TQS: Product specification report

***Marcel de Araújo Santos Souza 101043***

v2023-05-18

[1 Introduction 1](#_Toc39943375)

[1.1 Overview of the project 1](#_Toc39943376)

[1.2 Limitations 1](#_Toc39943377)

[2 Product concept 2](#_Toc39943378)

[2.1 Vision statement 2](#_Toc39943379)

[2.2 Personas 2](#_Toc39943380)

[2.3 Main scenarios 2](#_Toc39943381)

[2.4 Project epics and priorities 2](#_Toc39943382)

[3 Domain model 2](#_Toc39943383)

[4 Architecture notebook 3](#_Toc39943384)

[4.1 Key requirements and constrains 3](#_Toc39943385)

[4.2 Architetural view 3](#_Toc39943386)

[4.3 Deployment architeture 4](#_Toc39943387)

[5 API for developers 4](#_Toc39943388)

[6 References and resources 4](#_Toc39943389)

# Introduction

## Overview of the project

O objetivo do presente trabalho é apresenta, em forma de um produto, todos os conceitos aprendidos na cadeira de TQS, tais como diversos tipos de teste, CI e CD. Além disso, são utilizados ocnhecimentos da cadeira de IES para a construção de API em Java Springboot e utilização de docker para a conteinerização do produto.

A aplicação SpringZoom tem como objetivo apresentar uma app aos moldes da aplicação de chamadas de video Zoom, utilizando no entanto React e SpringBoot para realizar tal feito. Com SpringZoomos utilizadores podem adicionar novos contatos e marcar novas reuniões na plataforma. Além disso, podem aceder todas as informações das meetings marcadas e editar as informações das mesmas.

## Limitations

Dentre as limitações, podemos citar a falta de um Continuos Deployment para a aplicação e a falta do filtro para meetings entre um determinado range de datas. Além disso, por dificuldade em trabaçhar com SonarQube Cloud e o Gradle, a porcentagem de teste realizado na aplicação (que é fornecido pelo SonarQube) foi preojudicado e não consegui implementar.

# Product concept

## Vision statement

A aplicação SpringZoom pode ser utilizada para marcar, editar, deletar e visualizar dados de reuniões em vídeo-chamada. Em resumo, resolve o problema de comunização entre 2 personas que desejem realizar uma ligação por vídeo. Além disso, permite o registo e login dos utilizadores, evitando que qualquer utilizador tenha acesso aos dados dereuniões das quais não fazem parte. Também é possível adicionar contatos à um determinado utilizador registado, evitando que um utilizador marque uma reunião com um outro que não esteja nos teus contatos.

<optional: you may include a UML Use Case diagram to support the explanation>

## Personas and scenarios

Persona 1: Caio Bruno

Caio Bruno é um homem, médico, imigrante brasileiro em Portugal e tem 40 anos e 1 filha, que ainda vive no Brasil com sua ex-mulher. Comoele está morando em Portugal desde 2019, sente falta de falar com sua filha e de vê-la, portanto, decide utilizar o serviço do SpringZoom para marcar uma conversa por vídeo com ela. Como a sua filha, Júlia, ainda é bem pequena, pede para que sua ex-mulher Reanata se registe na plataforma SpringZoom.

Após Renata se registar, Caio bruno adiciona o contato dela na sua lista pessoal de contatos do SpringZoom e marca a conversa para o dia 28/07 as 15:00. Renatta então percebe que na sua página principal do SpringZoom aparece esta nova reunião. No dia determinado, Renata coloca a filha do casal em frente ao computador e Júlia e Caio podem enfim conversar por vídeo.

Persona 2: Raffaelle Bua

Raffaele é um homem de 34 anos e trabalha como recrutador de programadores para a empresa italiana LeafSpace. Como as reuniões de contratação são todas feitas de forma remota e a empresa de Raffaelle decide não gastar mais dinheiro com os planos ilimitados das empresas que fornecem este serviço (Microsoft Teams, Skype, Zoom), ele decide migrar as reuniões para a plataforma SpringZoom.

