INTEGRATIVE PROJECT OF THE 4TH SEMESTER OF LEI-ISEP

2021-2022 (version II.b)

PART II – System Specification

1 Context

In the 2021-2022 academic year, the fourth semester (i.e. 2nd year, 2nd semester) of the Degree in Informatics Engineering (LEI) of the Instituto Superior de Engenharia do Porto (ISEP) adopts a teaching-learning process based on the development of a single project that enhances the integration and application of knowledge, skills and competencies of all course units taught through the semester: Applications Engineering (EAPLI), Laboratory and Project IV (LAPR4), Languages and Programming (LPROG), Computer Networks (RCOMP) and Computer Systems (SCOMP).

The project, common to all course units, consists of developing the system described in this document in accordance with the general procedures described earlier in the document (written in Portuguese) with the same designation, whose subtitle is "PARTE I – Descrição de Funcionamento".

2 System Description

The ***Smart Systems for Industry (SS4I)***[1] is an IT company specialized in the development of systems/applications able to exploit the most recent technological advances that are leading to the ongoing industrial revolution known as Industry 4.0. In short, the Industry 4.0 denotes the shift to the digitalization and automation of every kind of processes to reduce organizations' dependence on human labor and decision-making and, therefore, minimizing human error, increasing their productivity and shorten time to market. It is worth noticing that these processes might range from manufacturing processes to more high-level processes such as buying/selling/delivering products and/or services.

Within this scope, and challenged by an important client, *SS4I* aims to develop a products orders (and sell) management system (from now on referred to as SPOMS) having in mind two distinctive aspects:

- The products being sold are stored in a (almost) fully robotized warehouse where customized automated guided vehicles (AGV) are used to pick-up orders' products.
- The ability to specify and carry out questionnaires/surveys about any intended topic (e.g.: about a concrete product, a brand, the customer satisfaction regarding the system and/or regarding the delivery process) in an easy and integrated way.

[1] Fictional company.

Taking these objectives into account, throughout this project it is intended to develop a functional prototype that (i) allows validating the main ideas exposed and (ii) constitutes, by itself, the pillars of what the end system can be.

> **Note:** The system described next is a big simplification of what a products order management system traditionally is. Simplifications are made, on one hand, to turn the project feasible within the scope of the LEI semester and, on other hand, to focus on some innovative aspects. As so, you should pay attention to the simplifications and assumptions presented throughout the system description.

2.1 Overview

As in any traditional products order management system, there is a minimum core information regarding customers and the products catalog that is required to the system work properly. Thus, it is important that SPOMS provides the users with the ability of either (i) to create and manage products/customers information directly and (ii) to import such information from external systems in use in the organization where the system is being deployed.

Products basic information comprehends a unique internal code, a short and an extended description as well as a more technical description, a set of photos, a brand name and reference. Moreover, products can be organized by categories. By simplicity, a category consists only of an alphanumeric code, and a description. Each product belongs mandatorily to a single category. To foster integration with other information systems, a product also has (i) a mandatory barcode according to a valid coding standard such as, for instance, EAN-13, UPC and (ii) an optional production code. Since products' prices vary through time (e.g.: promotions) and by applicable taxes that differ from one location/country to another, by now, it is just worth considering the current list price with and without taxes. At last, it is required a mechanism to inhibit products (one or more) from being temporarily (or permanently) ordered.

Regarding customers, the minimum required information is its name, a valid Value-Added Tax (VAT) identifier, an email address, and a phone number. Optionally, customers might state their birthdate and gender and have/manage several billing and delivering postal addresses. These addresses should be kept updated by customers during the ordering process, namely while they are concluding a new order. Every customer also has a unique identifier that is automatically generated by the system at the customer registration time. In this regard, it is important noticing that a customer might register his/herself on the system or, instead, be registered by an organization salesclerk, which may type or import customer' information from another system running into the organization. In the former scenario, the customer must undergo an email/phone-based validation process to get his/her credentials to access the system. In the latter scenario, no customer credentials are generated automatically.

