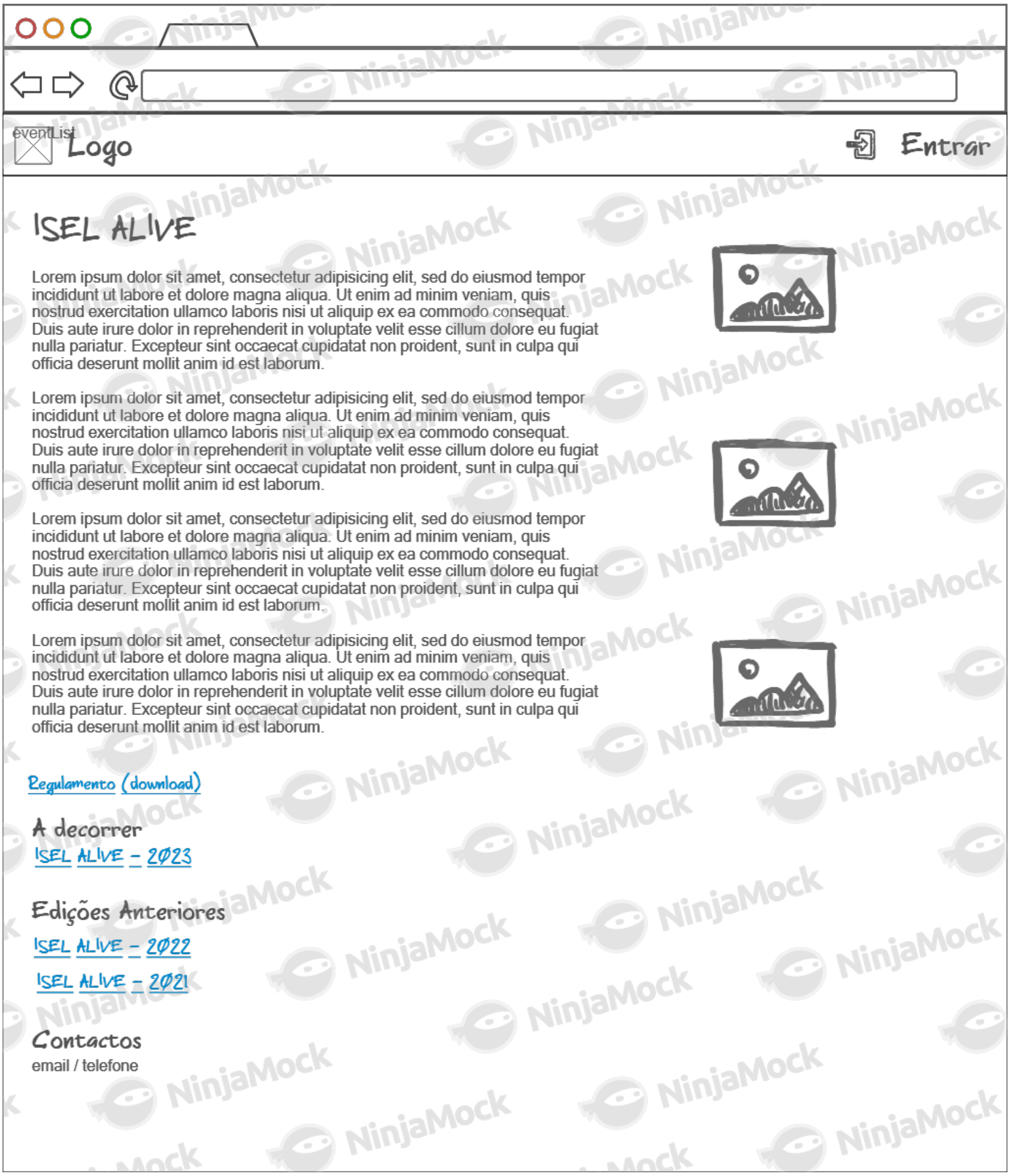

Figura B.2: Página do evento.

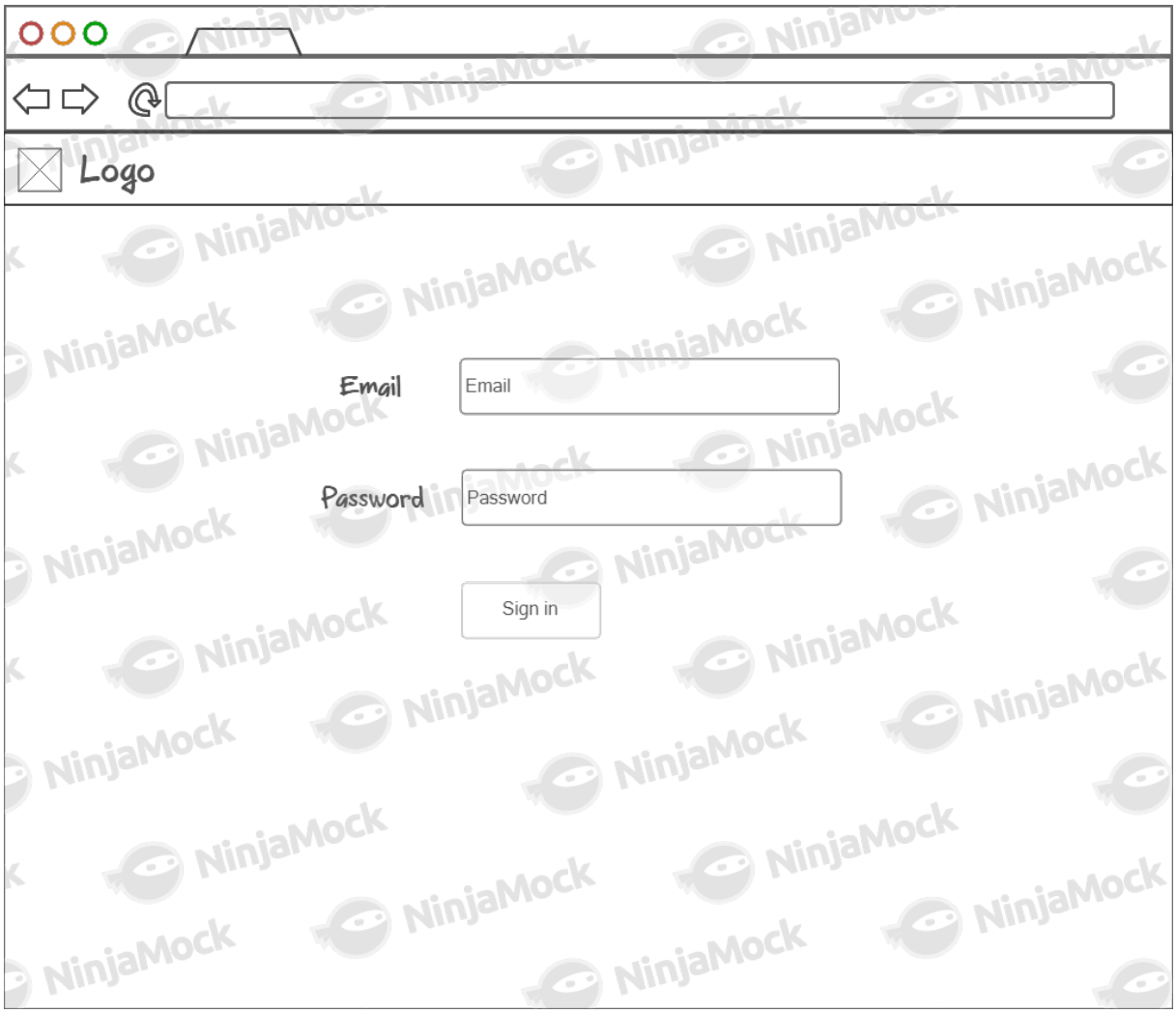

Figura B.3: Página de autenticação.

