TQS: Product specification report

***Ana Alexandra Antunes [876543]*, *Bruno Bernardes [876543]***

v2023-04-18

[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)

[This report should be written as the main source of technical documentation on the project, clarifying the functional scope and architectural choices. Provide concise, but informative content, allowing other software engineers to understand the product and quickly access the related resources.

Tips on the expected content, along the document, are meant to be removed.

You may use English or Portuguese; do not mix.]

# Introduction

## Overview of the project

<contextualize the objectives of this project assignment in the scope of the TQS course>

<introduce your application/product: brief overview of the solution. What is it good for? Introduce the name of the product if it has one>

## Limitations

 <explain the known limitations/unimplemented (but planned) features>

# Product concept and requirements

## Vision statement

<functional (black-box) description of the application: what will you system be used for? Which is the high-level/business problem being solved by your system?>

<if needed, clarify what was planned/expected to be included but was changed to a different approach/concept >

<optional: how is your system different or similar to other well-known products?>

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

<optional: additional details on the process for the requirements gathering and selection (how did we developed the concept? Who helped us with the requirements? etc)>

## Personas and scenarios

<Uma Persona é uma personagem utilizada para contar histórias representativas da futura utilização do sistema. Uma Persona é um Actor instanciado, à qual se dá um conjunto de caraterísticas para a humanizar e definir o contexto em que usará o sistema e as suas motivações.

“Personas are fictional people. They have names, likenesses, clothes, occupations, families, friends, pets, possessions, and so forth. They have age, gender, ethnicity, educational achievement, and socioeconomic status. They have life stories, goals and tasks. Scenarios can be constructed around personas, but the personas come first. They are not ‘agents’ or ‘actors’ in a script, they are people. Photographs of the personas and their workplaces are created and displayed. […] It is to obtain a more powerful level of identification and engagement that enable design, development, and testing to move forward more effectively”. Adapted from Grudin, J. and Pruitt, J., 2002, June. Personas, participatory design and product development: An infrastructure for engagement. In Proc. PDC (Vol. 2).

Exemplo: ver [secção 4.1, neste artigo](http://www.mdpi.com/1424-8220/18/4/1285) (open access)] >

<You don’t need to include all possible details. Pick the main scenarios, related to the core value of the system.>

<The scenarios tell the story of the Personas in their lives, doing their daily/professional activities that are relevant to find the points of contact with the system under specification.

Scenarios are somewhat similar to use cases (they have a goal and tell a story), but, unlike use cases, they capture a larger process, with activities that may not use the software. Scenarios don’t required a “template”, like the usual use cases description.>

Exemplo: ver [secção 4.2 neste artigo](http://www.mdpi.com/1424-8220/18/4/1285) (open access)] >

## 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) ]

# 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.>

## Architecture view

→ Discuss architecture planned for the software solution.

→ include a diagram ( a package or block diagram)

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

→ discuss 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

[Explicar a organização prevista da solução em termos configuração de produção (*deployment*). Anotar, no diagrama, as tecnologias de implementação, e.g.: colo aro simbolo do PostgreSQL na Base de dados,…]

# API for developers

[Explicar a organização da API. Os detalhes detalhes/documentação dos métodos devem ficar numa solução *hosted* de documentação de APIs, como o [Swagger](https://swagger.io), Postman documentation, ou incluída no próprio desenvolvimento (e.g.: maven site)

<what services/resources can a developer obtain from your REST-API?>

<document the support endpoints>



# References and resources

<document the key components (e.g.: libraries, web services) or key references (e.g.: blog post) used that were really helpful and certainly would help other students pursuing a similar work>