Customers and other kind of users frequently access the system to browse and search for products. As so, a special attention to this feature should be devoted, namely to which concerns its usability. While customers are browsing the products catalog, they should be able to easily add a given product to his/her shopping cart. By default, the added product quantity is one unit. However, the customer

might opt to set more units (no decimals are allowed). The shopping cart content should be preserved from one customer session (e.g.: occurred for instance two days ago) to another (e.g.: occurring now). Moreover, customers should be able to easily convert their shopping cart content into a new products order (or simply order). Items successfully converted on an order should be automatically removed from the shopping cart.

A products order has an automatically set numeric identifier and the datetime it has been registered in the system. It mandatorily refers to a single customer and records the typed/selected billing and delivering address, the ordered products and its quantities and unitary prices as well as the order total amount (before and after taxes). In this respect, it is worth highlighting that products information (e.g.: prices but sometimes also descriptions) change over time. Such changes cannot affect the past orders original information. Besides that, an order also comprehends:

- The shipment method selected (e.g.: Standard, Blue, Green) and the respective cost, which varies in accordance with the selected method, the volume of the products packages and its weight.
- the (automatic) payment method (e.g.: paypal) selected to pay the order should also be recorded together with the payment confirmation received from the respective payment service. Notice that, there is no need/intent to record the concrete payment data used/typed by the customer (e.g.: the paypal credentials). It is enough knowing the method and the confirmation received by the respective payment service.

While developing the system prototype, these two issues (i.e.: shipment and payment) must be considered. Although, since both are not perceived as representing a risk to the project, the shipment cost computation, and the connections to external systems (e.g.: carriers and payment services) should be mock.

At least for some organizations, it is also important that the system allows sales department clerks to create orders on behalf of a given customer. Thus, the system should collect the required information to distinguish between those orders registered directly by the customer and the ones registered by a clerk. In the latter case, due trackability concerns, the system should be able (at least) to identity the respective clerk.

Another relevant aspect the system needs to track properly concerns the several states that every order goes by since it is registered in the system to the moment it is received by the customers. Some possible states that can be considered are: (i) registered, (ii) payment pending, (iii) to be prepared (waiting for an initial warehouse action), (iv) being prepared on the warehouse, (v) ready for packaging; (vi) ready for carrier dispatching, (vii) dispatched, (viii) delivered by carrier, (ix) received by customer. Notice that all relevant states might not be listed and some of the listed states might be redundant and/or not necessary by some reason. As so, this matter should be carefully analyzed.

As stated previously, the SPOMS is being conceived to operate/handle on scenarios where products are stored in highly robotized warehouse(s). This kind of warehouses are typically organized by aisles (or corridors), each aisle is vertically split on rows and each row is horizontally split on/by shelves (cf. Figure 1). Each storage area, defined by a row and a shelf, comprehends multiple similar bins, each one having one product unit inside. The bins size may vary from one storage area to another. Figure 1 depicts the structure of a generic aisle comprehending three rows. The first row has three storage areas since it is split by three shelves. The second row has two storage areas and one of those areas is empty (on top). The third row has a single shelf and therefore, it consists of a single storage area. Distinct bins (represented by colored rectangles) are being used.

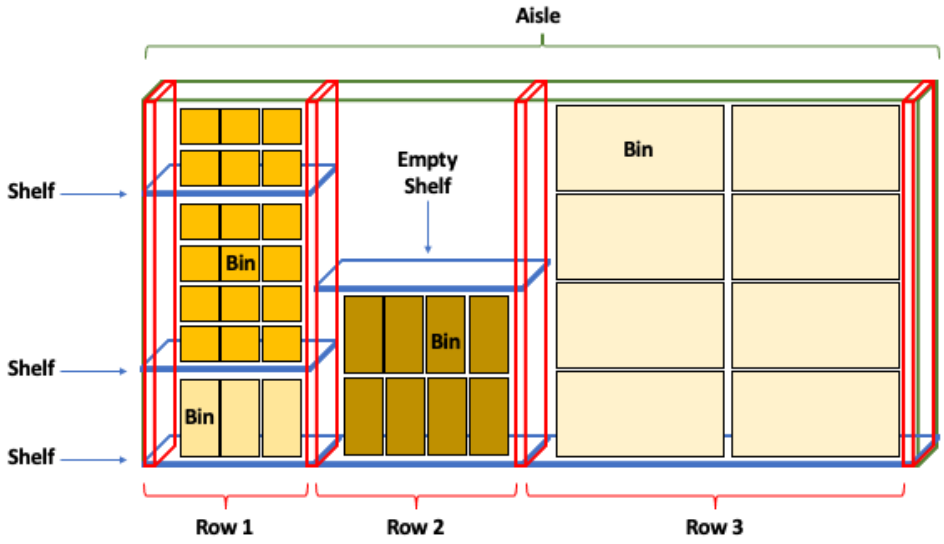

FIGURE 1. AISLE STRUCTURE.

