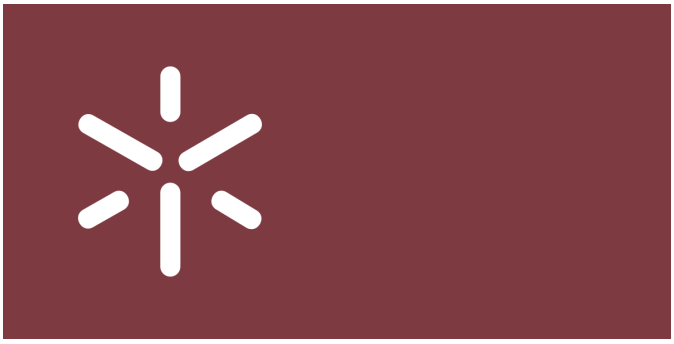


Universidade do Minho

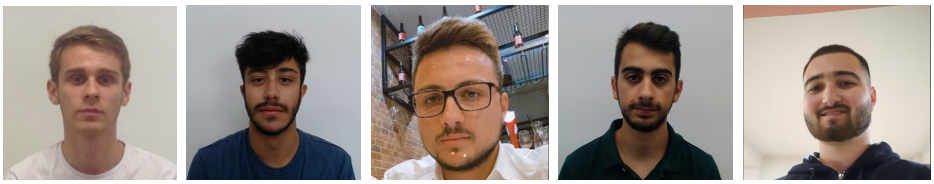
Mestrado em Engenharia Informática

Requisitos e Arquiteturas de Software

RASBet

Grupo XX,[PL2] / Entrega 2

PG47166 Eduardo Teixeira
A86617 Gonçalo Nogueira
A74806 João Amorim
PG46538 José Ferreira
PG46540 Luis Ribeiro



17 de Dezembro de 2021
2021-2022

Índice

Lista de Figuras

Lista de Tabelas

1 Introduction and Goals

1.1 Requirements Overview

Em *Requirements Overview* apenas se faz uma referência à secção onde é feita a obtenção e análise de requisitos funcionais que a equipa realizou, a secção **3.3** *Functional and Data Requirements*.

1.2 Quality goals

Na secção de *Quality Goals* referimos os objetivos obtidos dentro do estudo dos requisitos não funcionais (destacados na secção **4** *Non-Functional Requeriments*) cujo cumprimento é da maior importância para a equipa. Objetivos como *Performance Efficiency*,*Security* e *Operability/Usability* são fundamentais e detalhados nessa secção.

Quality	**Motivation**
Performance Efficiency	O sistema terá de estar preparado e desenhado para um *flow* constante de clientes. Manter uma performance adequada aos recursos disponibilizados
Security	Sistema terá de garantir um mecanismo de confiança para com os dados dos seus utilizadores e de garantir a sua privacidade.
Operability	Sistema acessível aos utilizadores incluidos nas faixas etárias permitidas à sua utilização

Tabela 1.1: *Quality Goals*

1.3 Stakeholder

Para a secção *Stakeholder* apenas se refere a secção onde estão fundamentados aspetos como *costumers and clients* e também os *stakeholders*, nas secções **1.2** *Client, Customer, Stakeholders* e **1.3** *Users of the System* do relatório anterior.

2 Architecture Constraints

2.1 Content

Nesta secção são abordadas os requerimentos que impõe algum tipo de restrições ao desenvolvimento e ás decisões a tomar sobre a arquitetura de software sobretudo quanto à liberdade de design e forma de implementação. Estas restrições podem ir além de um sistema único e serem válidas para todos os sistemas de uma organização.

Estas restrições são importantes para a equipa conhecer onde tem exatamente liberdades de decisão e onde tem de seguir um conjunto de regras impostas pelos *stakeholders*.

2.2 Form

As restrições existentes no projeto podem então ser divididas em diferentes categorias como técnicas, organizacionais e políticas.

Desta forma destacam-se inicialmente os requisitos não funcionais REQN007, REQN008, REQN009, REQN010, REQN014 e REQN015 presentes no relatório anterior.

Os fatores mais importantes são a considerar são os seguintes:

- A obrigatoriadade do uso do esquema de cores indicado pela empresa.
- O nome da plataforma web terá de ser **RASBet** e seguir a capitalização indicada.
- Terá de haver um sistema de notificações que notifique os utilizadores das suas apostas.
- A plataforma terá de estar concluída até ao dia 21 de janeiro de 2022.
- Terá de ser utilizada a framework *Vue* bem como o *Vuetify* de forma a tornar o website responsivo.
- A arquitetura do sistema desenvolvido terá de seguir a estrutura **REST**.

3 Context and scope

Este tópico de comunicação e apresentação de permite-nos delimitar o nosso Sistema. Nesse contexto foi feita uma descrição do ambiente e de como se enquadra a nossa aplicação nesse mesmo ambiente através do uso de vários modelos UML .Essa descrição foi apresentada e detalhada na secção **3.1** 3 **3.2** do relatório anteriormente realizado. Visto que foi referido anteriomente que o tratamento de transações monetárias estava associado a um sistema externo. a equipa decidiu não demonstrar as interfaces associadas, e os inputs e outputs desse dominio específico.

4 Solution Strategy

Através de uma análise detalhada de requisitos não funcionais a equipa reuniu, separando por tópicos uma série destes requisitos que a aplicação deveria possuir. De todos estes é de realçar aqueles que se definiu como sendo prioritários e mais importantes.

Após várias reuniões a equipa decidiu alterar o 3º requisito mais importante visto que este se enquadrava pouco no domínio da nossa aplicação:

REQN013 - ***Aplicação deve garantir ao utilizador segurança em movimentos de conta.***

Ou seja, visto que foi destacado na secção **2.3** *Facts and Assumptions* que se assumia que as transações monetárias de clientes seriam realizadas por um serviço externo à nossa aplicação este 3º requisito fica pouco consolidado e algo confuso no que a que realmente se refere. Do ponto de vista da equipa, o REQN013 refere-se à permanência de dados referentes a valores monetários na conta de um utilizador, para que estes consigam fazer login e logout sem ter a preocupação de estes desaparecerem. De modo a tratar desta ambiguidade a equipa decidiu substituir o 3º requisito mais importante anterior pelo requisito REQN006 visto que este se enquadrava melhor no contexto de relevância na nossa aplicação. O REQN006 está identificado na tabela em baixo apresentada.