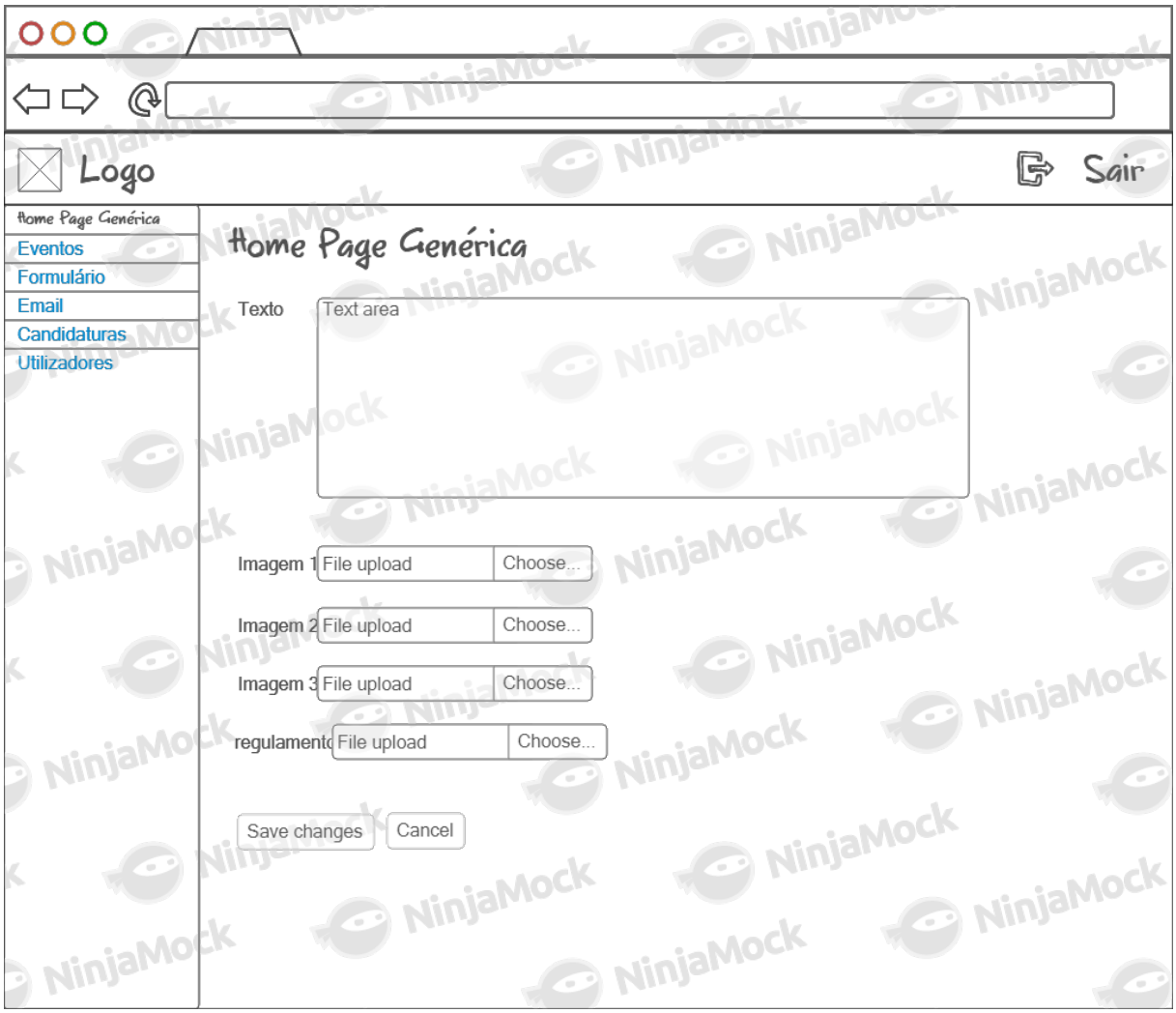

Figura B.4: Página de edição da página inicial.