Throughout a warehouse plant this information is provided on multiple places (e.g.: on the floor and walls) and by multiple ways (e.g.: QR codes, signs) for either the operating machines (i.e.: AGVs) and to humans. Despite several warehouses might exist and products be in multiple locations (i.e., storage areas) within the same warehouse, for the prototype development one will consider a single warehouse and a single product location on that warehouse. Even though, some vital information needs to be managed by the SPOMS such as:

- The warehouse plants (check section 5.2 for an example).
- The AGVs operating on the warehouse, its characteristics (e.g.: identifier, short description, model, and maximum weight it can carry) and its base location (i.e., the AGV dock). In addition, it is necessary to know the AGV status regarding its autonomy (e.g.: 2 hours left) and current task (e.g.: free, charging, occupied serving a given order).
- The products' location in the warehouse, which corresponds to a storage area i.e., the aisle identifier, the row identifier, and the shelf identifier. All these identifiers are numeric. Products with an unknown location cannot the ordered.

Any order that has been successfully paid is ready to be prepared by the warehouse. In case an AGV is free, and it has the capacity to carry all the products order (regarding the order total volume and weight), such order can be assigned (as a task) to the AGV. After accepting a task, the AGV tries to complete it autonomously by leaving its base location and moving itself by the warehouse to pick-up the products one by one out of the respective storage areas. When the AGV has collected all the products, it returns to its base location and drops the products on a dedicated output place. At this time, the order status should change from "being prepared" (set when the AGV accepted the task) to "ready for packaging". The process of assigning tasks to AGVs can be done either automatically by the system and/or manually by a warehouse employee. At last, a warehouse employee is responsible to inspect the collected products, package them properly (if necessary) and dispatch the packages for customer delivery (through a carrier).

Concerning AGVs, during this project, there is the aim of developing an AGV digital twin in two consecutive phases:

- First, at an initial phase, it is only necessary to address the interactions regarding tasks acceptance and its completion by mocking somehow the process of picking up products and moving through the warehouse.
- Further, at a later phase, there is the aim to evolve the digital twin to address (by simulation) the challenges (e.g.: avoiding obstacles) brought by having several AGVs moving simultaneously in the warehouse.

So, a dashboard presenting the (current) status of the AGVs as well as their position in the warehouse is envisioned.

A distinctive aspect of the SPOMS concerns its intrinsic ability to conduct surveys about a product or a brand or the customers' satisfaction or even about any other topic that a sales manager intends to. Since, nowadays, this is seen as a common and recurrent feature in a huge variety of systems, the development team is requested to conceive/design/implement a generic base solution that can be easily reused on other systems/domains. In this respect, the generic solution must focus on the following aspects: (i) the specification of a questionnaire to be answered by the target audience; (ii) the ability to generate dynamically (at runtime) a proper user interface (UI) for any given questionnaire; (iii) to validate a given questionnaire response in accordance with the questionnaire specification; and (iv) to provide a statistical report given a set of responses to a given questionnaire. Other aspects such as the data/information persistence and the distribution of the questionnaire by the target audiences can become dependent of the target system (e.g.: SPOMS).

With this context, a questionnaire is a set of questions organized into one or more sections (i.e., a group of questions). Each question is of a certain type (e.g.: free text, multiple choices, ordering/ranking choices) which constraints the information to be requested. For instance, on a "free text" question one might stipulate the maximum number of chars that an answer can have while on a "multiple choices" question one needs to define each possible choice. Yet, it is worth considering that a questionnaire might have mandatory and optional sections/questions and that some sections/questions might become mandatory/optional depending on the answer given to a previous question. At last, it must be allowed that questions of a given section are repeatedly made depending on some condition (e.g.: questions of section X needs to be answered 3 times since "3" is an answer provided previously on another section). More information related with questionnaires is provided on section 5.1.