Requisito Não Funcional	**ID**	**Abordagem Arquitetural**
Aplicação com Navegabilidade simples e intuitiva	REQN002	Desenvolvimento de interface gráfica através do uso de *Vuetify*, *framework* com libraria que disponibiliza componentes gráficos. O desenvolvimento está associado aos Mockups apresentados na secção 3 de *Functional Requirements*
Velocidade de tarefas e respostas do sistema devem ser realizadas no menor tempo possível.	REQN004	Arquitetura compacta e bem distribuida, usando ferramentas tais como Node.js e Axios para tratamento eficiente de dados sobre uma API e Base de Dados.
Sempre que algo é modificado na base de dados, esta informação deverá ficar atualizada para todos os intervenientes do sistema.	REQN006	Em conjunto com as ferramentas já mencionadas e usando Mongo DB, uma base de dados flexível, escalável e distribuida, capaz de garantir confiabilidade e consistênsia de dados para o tipo de *requests* que serão efetuados.

Tabela 4.1: Requisitos Prioritários

5 Building Block View

5.1 Content

Nesta secção, efetuando uma abstração do *source-code*, será feita uma decomposição estática do sistema, apresentada como uma hierarquia de *white boxes*, contendo *black boxes*, sendo cada nível inferior da hierarquia, mais detalhado do que o seu superior. Serão apresentados os blocos do Sistema, incluíndo módulos, componentes, subsistemas, classes, entre outros.

Esta secção é fundamental para cada documentação referente a uma arquitetura, ajudando a manter uma visão geral do nosso *source code*, permitindo uma comunicação, com os *stakeholders*, sem que seja necessário descrever os vários detalhes da implementação.

5.2 Form

O *Building Block View* é então composto por uma hierarquia definida por coleções de *White Boxes* e *Black Boxes*, sendo que para a documentação da nossa arquiterura, iremos dividir em 3 partes:

- Scope and Context - Representação Geral do Sistema
- Level 1 - *White Box* com a descrição do Sistema em geral, caracterizada pelo conjunto das descrições das *Black Boxes* contidas em cada *Building Block*.
- Level 2 - Caracterização de cada *Building Block* representado no *Level 1*, e de cada *Building Block* interno.

5.3 Scope and Context

Para o *Scope and Context*, foi obtido o seguinte diagrama:

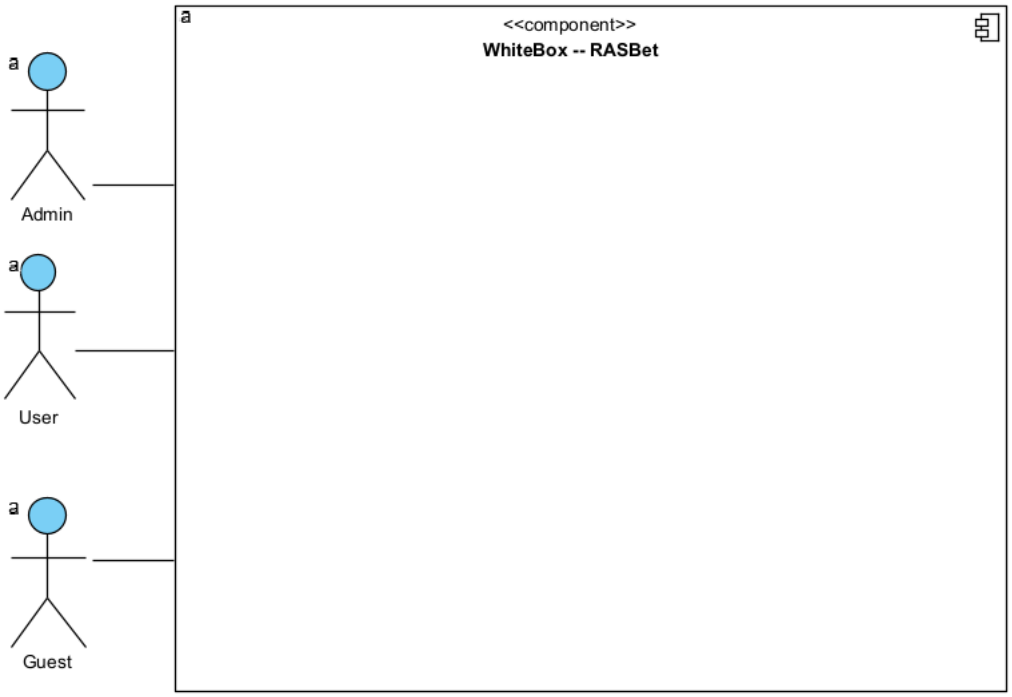


Figura 5.1: Scope and Context

No *Scope and Context*, é apresentado todo o Sistema como um todo, neste caso a Aplicação **RASBet**, e a interação que irá existir com os Atores do Sistema. Neste caso, os Atores serão o Administrador, os Utilizadores registados e os Guests.

5.4 Level 1

Neste nível, é apresentada a *White Box* com a descrição do sistema em geral com os componentes principais.