Para cada novo candidato para a Leaf Space, Rafaelle pede para que eles se registem na plataforma e, logo depois, adiciona eles à sua própria lista de contatos. Após isto, Raffaelle cria uma nova meeting, que aparece na página inicial do candidato na plataforma. Caso o candidato não se sinta confortável com o horário que Raffaelle selecionou, pode editar o horário da reunião e assim, na página inicial de Raffaelle na plataforma, a mudança será visualizada e aprovada ou não por ele.

## Project epics and priorities

**[**Apresentar um plano indicativo para a implementação incremental da solução ao longo de várias iterações/releases, explicando as funcionalidades a atingir por [*epics*](https://www.atlassian.com/agile/project-management/epics-stories-themes) ]

O projeto pode ser extendido por diversas iterações posteriores. No quesito segurança e funcionalidade, algumas coisas ainda não foram completamente desenvolvidas ou necessitam de um refinamento antes do projeto ir ao público de forma mais robusta. Desta forma, abaixo seguem alguns epics:

- Epic 1: Notificação de Meeting

Funcionalidade:

* Notificar os usuários envolvidos em uma meeting de que a meeting está para iniciar;
* Visualizar a notificação e ao clicar ir diretamente para a reunião;

- Epic 2: Implementar o servidor de vídeo chamadas

Funcionalidade:

* Ao entrar na sala de uma meeting agendada, iniciar a vídeo chamada utilizando algum serviço terceiro para tal;
* Assegurar a segurança dos dados dos utilizadores para com o serviço terceiro;

- Epic 1: Permitir mais de 2 utilizadores em uma chamada;

Funcionalidade:

* Permitir o organizador inicial da meeting criar um link para enviar à outros participantes;
* Todos os participantes acessam a room da meeting e iniciam em sincronia a vídeo chamada;

# Domain model

<which information concepts will be managed in this domain? How are they related?>

<use a logical model (UML classes) to explain the concepts of the domain and their attributes>

# Architecture notebook

## Key requirements and constrains

<**Identify issues that will drive the choices for the architecture** such as: Will the system be driven by complex deployment concerns, adapting to legacy systems, or performance issues? Does it need to be robust for long-term maintenance?

Identify critical issues that must be addressed by the architecture, such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions? Are there integrations with external systems? Is the system to be offered in different user-interfacing platforms (web, mobile devices, big screens,…)?

E.g.: (the references cited in [XX ] would be hypothetical links to previous specification documents/deliverables )

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

* The existing legacy Course Catalog System at Wylie College must be accessed to retrieve all course information for the current semester. The C-Registration System must support the data formats and DBMS of the legacy Course Catalog System [E2].
* The existing legacy Billing System at Wylie College must be interfaced with to support billing of students. This interface is defined in the Course Billing Interface Specification [E1].
* All student, professor, and Registrar functionality must be available from both local campus PCs and remote PCs with internet dial up connections.
* The C-Registration System must ensure complete protection of data from unauthorized access. All remote accesses are subject to user identification and password control.
* The C-Registration System will be implemented as a client-server system. The client portion resides on PCs and the server portion must operate on the Wylie College UNIX Server. [E2]
* All performance and loading requirements, as stipulated in the Vision Document [E2] and the Supplementary Specification [15], must be taken into consideration as the architecture is being developed.>

## Architetural view

→ Discuss architecture planned for the software solution.

→ include a diagram ( a package or component diagram)

<detail the specific technologies/frameworks that were used>

→ explain how the identified modules will interact. Use sequence diagrams to clarify the interactions along time, when needed

→ dicuss more advanced app design issues: integration with Internet-based external services, data synchronization strategy, distributed workflows, push notifications mechanism, distribution of updates to distributed devices, etc.>

## Deployment architecture

DEPLOY NÃO REALIZADO

[Explicar a organização prevista da solução em termos configuração de produção (*deployment*). Modelar num diagrama de *deployment*]

# API for developers

Toda a documentação está acessível por meio do URL do swagger. Os desenvolvedores que acederem este URL, podem também testar e visualizar as respostas de todos os endpoints da plataforma, tais como os erros que são tratados. Em geral, as respostas para os endpoints são retornados em formato JSON.

URL: localhost:8080/swagger-ui/index.html



# References and resources

* MySQLDriver for Java SpringBoot
* ReactJS
* BootStrap
* Axios
* Docker
* Github Actions
* MySql