2.2 System Actors

The system to be developed must be able to deal with different user profiles (e.g.: customer, salesclerk, sales manager, warehouse employee, administrator), and the features available to each user depend on the profile to which they are associated.

Below are the most common profiles and their features:

- Customer: responsible to keep his/her own personal information updated, to create orders and follow their status and answer a survey questionnaire when requested and at his/her willing.
- Sales Clerk: responsible to manage information related with the products catalog, categories, and families as well as register order on customers behalf.
- Sales Manager: in addition to salesclerk responsibilities, (s)he can carry out surveys and view some sales reports.

- Warehouse employee: responsible for managing the warehouse information (e.g.: plants and AGVs) as well as checking and controlling its functioning and by packaging and sending for customer delivering orders.
- Administrator: responsible for managing (e.g.: creating, deactivating) the system users and their respective permissions as well as the general configuration of the system.

3 Use Cases

Having in mind a fast development of a prototype that serves as a proof of concept to the ideas underlying this project, at this moment, it is not intended to spend (many) efforts on functionalities related to the users' authentication and authorization processes and the respective management and/or administration. Thus, the functional areas described below do not include requirements of this nature. After validating the proof of concept, and when it becomes necessary, those features will be explicitly requested.

3.1 Orders Management

Among other functionalities, the orders management area includes:

1. Create a new product category. A category is characterized by an alphanumeric code, a description, and, optionally, a reference to its super-category.
2. Specify a new product. A product specification comprehends multiple attributes (cf. section 2.1).
    a. A product might be manually created by a salesclerk by typing (or selecting) all the mandatory data required to be order by a customer.
    b. Alternatively, products might be imported from a text file generated by an external system. Multiple text file formats need to be supported (e.g.: CSV, XML).
    c. Imported products might have some missing information (e.g.: its warehouse location), which can be further added by the salesclerk.
3. Browse and search the products catalog. With the purpose of ordering and/or as a mean to complete other functionality, several users need to browse and search the products catalog.
    a. It must support filtering and sorting operations by one or more product attributes (e.g.: category, brand, description, status).
    b. The filtered products catalog should be exported to a CSV file.
    c. Add any listed product to the (customer) shopping cart.
4. Register a new customer. A customer registration comprehends multiple attributes (cf. section 2.1). It might be done by three distinct ways:
    a. The customer accesses the application (without being authenticated) and proceeds to his/herself registration. The customer account needs to be further activated, otherwise the customer cannot access the system.
    b. The registration is manually performed by a clerk. No account activation is required.
    c. Alternatively, customers info might be imported from a text file generated by an external system. Multiple text file formats need to be supported (e.g.: CSV, XML).
5. Create credential for a customer. At any time, clerks might request the system to generate access credentials to a given customer access the system. The generated credentials are sent by email to the customer and are never shown to the clerk.
6. Create a new products order. An order can be created:

   a. By a clerk on behalf of a given customer. In this case, the system might request the clerk additional information regarding the source order information was provided (e.g.: phone, email, other).
   b. By the customer using the items that are currently on his/her shopping cart.
7. Get a list of orders with a given status. Details regarding any listed order must be presented at user request. Depending on the selected status some operations should become available:
   a. On orders having a “payment pending” status, it might be possible clerks mark such orders as “paid”.
   b. On orders having a “dispatched for customer delivery” status, it might be possible clerks marks such orders as “being delivered”.
   c. On orders having a “delivered by carrier” status, it might be possible clerks marks such orders as “delivered”.
8. Check the status of my (customer) orders. All the customer orders that have not yet been marked as “delivered” should be presented to the customer.
   a. On orders having a “delivered by carrier” status, it might be possible the customer marks such orders as “delivered”.
   b. Details regarding any listed order must be presented at user request.
9. Get multiple reports such as:
   a. For a given time interval, listing and/or summarizing the quantity and value of orders received.
   b. For a given time interval, reporting the most ordered products considering all categories or just of a given category.
10. Update product information. Occasionally, it is necessary to update the products information, namely regarding:
   a. The ordering prices.
   b. The technical description.
   c. Add/remove photos.
   d. Inhibit a product from being ordered.
   e. (Re)Activate the possibility of a product being ordered.