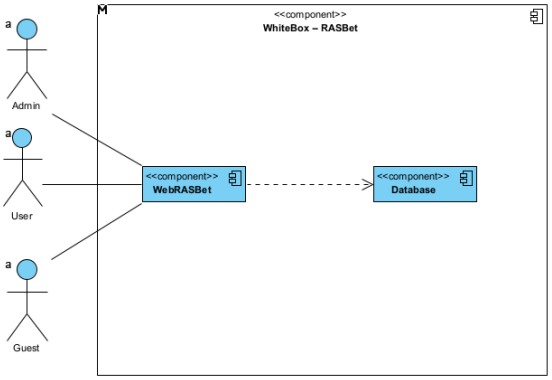


Figura 5.2: Level 1

No caso da nossa arquitetura, estes *Building Blocks* correspondem ao conjunto correspondente ao Frontend e Backend da aplicação, assim como a Base de Dados que possuirá todos os dados do Sistema. A "WebRASBet"tratará todos os pedidos realizados no Sistema, mandando as respostas necessárias em formatos especificos e inteligiveis para o lado da Base de Dados, garantido a consistência de dados de toda a Aplicação.

5.5 Level 2

Neste segundo nível, são caracterizadaos cada um dos *Building Blocks*, apresentando as *Black Boxes* que representarão as funcionalidades de cada *Building Block*

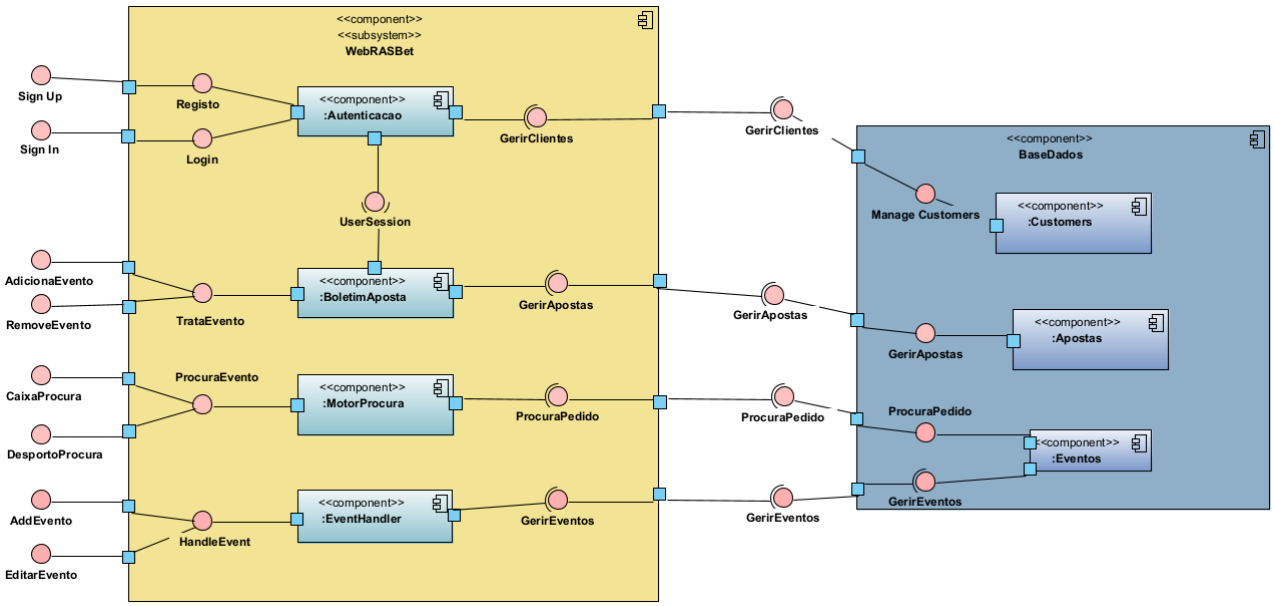


Figura 5.3: Level 2

No que diz respeito a cada *Black Box*, temos que:

- Autenticação - Corresponde a todos os processos necessários para garantir a Autenticação de um Ator no Sistema, incluíndo Registo, Login, Edição de Detalhes de Conta e garantia de que o estado de Autenticação de um Ator se mantêm ao longo de todo o processo de vida de realização de uma aposta ou de alteração de detalhes de conta.

- BoletimAposta - Corresponde ao processo de criação e confirmação de um Boletim de Apostas. O Ator edita o Boletim consoante a sua aposta e no momento da confirmação, é feita a comunicação com a Base de Dados para que os dados da mesma fiquem guardados.

- MotorProcura - Funcionalidades do Sistema para procura de Eventos já existentes, a partir de Menus ou Barra de Pesquisa. "WebRASBet"efetua pedidos à Base de Dados com os dados da pesquisa e recebe respostas a serem apresentadas ao Ator.

- Customers - *Building Block* da Base de Dados correspondente aos dados que caracterizam os Atores do Sistema. Dados são povoados sempre que um Utilizador se Regista e alterados sempre que um Utilizador efetua uma Edição de Detalhes.

- Apostas - *Building Block* da Base de Dados correspondente aos dados que caracterizam as apostas já efetuadas pelos Utilizadores.

- Eventos -*Building Block* da Base de Dados correspondente aos dados que caracterizam os Eventos do Sistema, sendo esta alimentação de dados efetuada pelo *EventHandler*.

- EventHandler - Corresponde a todas as ações do Administrador para a Adição e Remoção de Eventos. Este Administrador, será o responsável por toda a alteração da coleção Eventos na Base de Dados, sendo que este *Building Block* será quem efetuará a devida comunicação com a Base de Dados.

6 Runtime View

6.1 Content

Esta secção descreve o comportamento concreto e a interação dos diferentes componentes do sistema. Esta descrição é feita recorrendo ao uso de diagramas de sequência e diagramas de atividade que representam a forma como o sistema executa os diferentes *Use Cases* ou funcionalidades.

O principal critério utilizado para a escolha das funcionalidades a descrever recai sobre a sua relevância arquitectural não sendo por isso necessário a descrição de vários cenários mas sim a documentação de uma seleção representativa.

Esta secção é de grande importância para uma maior facilidade de comunicação e esclarecimento sobre o funcionamente do sistema bem como a interação dos seus componentes aos *Stakeholders*.

6.2 Form

O *Runtime View* é então composto por um diagrama de sequência e um diagrama de atividade para cada *Use Case* considerado crucial para uma correta implementação das funcionalidades principais do sistema.