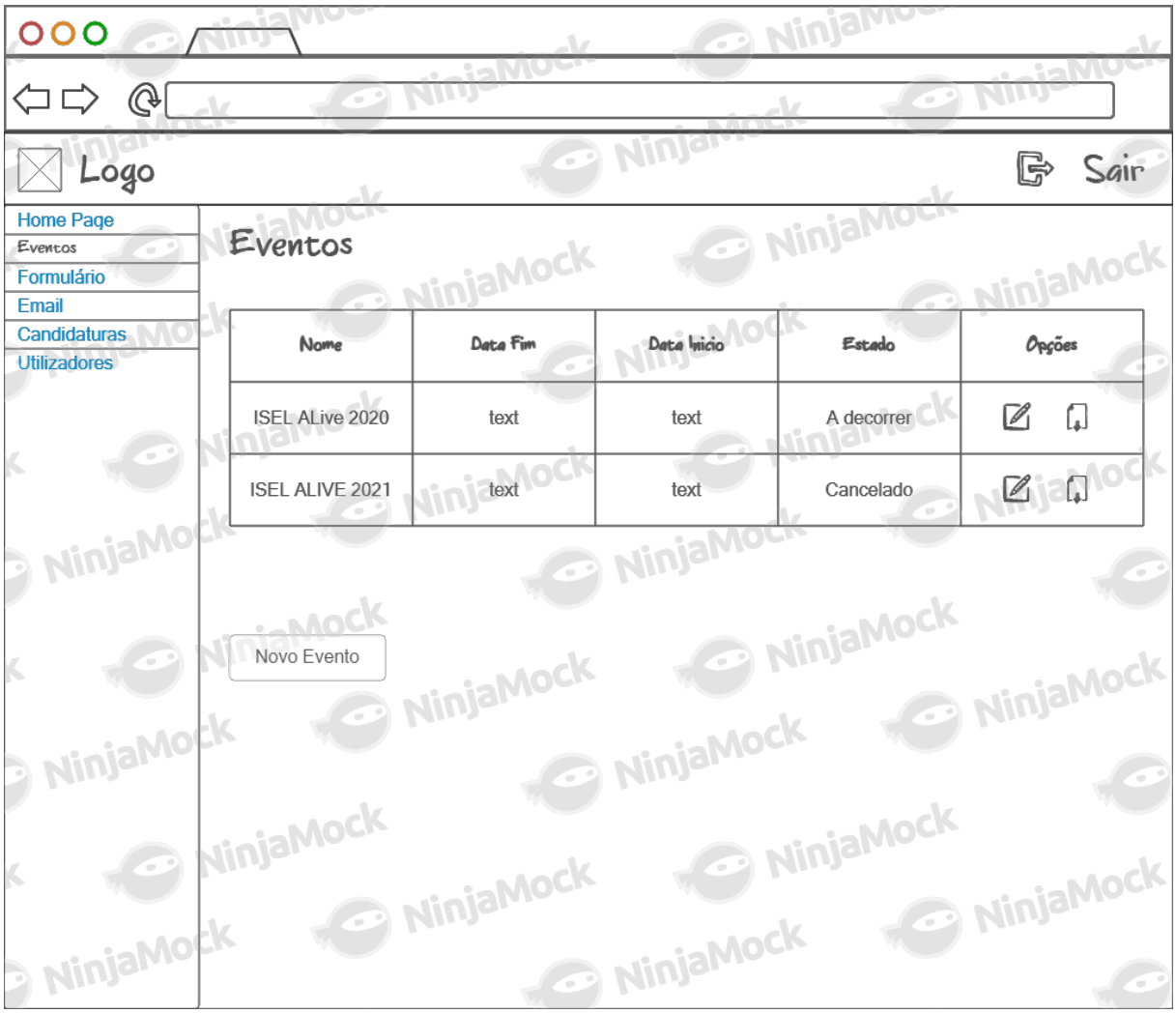

Figura B.5: Lista de Eventos.

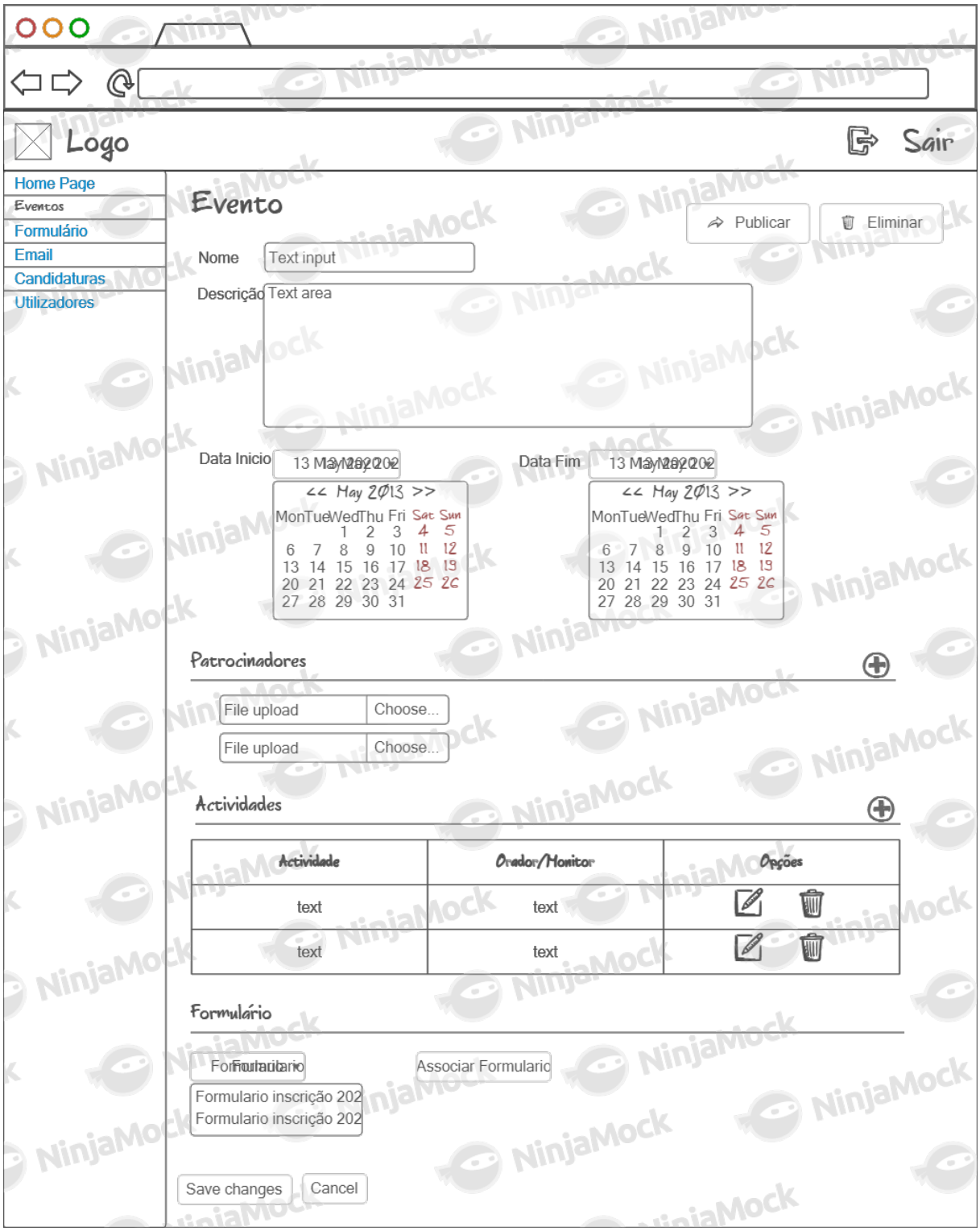

Figura B.6: Detalhe do evento.