3.2 Warehouse Management

Among other functionalities, the warehouse management area includes:

1. Setting up the warehouse plant. It comprehends uploading a custom JSON file depicting the plant (or layout) of the robotized warehouse as well as all possible locations (of products and AGVs).
2. Configure an AGV. It comprehends defining the AGV identifier, a short description, its model, the maximum weight, the volume it can carry and its base location. The number of AGVs are limited to the number of base locations existing on the warehouse.
3. Monitoring the warehouse status. It aims presenting a dashboard graphically depicting the layout of the warehouse as well as the location and status of the AGVs. This dashboard must keep updated automatically (e.g.: at every 5 seconds). The updating time must be configured at deployment time.
4. Assign a task/order to an AGV. Any order that has been paid represents a task that can be assigned to an AGV. The task consists of picking up all the items of the respective order. The assignment of a task to an AGV can be done:
   a. Automatically by the system. Currently, a basic FIFO (first in, first out) algorithm is foreseen as acceptable. However, latter on a more complex algorithm might be needed.

   b. Manually by a warehouse employee. In this case, the employee identifies both the intended order and the intended AGV.
   c. Either done automatically or manually the AGV constraints regarding volume and weight must be verified.
5. Mark orders as dispatched for customer delivery. All orders whose products have already been picked up by an AGV are (manually) inspected by a warehouse employee, which after that is responsible for packaging and dispatching those products for customer delivery. At the end, (s)he must inform the system that such work has been done. For that, the system should provide the employee a list of orders on such condition. The ability to filter those order by the AGV that have collected those products is request.
6. Get multiple reports such as:
   a. For a given time interval, summarizing the number of orders attended, i.e.: prepared and dispatched for customer delivery and be able to discriminate such information by AGV and/or employee.
   b. For a given time interval, listing the products dispatched for customer delivery and the respective quantity.

3.3 Surveys

Among other functionalities, the surveys area includes:

1. Create a new survey. A survey is characterized by an alphanumeric code, a description, the period (in days) it will be performed, the intended questionnaire and a set of rules that allows the system to determine the survey target audience. Some examples of those rules are: (i) customers that have ordered a given product during a given period; (ii) customers that have ordered any product of a given brand or category during a given period; customers of a given gender; (iv) customers that fit on a given age group (e.g.: 20-29).
2. View notifications to answer a survey/questionnaire. Customers should be able to get a list of all the notifications received to answer a given survey. A customer receives a notification when (s)he meets the target audience of a given survey.
3. Answer a survey/questionnaire. Following a received notification, customers answer the questions of the requested questionnaire.
4. Consult answers of a surveys. It consists of presenting in raw all the answers received regarding a given survey. It must allow:
   a. To check the details of each answer (i.e.: the question answers given by a customer).
   b. To export those answers to a CSV file.
   c. To get a statistical report about each question. E.g.: On question 2, option “A” was given by 37% of the customers, option “B” by 51% and option “C” by the remaining 12%.

3.4 System Administration Area

The system administration area manages users and their permissions.

1. Login/Logout/Change Password: login and logout and password change.
2. Creation of users: addition of new users and the respective roles. The same user can play more than one role within the organization. It should also be noted that users can represent people or external systems. Unlike the former, the latter do not support any interaction through a User Interface (UI).

3. Activate/Deactivate users. When a user is activated/deactivated, it is necessary to inform the user by email and register the reason in question. Possible reasons are pre-defined.
4. Resetting a user's password: it involves generating a new password randomly and sending it by email to the respective user.
5. List users: the list must distinguish between (i) active and non-active users and (ii) user accounts representing human beings and the ones representing external systems.

Regarding these features, it is foreseen that each team would have a great benefit by reutilizing the framework provided by the EAPLI course unit. As so, its adoption is recommended, but not mandatory.

4 Envisioned Applications

Considering the use cases described and aiming to guide the development of the intended prototype system, it is recommended that it comprises (at least) the set of applications and/or components represented in the Figure 2.

