G2

Logística
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
10/10/2017	1.0		Orlando y Ormachea

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

Along the document we aim to show all the architectural views of the system, and be able to provide our client the decisions made regarding software architecture.

1.3 Definitions, Acronyms, and Abbreviations

Some definitions used in this document are:

- class: an extensible program-code-template for creating objects, providing initial values for state and implementations of behavior.
- design model: portraits the classes and methods included in each one of them, providing a general idea of the design of the Object-Oriented Programming design.

1.4 References

The only reference made in this document is the one to the design model.

1.5 Overview

Throughout the rest of the document we will cover our architectural design choices, the main goals aimed, a broad overview of each class' structure and a brief description of how the deployment and implementation will take place.

2. Architectural Representation

This architecture´s design consists of four classes. The first one is the user, which can be classified into two types: Business User, which is the user that offers the transportation services, and Client User, which is the user that wants to transport something. The other three classes are Transport, which contains the information of the things to be transported, Service, which refers to the characteristics of the offered transport and Admin, which is used just to control the app and maintain it .

3. Architectural Goals and Constraints

The goals of the architectural design are pretty self explanatory. As an app that works as a mean of communication between two users, the design is trivial, just the parties involved and the objects that they each offer.

4. Use-Case View

The architectural design is entirely based on the use-case model. Every behavior from the classes tries to cover some use-case. Therefore it would be repetitive to list all the use-cases here.

5. Logical View

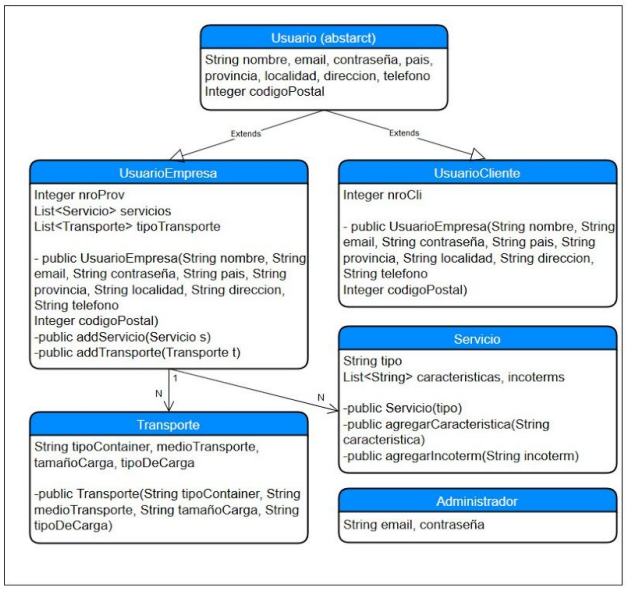
5.1 Overview

The design model has two packages, the view and the backend one. As there are five classes in the backend and ten in the backend since it is the most complete package.

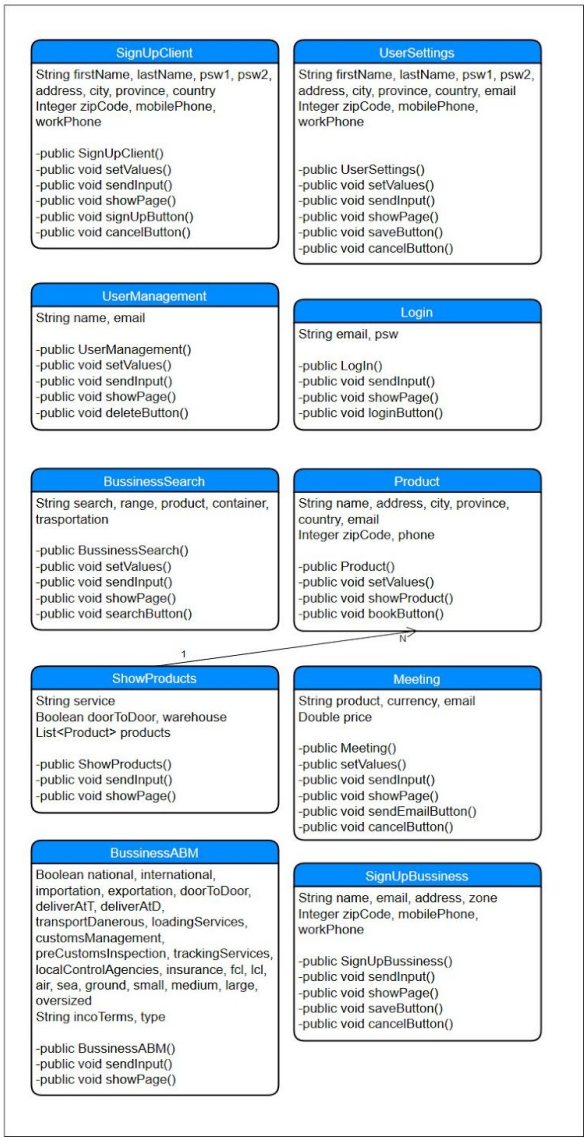
5.2 Architecturally Significant Design Packages

There are two packages, one for the view and another one for the backend.

Model



View



5.3 Use-Case Realizations

Let's exemplify with use-case number 9: The user now becomes a business user. A specific id is created for him (nroProv) and all his personal data is stored in the user variables while all the business-relevant data is stored in the “servicios” list and the “tipoTransporte” list. Now the business user is created with class constructor. Moreover, when the business user enters information about its transports as well as its services, each transport and service should be created accordingly if it doesn't already exist, with its constructors.

6. Process View

Regarding the different processes involved in the system, there is not any difference between each of their complexities. As the system is only intending to provide a platform of communication between clients and service providers, and providing structure for the information exchanged to simplify their business relation, both processes require almost the same amount of processing.

7. Deployment View

The configuration is simple. Just one server containing the entire database and which connects to app through the cloud, where it receives the new information to store, stores it and then deletes the info from the cloud.

8. Implementation View

8.1 Overview

8.2 Layers

9. Data View (optional)

All the data will be stored in a single database.

10. Size and Performance

The size of this implementation should be quite small as there are not many classes to handle. Therefore performance should be optimal.

11. Quality

The software architecture is the key of the software. Without it, everything would be messed up and intercommunication within processes would be nearly impossible.