Os *Use Cases* escolhidos para uma explicação mais detalhada foram os seguintes:

- Consulta de eventos por equipas ou por desporto.
- Tratamento de aposta
- Gestão de eventos por parte do administrador
- Sistema de notificações

É importante mencionar que os diagramas de atividade já foram apresentados no relatório anterior oa longo da secção **3.2** e por isso não se encontram descritos neste relatório.

6.3 Consulta de Eventos

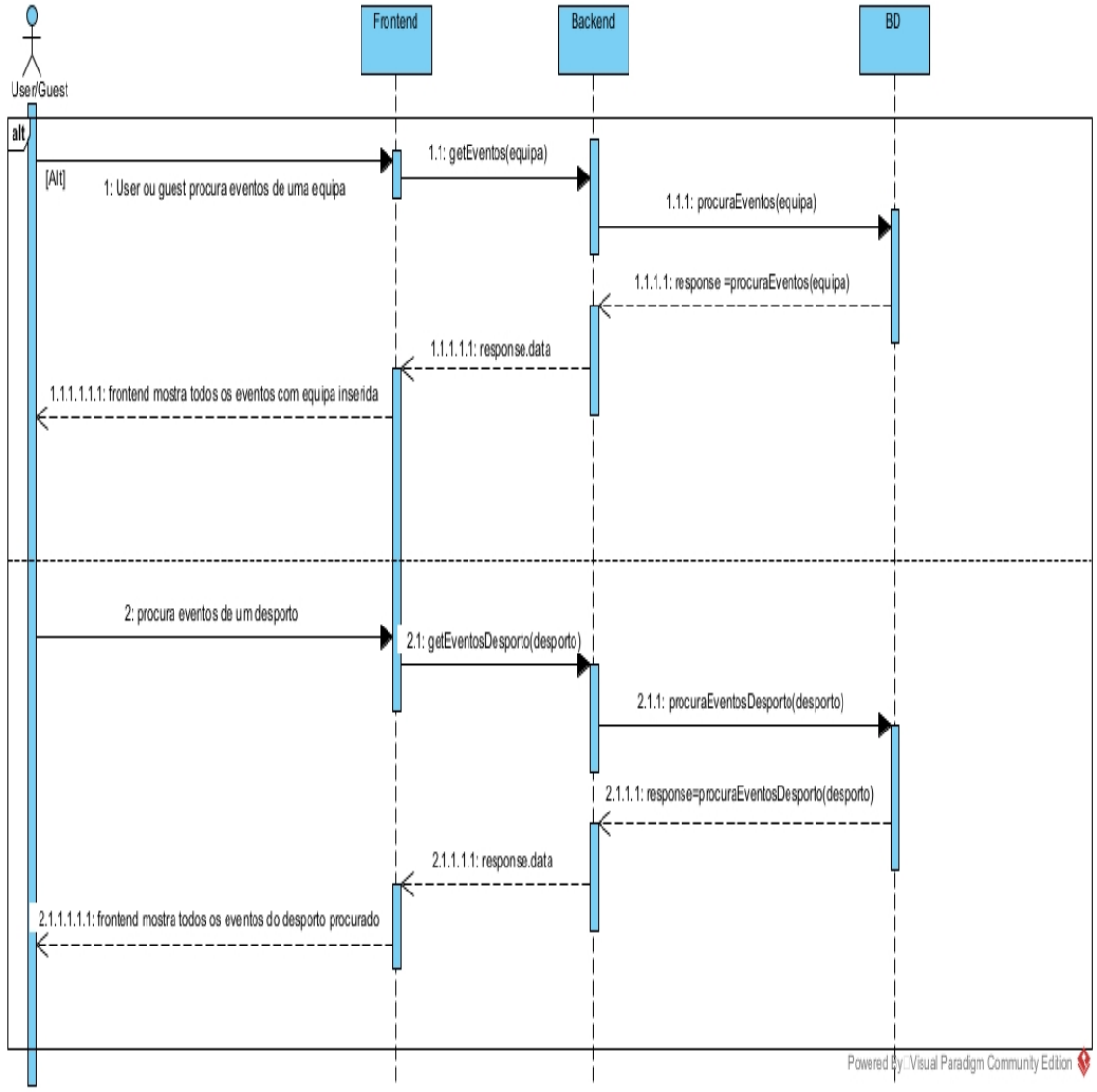


Figura 6.1: Consultar Eventos

6.4 Tratamento de uma aposta

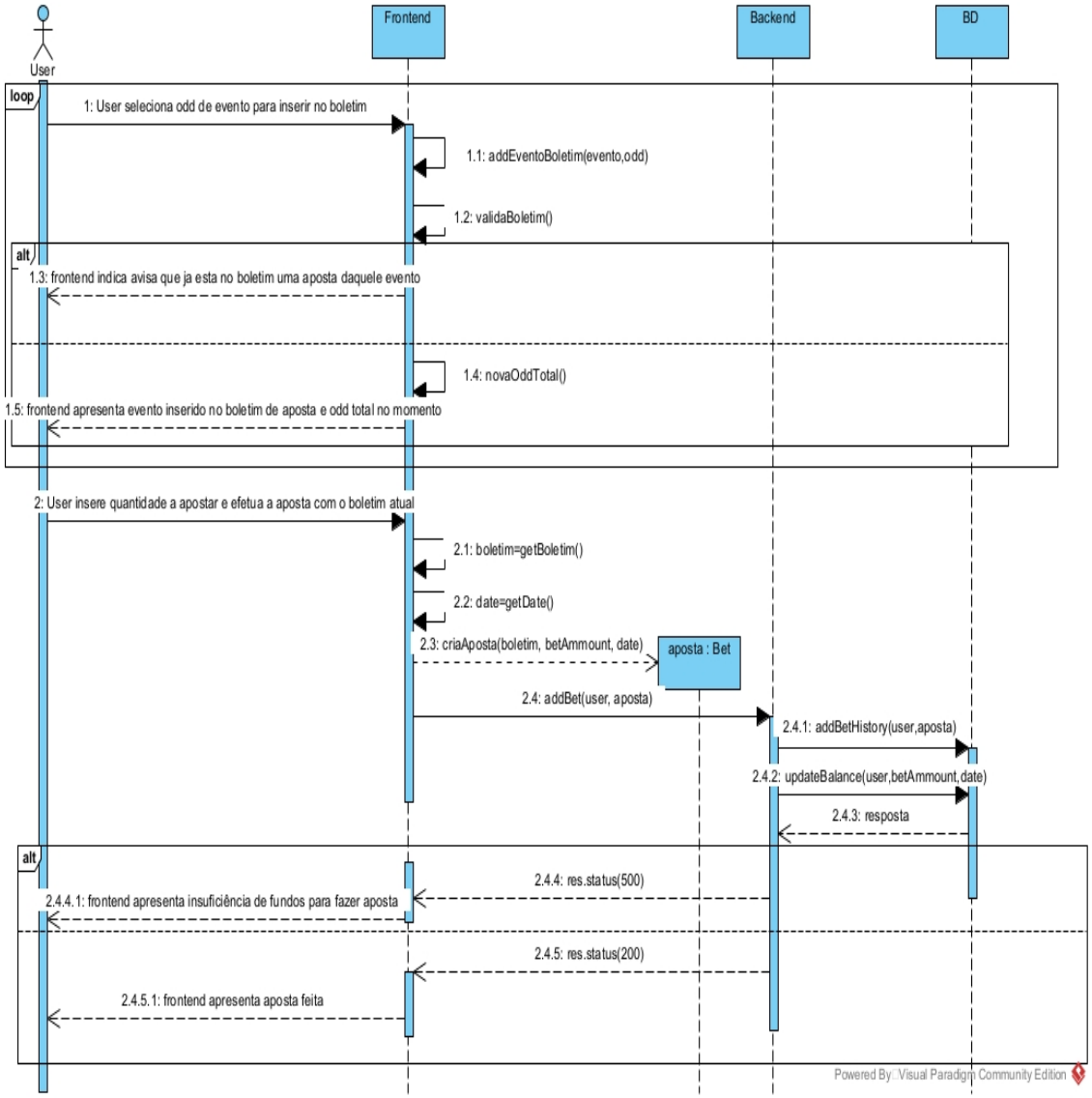


Figura 6.2: Tratamento de aposta

6.5 Gestão de Eventos

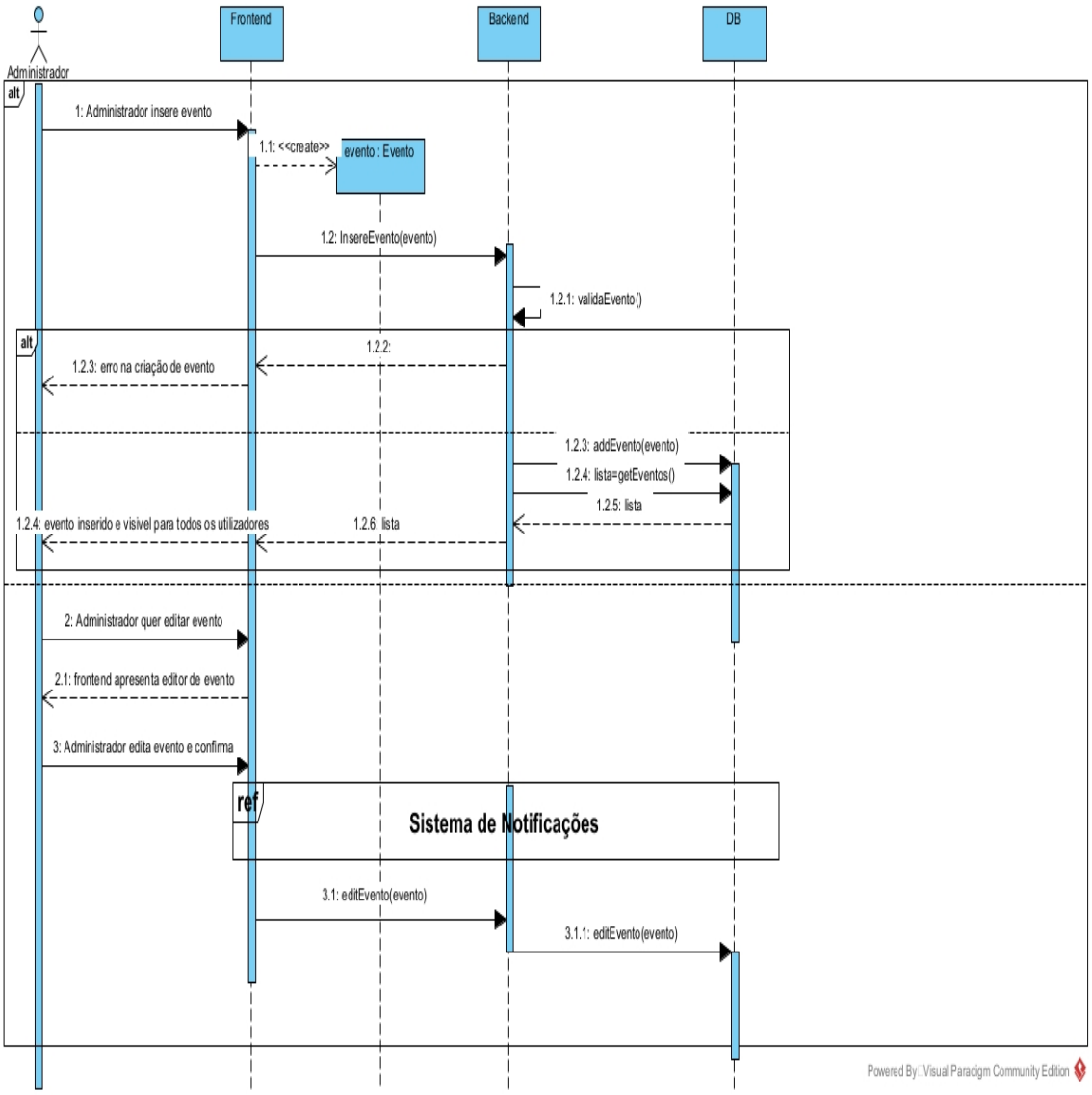


Figura 6.3: Gestão de Eventos

Este diagrama referencia o diagrama seguinte que aborda o aspeto das notificações para o utilizador.