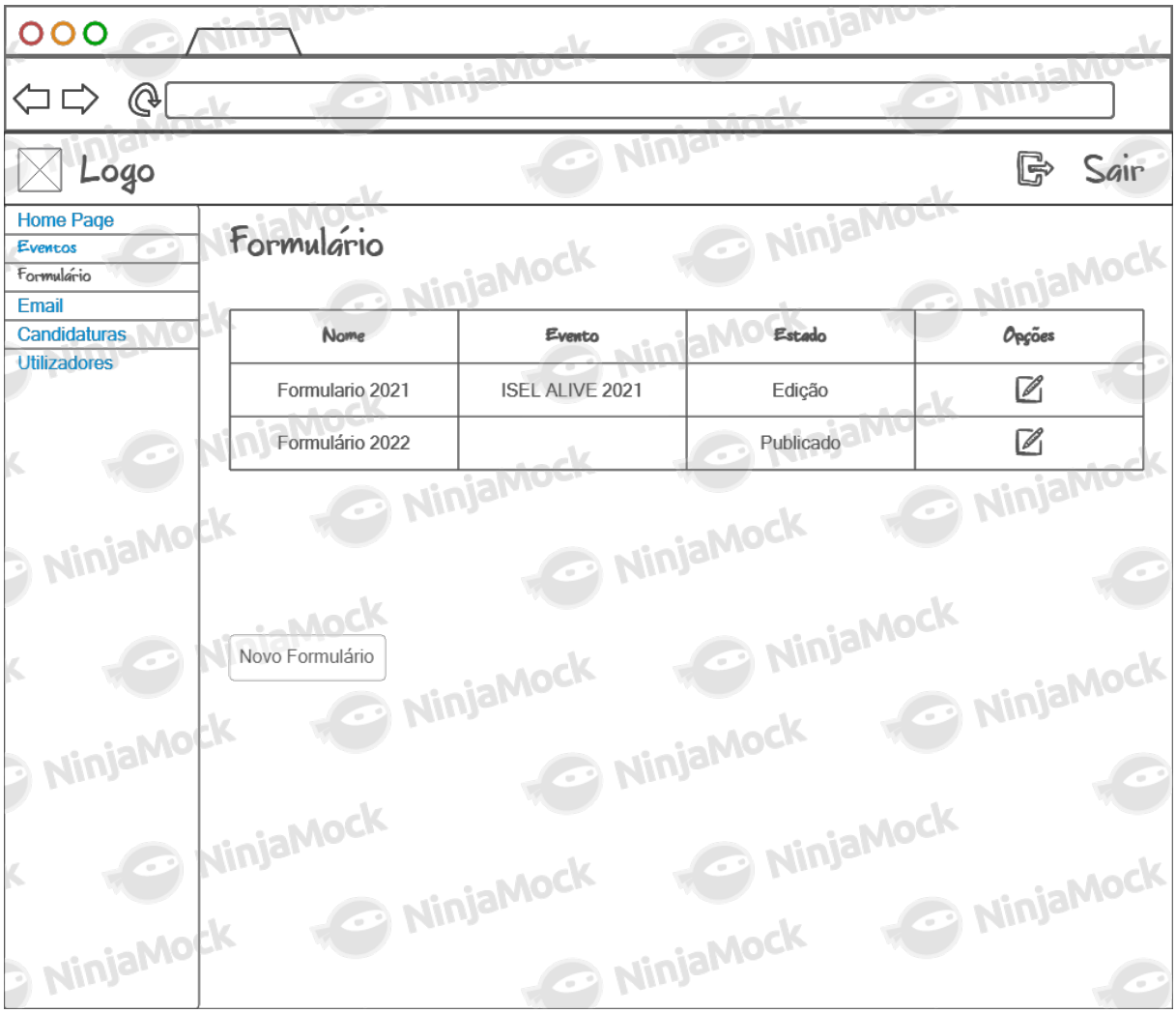

Figura B.7: Lista de formulários.

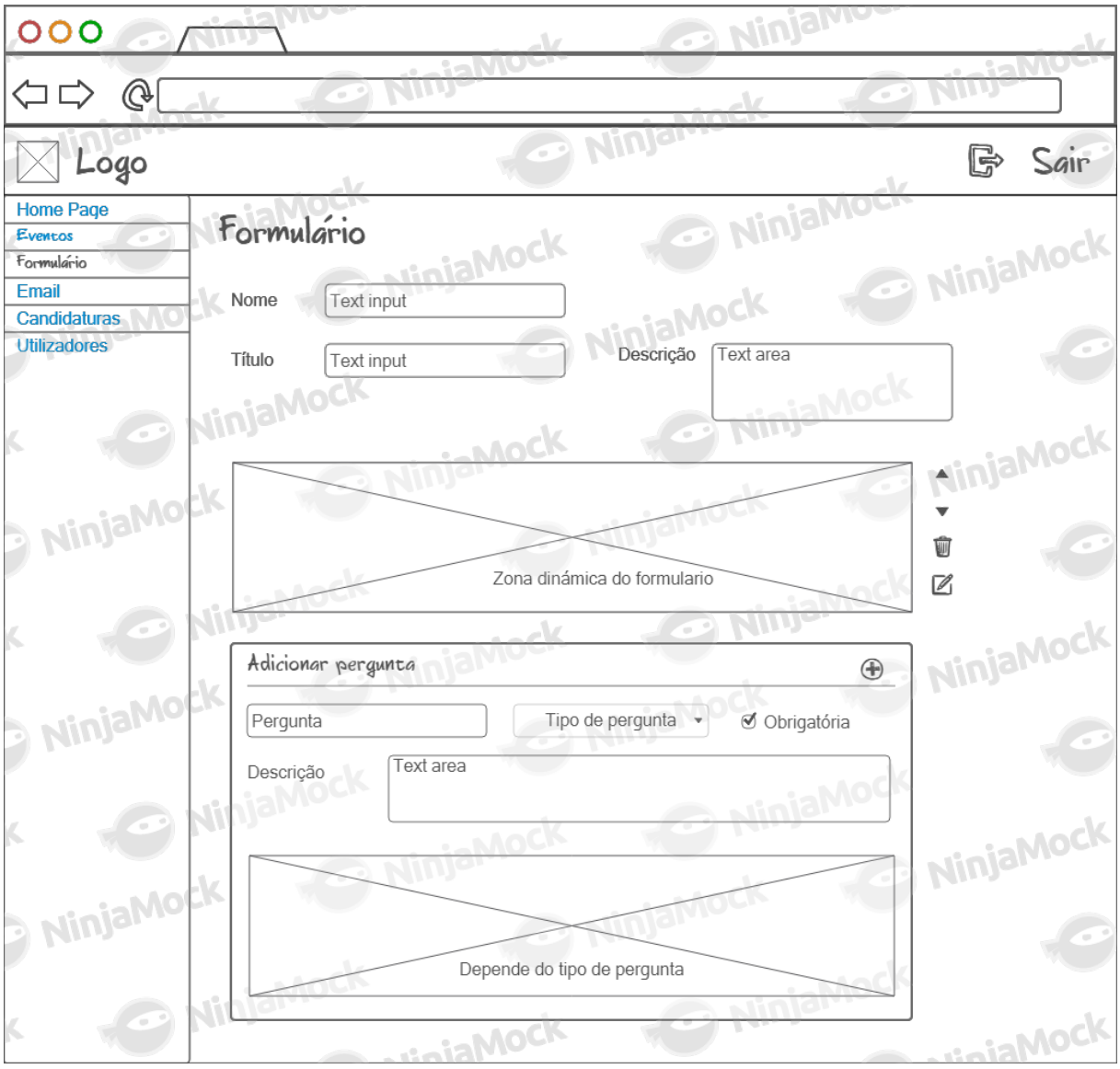

Figura B.8: Detalhe do formulário.