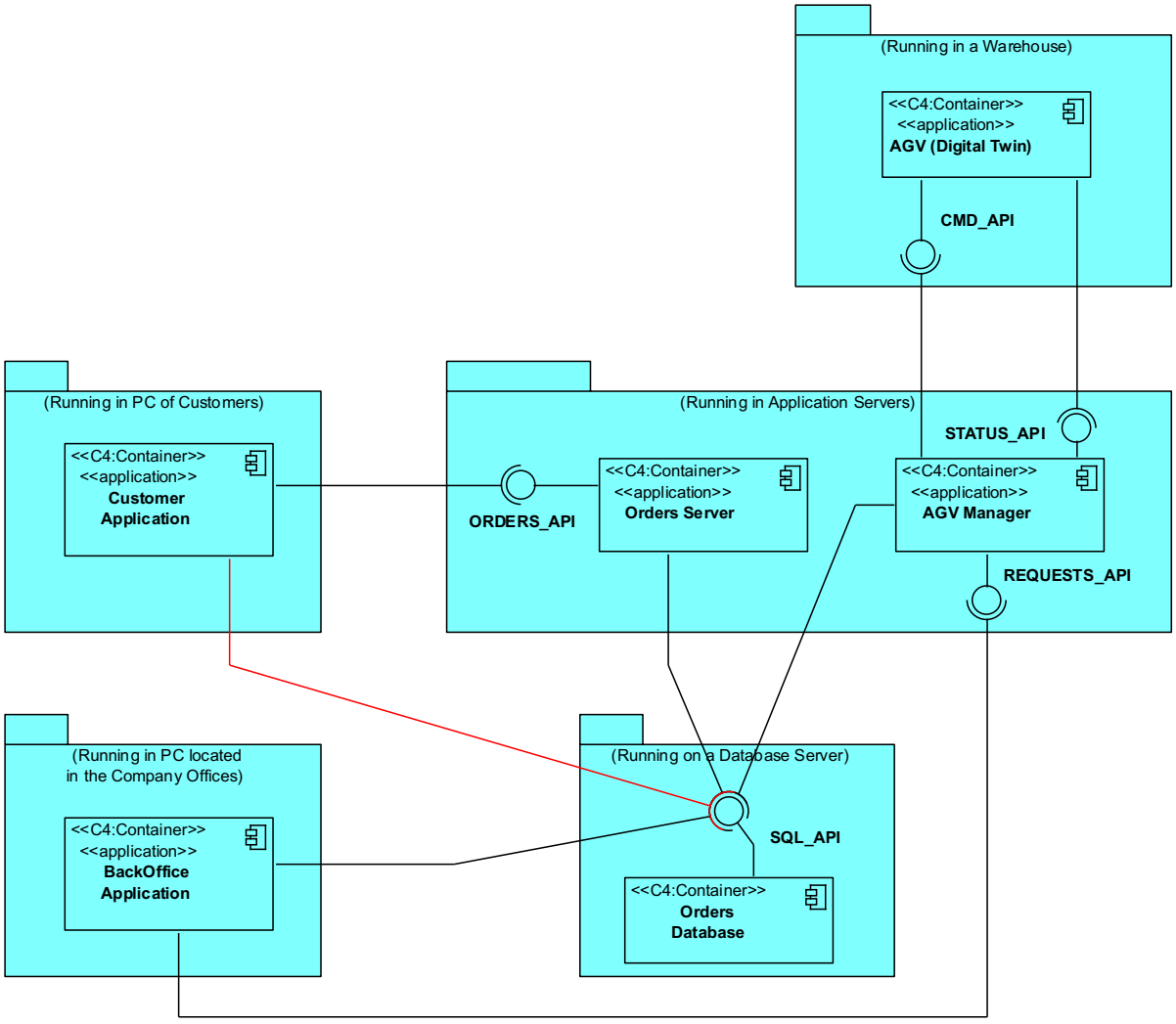

FIGURE 2. SET OF ENVISIONED APPLICATIONS.