6.6 Sistema de notificações

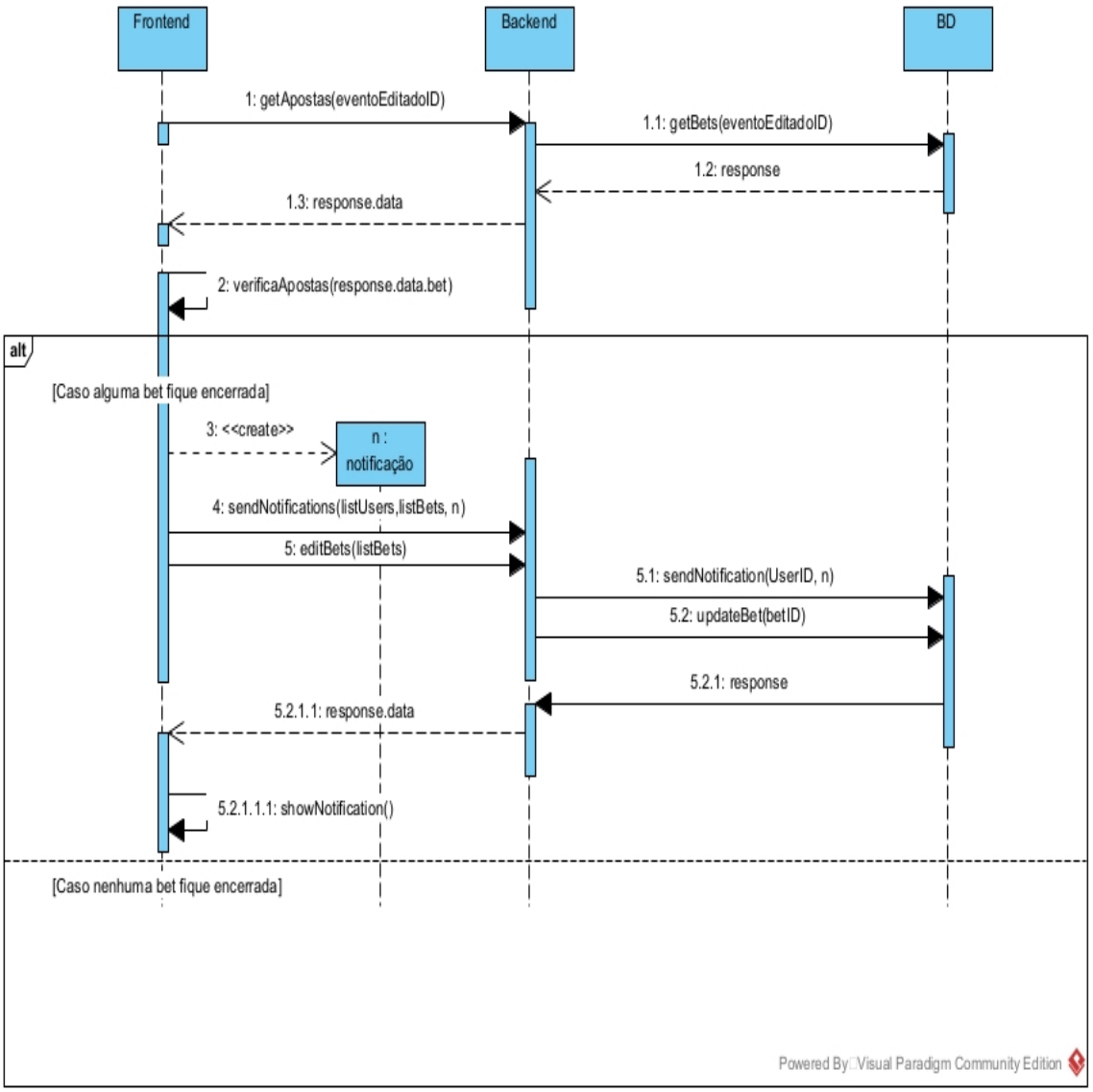


Figura 6.4: Sistema de Notificações

7 Crosscutting Concepts

7.1 Content

Nesta secção, será feita uma descrição das principais ideias de soluções que podem abranger e serem importantes para várias partes do sistema, estando portanto estes conceitos relacionados com vários *Building Blocks*.

É de salientar a elevada importância dos conceitos presentes nesta secção que formam a base da integridade conceptual da arquitetura do sistema e que não podendo ser atríbuidos a componentes individuais necessitam de uma explicação mais coesa.