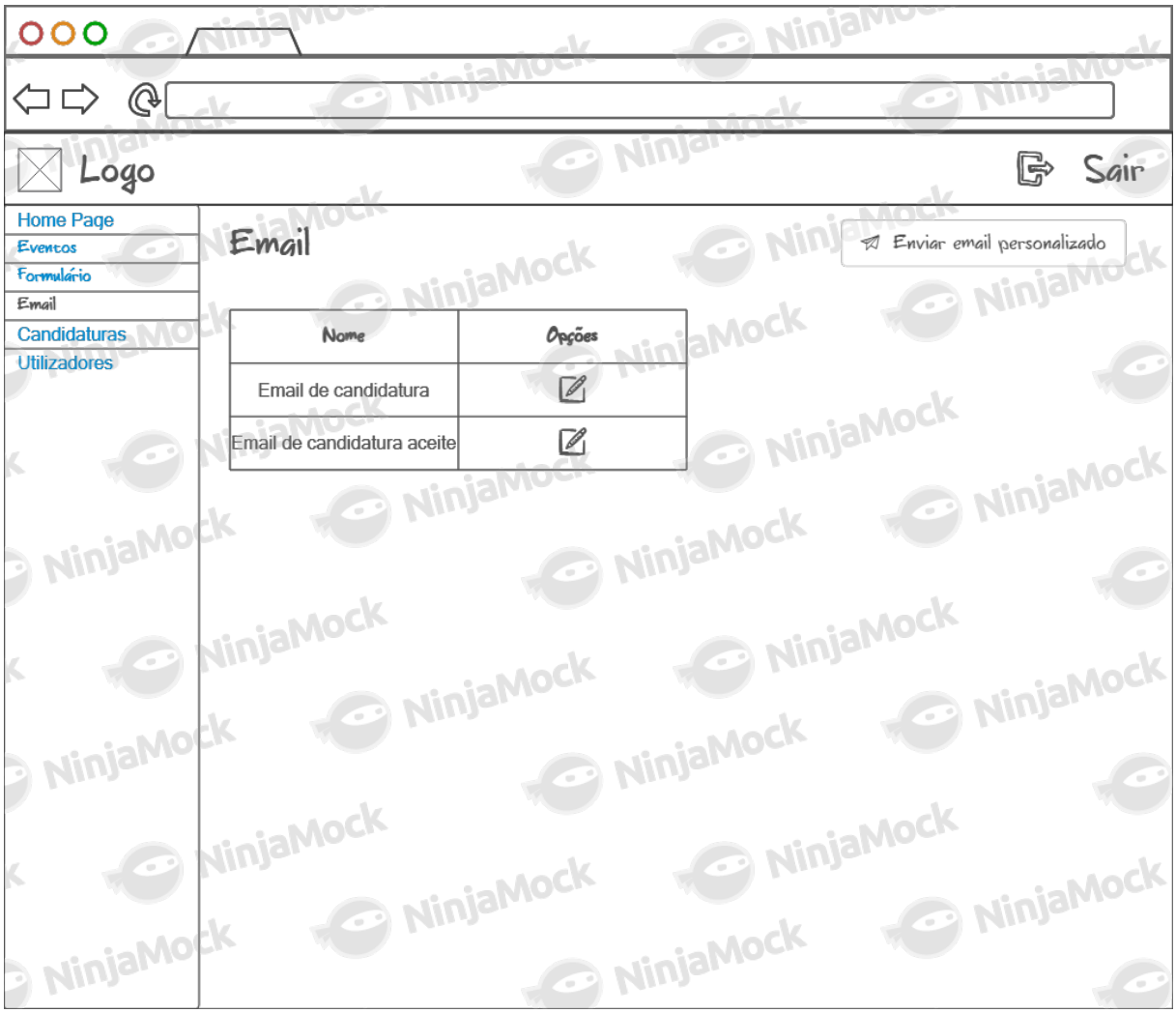

Figura B.9: Lista de emails.

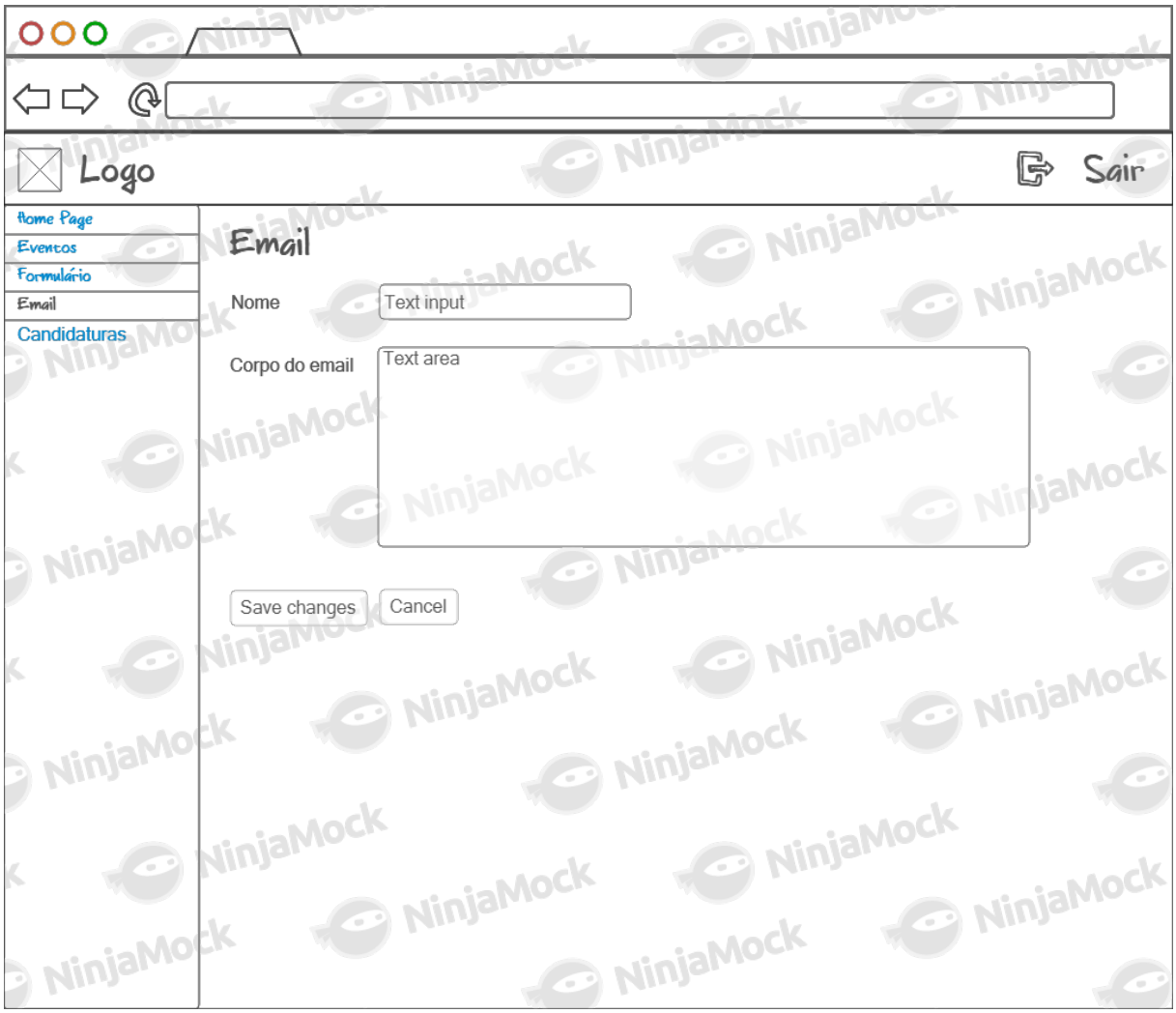

Figura B.10: Detalhe do email.