7.2 Domain concepts

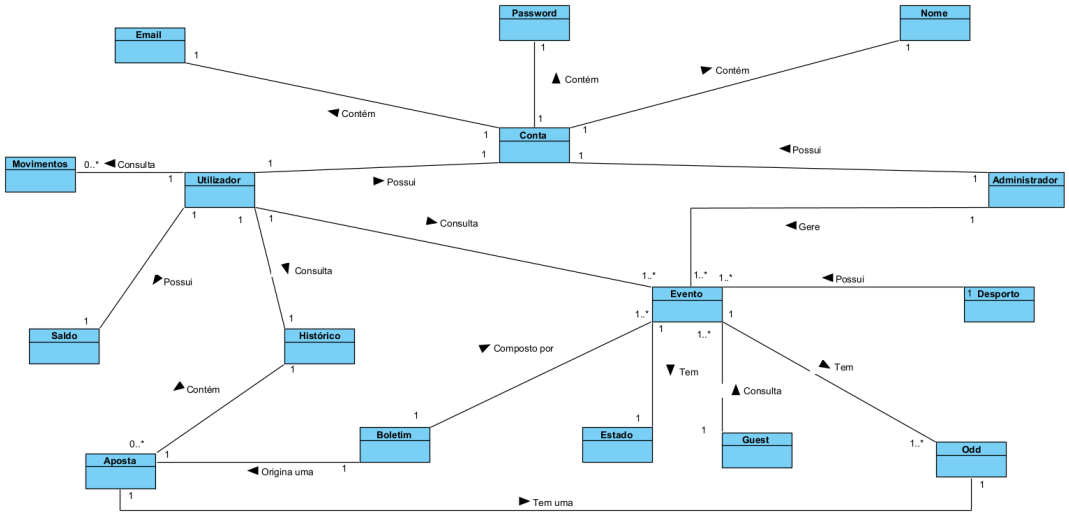


Figura 7.1: Modelo de Domínio

7.3 User Experience concepts (UX)

7.3.1 User Interface

Relativamente ao interface a ser desenvolvido é importante ter em conta os requisitos não funcionais REQN001, REQN002, REQN003 e REQN004 presentes nas secções **4.1**, **4.2** e **4.3** do relatório anterior.

7.4 Safety and security concepts

7.4.1 Security

Sobre conceitos de segurança é importante salientar a privacidade de cada utilizador bem como todas as suas informações pessoais e monetárias. Os requisitos a ter em conta nesta parte são os REQN011, REQN012 e REQN013 presentes na secção **4.6** do relatório anterior.

Com objetivo de alcançar estes objetivos, pretende-se desenvolver uma sistema de registo e autenticação de utilizadores.

7.4.2 Architecture and design patterns

O principal fator que contribuiu para a arquitetura do sistema foi a decisão de desenvolver uma plataforma web, o que implica a estrutura habitual de *Frontend* e *Backend* com uma ligação a uma base de dados, neste caso pretende-se utilizar uma base da dados não relacional e por isso optamos pelo MongoDB.

Além desta decisão que afetou a decisão da arquitetura do sistema, usamos também um *Design pattern*, o **Observer Pattern**. Este padrão foi utilizado para atingir o objetivo de enviar notificações ao utilizador relativas ás suas apostas feitas e o seu funcionamento é representado no diagrama de sequência presente na secção **6.0.6** deste relatório.

7.5 "Under-the-hood" concepts

7.5.1 Persistency

De forma a garantir a persistência de dados e tal como referido na secção anterior, decidiu-se utilizar uma base de dados não relacional em *Mongo*.

Para aceder á base de dados o backend utiliza uma livraria designada de *mongoose*, que permite uma ligação entre o servidor e a base de dados e a obtenção e manipulação de dados.

7.5.2 Session Handling

Após a autenticação de um utilizador é gerado um token que é válido para a sessão desse utilizador e permite navegar e usufruir das funcionalidades do sistema como um utilizador autenticado.

O token é gerado recorrendo á livraria *JWT* que através de um segredo consegue não só gerar mas descodificar um token recebido e garantir que de facto se trata de um token válido.

7.5.3 Plausibility checks and validation

A validação dos dados inseridos é sempre feita no *frontend* onde são definidas um conjunto de regras para cada tipo de dado.

8 Architectural Decisions

8.1 Interface Simples e Intuitiva

De modo a permitir que qualquer tipo de Ator no Sistema tenha a melhor experiência de navegabilidade possível, é necessário que a implementação de toda a Interface seja feita com certas características em mente, sendo estas:

- Simplicidade
- Navegabilidade Intuitiva
- Interface Apelativa

Para atingir estes objetivos, foi decidido que em termos arquiteturais, esta Interface será desenvolvida em Vue, uma framework de Javascript para o desenvolvimento de Interfaces para SPA's (Single Page Applications), e com recurso ao Vuetify, uma framework que disponibiliza inúmeras funcionalidades para o desenvolvimento gráfico, apresentando várias soluções já implementadas para cada componente que irá ter que ser implementado no nosso Sistema.

Com recurso às ferramentas apresentadas, iremos garantir que durante a utilização do nosso Sistema, o Utilizador conseguirá sempre perceber para onde navegar para completar as ações que pretende realizar, assim como garantir que toda a Interface é agradável aos olhos do mesmo, fazendo com que toda a experiência de utilização da Plataforma seja melhorada.

8.2 Consistência de Dados e Velocidade de Execução de Tarefas

Para garantir que todos os Atores do Sistema conseguem ter acesso a informação atualizada em tempo real, é necessário garantir que toda a comunicação entre os *Building Blocks* da Arquitetura é feita de forma rápida, eficiente e sem qualquer perda de dados.

De modo a implementar tudo isto, a comunicação entre Frontend e Backend e a comunicação entre Backend e Base de Dados, deverá recorrer a ferramentas que garantem a integridade e velocidade nos momentos de transferência de dados/pedidos de um componente para o outro. No caso da comunicação entre Frontend e Backend, o Axios irá garantir que, todos os pedidos e respostas HTTP recebidos/efetuados por ambos, irão conter toda a informação necessária para dar continuidade a todo o processo de vida de uma ação do Sistema assim como garantir a sua velocidade de entrega e

integridade. Isto é conseguido uma vez que o Axios é um cliente HTTP baseado em "promises", que envia mensagens HTTP assíncronas para os "endpoints"para os quais é necessário enviar essas mesmas mensagens.

Por outro lado, o "Mongoose"irá permitir modelar a comunicação efetuada entre o Backend e a Base de Dados, para que mais uma vez, seja garantido que todos os dados se mantêm integros e que toda a comunicação é efetuada de forma rápida.

9 Quality Requirements

Nesta secção iremos referir os objetivos obtidos dentro do estudo dos requisitos não funcionais (destacados na secção **4** *Non-Functional Requeriments*) que não foram destacados na secção **1.2** *Quality Goals*. Requisitos esses que são *Appearance*,*Maintenance and Support*, *Cultural and Political* e *Legal*.