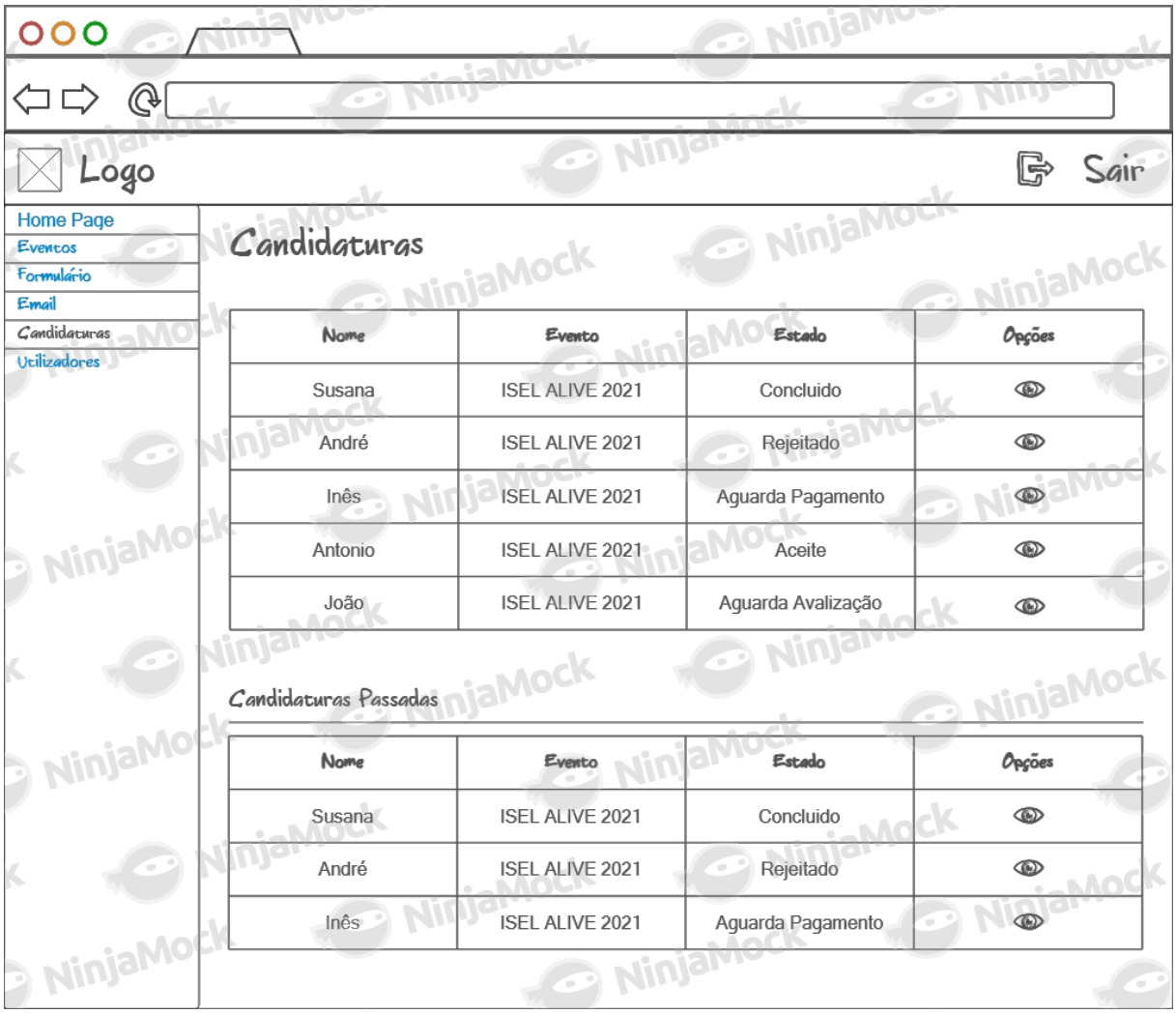

Figura B.11: Lista de candidaturas.

Figura B.12: Detalhe da candidatura.

C

Endpoints da API de gestão

Este anexo contem a lista completa de endpoints da API de gestão.

Método HTTP	**Url**	**Descrição**
GET	`/application`	devolve duas listas uma de candidaturas currentes e outra candidaturas passadas
GET	`/application/current`	devolve a lista de candidaturas currentes
GET	`/application/past`	devolve a lista de candidaturas passadas
GET	`/application/{id}`	devolve uma candidatura com base no id
PUT	`/application/{id}/accept`	aceita uma candidatura com base no id
PUT	`/application/{id}/reject`	rejeita uma candidatura com base no id
PUT	`/application/{id}/payment`	submete o compovativo de paagamento de uma candidatura com base no id
GET	`/email/`	devolve a lista de emails

GET	`/email/{id}`	devolve um email com base no id
PUT	`/email/{id}`	actualiza um email com base no id
GET	`/email/tags`	devolve a lista de tags de email
GET	`/event/`	devolve a lista de eventos
POST	`/event/`	cria um novo evento
GET	`/event/{id}`	devolve um evento com base no id
PUT	`/event/{id}`	actualiza um evento com base no id
DELETE	`/event/{id}`	apaga um evento com base no id
GET	`/event/contactTypes`	devolve a lista de tipos de contacto
GET	`/eventoccurrence/`	devolve a lista de edições
POST	`/eventoccurrence/`	cria uma edição de um evento
GET	`/eventoccurrence/{id}`	devolve uma edição de um evento com base no id
POST	`/eventoccurrence/{id}`	actualiza uma edição de um evento com base no id
DELETE	`/eventoccurrence/{id}`	apaga um evento com base no id
PUT	`/eventoccurrence/{id}/publish`	publica a edição de um evento com base no id
PUT	`/eventoccurrence/{id}/cancel`	cancela a edição de um evento com base no id
GET	`/eventoccurrence/{id}/form`	devolve o formulario para a edição de um evento com base no id
PUT	`/eventoccurrence/{id}/application`	submete uma candidatura para a edição de um evento com base no id
GET	`/eventoccurrence/forms`	devolve a lista de formularios
GET	`/eventoccurrence/events`	devolve a lista de eventos

GET	`/form/`	devolve a lista de formulario
POST	`/form/`	cria um novo formulario
GET	`/form/{id}`	devolve um formulario com base no id
PUT	`/form/{id}`	actualiza um formulario com base no id
DELETE	`/form/{id}`	apaga um formulario com base no id
GET	`/form/qtypes`	devolve a lista de tipos de pergunta
GET	`/user/`	devolve a lista de utilizadores
GET	`/user/{id}`	devolve um utilizador com base no id
PUT	`/user/{id}`	actualiza um utilizador com base no id
GET	`/user/roles`	devolve a lista de roles
GET	`/homepage/`	devolve a informação relativa à homepage
POST	`/homepage/`	actualiza a homepage
POST	`/auth/signin`	permite autenticar um utilizador
POST	`/auth/signup`	devolve registar um utilizador

Tabela C.1: Endpoints disponibilizados na API de gestão.

D

Modelo da camada de dados

Neste anexo encontra-se o modelo EA completo da implementação do modelo de dados.

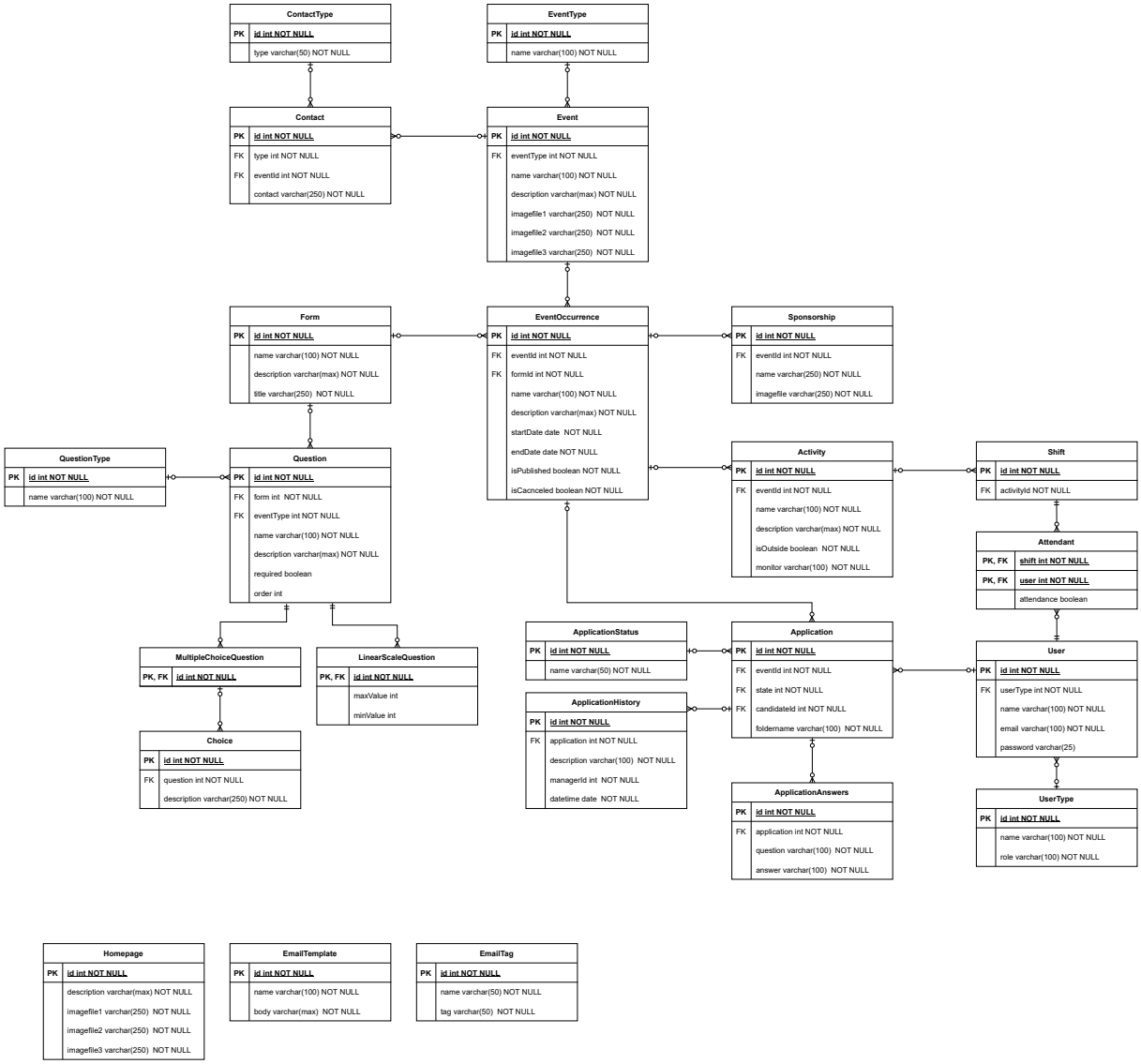

Figura D.1: Modelo Entidade Associação — completo.

E

Teste - Tempos de execução

Neste anexo encontram-se os tempos de execução dos testes desenvolvidos para a camada de serviços.

Teste	Duração (ms)
SmokeTest	
shouldLoadContext	109
ControllerTest	
shouldReturnOkToGetEventOccurrenceTest	140
shouldReturnOkToGetEventTest	16
shouldReturnOkToGetHomepageTest	16
shouldnReturnNotFoundToGetEventOccurrenceTest	16
shouldnReturnBadRequestToGetHomepageTest	15
shouldnReturnNotFoundToGetEventTest	6
HomePageServiceTests	
shouldReturnNullTest	156
shouldReturnHomepageTest	16
EventOccurrenceServiceTests	
shouldReturnNullTest	110
shouldReturnEventOccurrenceTest	15
EventServiceTests	
shouldReturnNullTest	15
shouldReturnEventTest	106
HomepageRepositoryTest	
shouldFindHomepageByIdTest	15
EventRepositoryTest	
shouldFindEventByEventIdTest	275
shouldFindAllEventTest	16
EventOccurrenceRepositoryTest	
shouldFindByIsPublishedIsTrueAndEndDateBeforeAndEventIdTest()	31
shouldFindEventOccurrenceByIdTest	16
shouldFindByIsPublishedIsTrueAndStartDateAfterAndEventIdTest()	31
ActivityRepositoryTest	
shouldFindActivityByEventIdTest	187
shouldFindAllActivitiesTest	16
FileManagerTests	
successGetFileFromFTP	384
failedToGetEncodedImage	282
failedToConnectToFTP	9
failedToGetFileFromFTP	284

Tabela E.1: Tempos de execução dos testes da API Pública.