Quality	**Motivation**
Appearance	O sistema terá de estar desenhado de forma a captar a atenção dos clientes e com as cores usadas pela empresa (vermelho, azul e branco).
Maintenance and Support	O sistema terá de estar com o codigo bem documentado para que qualquer pessoa possa perceber como funciona e o seu objetivo, como também seja fácil adicionar qualquer funcionalidade nova.
Cultural and Political	O sistema deverá estar em Português de portugal e a moeda de troca será o euro.
Legal	O sistema deverá garantir autorização de registo a apenas maiores de 18 e estará de acordo com a legislação em vigor sobre apostas online.

Tabela 9.1: *Quality Requirements*

9.1 Quality Tree

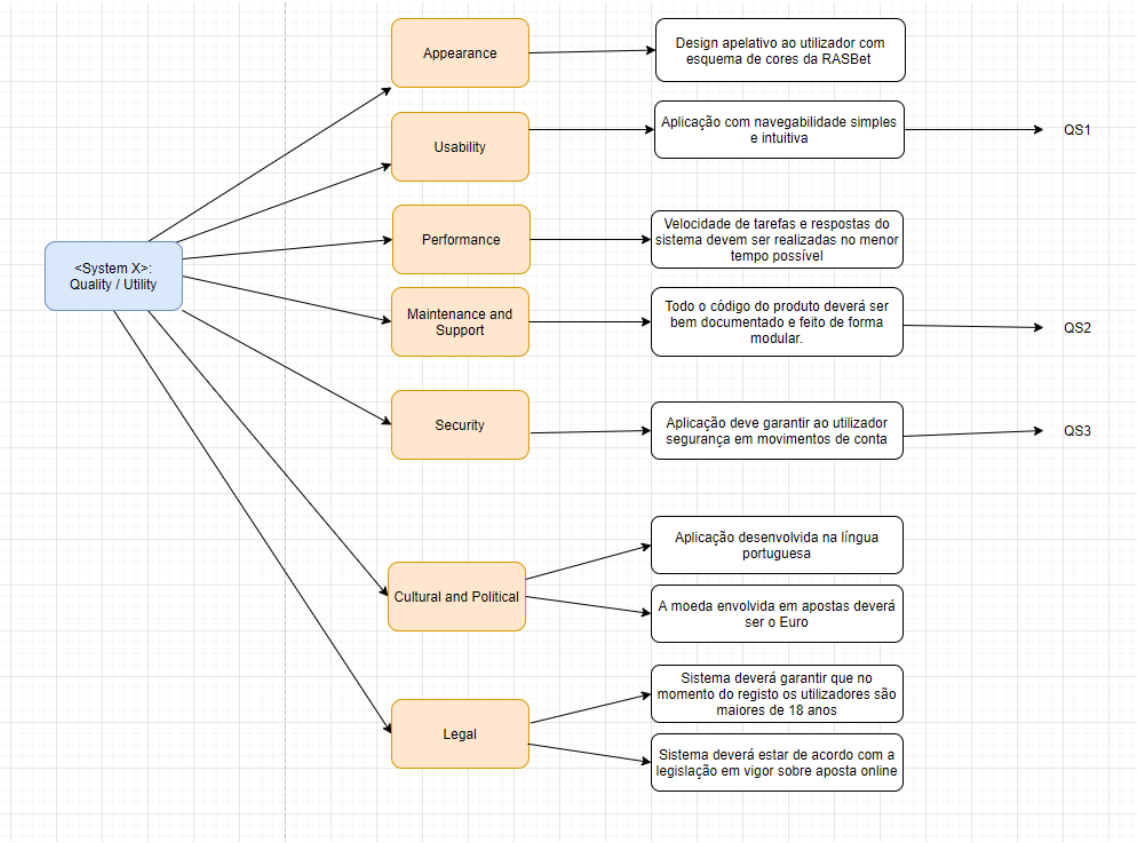


Figura 9.1: Quality Tree

9.2 Quality Scenarios

ID	Scenario
QS1	Um utilizador que não conhece o sistema pode operá-lo sem nenhuma experiência anterior.
QS2	O administrador deve conseguir adicionar um desporto ou evento de forma fácil.
QS3	Quando o utilizador acede aos detalhes de conta apenas pode ver a sua conta

Tabela 9.2: *Quality Scenarios*

10 Risk and technical debts

Nesta secção identifica-se os problemas técnicos que se podem desenvolver não só da arquitectura construida pela equipa como de fatores externos associados à aplicação. Podemos então separar estes riscos em Riscos internos e externos.

10.1 Risco internos

Nesta categoria encontramos riscos associados à arquitetura desenvolvida, ao próprio software e à sua manutenção:

Riscos	**Detalhe**
Availability risk	Caso exista excesso de flow de pedidos ao sistema por parte de utilizadores, pode tornar a resposta vinda do servidor lenta, sinónimo de perda de lucro.
Complexity risks	Interface pode ficar excecionalmente complexa ou difícil de interpretar devido ao uso de *frameworks* e estruturas de dados esotéricas

Tabela 10.1: Riscos internos

10.2 Risco externos

Embora seja referido ao longo dos dois relatórios um sistema externo responsável por toda as transações, nesta categoria encontramos riscos associados às mudanças que podem ocorrer (desconhecidas pela equipa) nesse sistema com repercussões à nossa aplicação:

Riscos	Detalhe
Volatility Risk	Probabilidade de interfaces externas mudarem e dessas mudanças terem impacto no modo como o sistema da nossa aplicação se ajusta e funciona. Valores e dados provenientes desse sistema tem de ser aceites e manuseados com alta flexibilidade pela nossa aplicação.
Accessibility Risk	Caso mudanças de interfaces externas causem impedimento de uso do sistema desenvolvido.

Tabela 10.2: Riscos Externos