Teste	Duração (ms)
SmokeTest	
shouldLoadContext	235
FormControllerTest	
deleteShouldReturnBadRequest	125
getAllWithWrongInputsShouldReturnBadRequest	47
createUserShouldReturnBadRequest	110
getShouldReturnForm	31
getShouldReturnBadRequest	16
getQuestionTypesShouldReturnBadRequest	16
updateUserShouldReturnBadRequest	15
getAllShouldReturnBadRequest	16
getAllShouldReturnPage	31
updateUserShouldReturnOK	16
getQuestionTypesShouldReturnOK	31
createUserShouldReturnOK	4
deleteShouldReturnOK	16
EventRepositoryTest	
shouldDeleteAllEventsTest	283
shouldDeleteEventsByIdTest	15
shouldFindAllEventPageableTest	16
shouldFindEventByEventIdTest	15
shouldFindAllEventTest	16
ActivityRepositoryTest	
shouldFindActivityByEventIdTest	46
shouldFindAllActivitiesTest	16
shouldDeleteActivitiesByIdTest	31
UserServiceTest	
shouldReturnAllRolesTest	95
shouldReturnAllUsersTest	16
shouldReturnUserTest	15
shouldReturnNullTest	16
shouldReturnNoneUsersTest	15
EmailRepositoryTest	
shouldFindAllUserTest	16
shouldFindUserByIdTest	15

shouldFindAllUserPageableTest	32
HomepageRepositoryTest	
shouldFindHomepageByIdTest	268
UserControllerTest	
getAllWithWrongInputsShouldReturnBadRequest	15
getShouldReturnUser	16
getShouldReturnBadRequest	16
getRolesShouldReturnBadRequest	16
updateUserShouldReturnBadRequest	15
getAllShouldReturnBadRequest	16
getRolesShouldReturnOK	16
getAllShouldReturnPage	15
updateUserShouldReturnOK	15
ApplicationRepositoryTest	
shouldFindApplicationIdTest	15
EmailControllerTest	
getAllWithWrongInputsShouldReturnBadRequest	16
getShouldReturnUser	15
getShouldReturnBadRequest	16
updateUserShouldReturnBadRequest	15
getAllShouldReturnBadRequest	16
getAllShouldReturnPage	16
updateUserShouldReturnOK	15
getTagsShouldReturnOK	16
getTagsShouldReturnBadRequest	15
EventControllerTest	
deleteShouldReturnBadRequest	141
getAllWithWrongInputsShouldReturnBadRequest	15
getContactTypesShouldReturnOK	48
getShouldReturnBadRequest	16
getAllShouldReturnBadRequest	16
getAllShouldReturnPage	16
getShouldReturnOccurrence	15
getEventsShouldReturnBadRequest	15
deleteShouldReturnOK	16
getShouldReturnNotFound	12

HomepageControllerTest	
updateHomepageShouldReturnOK	15
getShouldReturnHomepage	16
FormRepositoryTest	
shouldFindAllFormPageableTest	370
shouldDeleteEventsByIdTest	15
shouldFindFormByIdTest	16
shouldFindAllFormTest	15
ApplicationControllerTest	
getAllWithWrongInputsShouldReturnBadRequest	172
getPastShouldReturnBadRequest	16
getPastShouldReturnPage	110
getShouldReturnUser	31
getCurrentShouldReturnPage	16
getShouldReturnBadRequest	16
getCurrentWithWrongInputsShouldReturnBadRequest	16
getAllShouldReturnBadRequest	15
getCurrentShouldReturnBadRequest	16
getPastWithWrongInputsShouldReturnBadRequest	18
getAllShouldReturnOK	16
SponsorshipRepositoryTest	
shouldFindEventByEventIdTest	16
ApplicationServiceTest	
shouldReturnNoneCurrentApplicationsTest	157
shouldReturnApplicationTest	20
shouldNotRejectApplicationBecauseStateTest	16
shouldAcceptApplicationTest	16
shouldNotAcceptApplicationBecauseStateTest	15
shouldReturnNonePastApplicationsTest	16
shouldRejectApplicationTest	16
shouldNotAcceptApplicationTest	16
shouldSubmitApplicationTest	15
shouldReturnAllPastApplicationsTest	15
shouldReturnNullTest	17
shouldNotRejectApplicationTest	18
shouldReturnAllCurrentApplicationsTest	16

HomepageServiceTests	
shouldUpdateHomepageAndImagesTest	661
shouldReturnNullTest	16
shouldReturnHomepageTest	15
shouldUpdateHomepageButNotImagesTest	110
ValidatorTest	
failedValidationEmail	16
successValidationEmail	15
successValidationPassword	15
failedValidationPassword	16
EventOccurrenceControllerTest	
deleteShouldReturnBadRequest	16
getAllWithWrongInputsShouldReturnBadRequest	47
getEventsShouldReturnEvents	16
getEventFormShouldReturnOK	67
getShouldReturnBadRequest	20
getEventFormShouldReturnBadRequest	16
publishShouldReturnBadRequest	16
cancelShouldReturnBadRequest	16
getAllShouldReturnBadRequest	15
getAllShouldReturnPage	31
getFormsShouldReturnBadRequest	16
getFormsShouldReturnForms	15
getShouldReturnOccurrence	16
getEventsShouldReturnBadRequest	15
getActiveEventsShouldReturnOK	15
getActiveEventsShouldReturnBadRequest	16
publishShouldReturnOK	24
deleteShouldReturnOK	16
cancelShouldReturnOK	16
EventOccurrenceServiceTests	
shouldReturnNoneEventOccurrenceTest	94
shouldReturnAllEventOccurrenceTest	16
shouldReturnNullTest	15
shouldReturnEventOccurrenceByIdTest	11
EventOccurrenceRepositoryTest	

shouldDeleteAllEventsTest	30
shouldDeleteEventsByIdTest	15
shouldFindAllEventPageableTest	16
shouldFindEventOccurrenceByIdTest	16
shouldFindAllEventTest	16
ApplicationHistoryRepositoryTest	
shouldFindApplicationIdTest	15
FileManagerTests	
successGetFileFromFTP	188
successGetHomeFilePath	4
failedToGetEncodedImage	239
successGetEventFilePath	16
failedToConnectToFTP	16
successGetEventOcFilePath	16
failedToGetFileFromFTP	187
EmailServiceTest	
shouldReturnEmailTest	32
shouldReturnAllEmailsTest	16
shouldReturnNoneEmailsTest	15
shouldReturnAllRolesTest	16
shouldReturnNullTest	16
shouldUpdateEmailTest	19
UserRepositoryTest	
shouldFindAllUserTest	16
shouldFindUserByIdTest	16
shouldFindUserByEmailTest	31
EventServiceTests	
shouldReturnEventsTest	31
shouldCreateEventButNoImagesTest	16
shouldDeleteEventTest	15
shouldReturnEventByIdTest	16
shouldReturnContactTypesTest	16
shouldReturnAllEventTest	14
shouldReturnNullTest	17
shouldReturnNoneEventTest	16

Tabela E.2: Tempos de execução dos testes da API Privada.