As shown, the prototype system consists of:

- A relational database, called “Orders Database”, responsible for the centralized persistence of all data/information handled/managed by the system and for the respective access to it. The system must be prepared to easily adopt different Relational Database Management Systems (RDBMS), such as MySQL, MS SQL Server, and Oracle to host this database.
- A console application, called “Backoffice Application”, suitable for (i) the Salesclerk, (ii) the Sales Manager and (iii) the Warehouse Employee carry out their functions. This application will only be installed/running on computers located in and fully managed by the company.
- Another console application, called “Customer Application”, suitable for the customer needs. This application will be installed/running on the customers personal computers, not managed by the company. For that reason, a direct connection to the database (graphically depicted on red) must be avoid. Instead, data requests should be made through the “Orders API” exposed by the “Orders Server” component.
- A server application, called “Orders Server”, responsible to fulfill the all the data requests necessary to the “Customer Application” works properly. The set of available/possible requests must be minimized and, each one, should have a very specific purpose (e.g.: add a product to the shopping cart, check the status of his/her open orders).
- Another server application, called “AGV Manager”, responsible for managing and interacting with all the AGVs operating on the warehouse, namely (i) to send/assign tasks to a given AGV through the “CMD API”; and (ii) to receive from AGVs data to update their status on the system through the “Status API”.
- An application, called “AGV Digital Twin” able to mimic the general behavior of a real AGV. Among other things, it should (i) accept tasks to collect a set of products, (ii) report task completion, and (iii) simulate being moving through the warehouse to pick up products.

It is foreseen that multiple instances of the “AGV Digital Twin” are running simultaneously (but with distinct configurations) to simulate the existence of several AGVs on the warehouse.

The interaction between the different applications, represented through the provided and requested interfaces (e.g.: “CMD_API”, “STATUS_API”, “REQUESTS_API”, “ORDERS_API”), must be carried out in accordance with the communication protocol specified in the document called “Part III – Communication Protocol”. The unique exception is the interaction with the database which relies on the “SQL_API” provided by the adopted RDBMS and it should be abstracted by adopting an Object-Relational Mapping (ORM) tool.

Finally, it should be noticed that the development of these applications does not invalidate, in any way, the existence of a set of software components/modules that are common/shared between them. On the contrary, this practice is encouraged to promote an easy, fast, efficient, and effective maintenance, evolution and updating of the system as a whole and, namely, of the business rules.

5 Supplementary Information

This section provides some supplementary information to enrich previous business description and, therefore, to enhance business understanding. Some examples are also provided.

5.1 Questionnaires

Next, with the intent to demonstrate the kind of questionnaires the system should support, it is (i) suggested a base data structure for the questionnaires (cf. 5.1.1). Further, some example questions are provided together with an enumeration of the question types already identified (cf. 0). At last, a questionnaire for market prospection regarding cosmetic products is outlined (cf. 5.1.3).

5.1.1 Questionnaire Data Structure

As stated before, a data structure for the questionnaire's module is suggested (cf. Table 1, Table 2 and Table 3). Even though, this structure should be validated and adapted/enhanced as need by each team to fulfil all the requirements.

TABLE 1. SUGGESTED BASE STRUCTURE FOR A QUESTIONNAIRE.

Field	Description/Example
Id	Mandatory alphanumeric value to univocally identify a questionnaire. E.g.: "COSM22-01".
Title	The title of the questionnaire. It is a mandatory short sentence. E.g.: "Cosmetics Questionnaire".
Welcome Message	An optional message to be presented before any section/question. E.g.: "Hello, This questionnaire aims to… It takes approximately 10 minutes. Thank you very much for your time and support."
List of Sections	A non-empty list of sections. Answering to a section might be "mandatory", "optional" or "condition dependent". At least one section needs to be "mandatory".
Final Message	E.g.: "You have successfully completed the questionnaire. […] Thank you for your help."

TABLE 2. SUGGESTED BASE STRUCTURE FOR A SECTION.

Field	Description/Example
Section ID	Mandatory numeric value to univocally identify a section inside a questionnaire. Usually, it follows the sequence. E.g.: 1, 2, 3, …
Section Title	The title of the section. It is a mandatory short sentence. E.g.: "Basic Personal Information".
Section Description	An optional message highlighting some concern/purpose of the section. Multiple sentences need to be supported.
Obligatoriness	It might be one of "mandatory", "optional" or "condition dependent". If the latter is selected, a condition needs to be set.
Repeatability	Optional condition stating if the questions of this section are to be answered more than once. At least two kinds of conditions need to be supported: (i) based on numeric answers or (ii) on a set of selected values.
Content	A non-empty set of questions forming the section content.

TABLE 3. SUGGESTED BASE STRUCTURE FOR A QUESTION.

Field	Description/Example
Question ID	Mandatory numeric value to univocally identify a question inside a questionnaire. Usually, it follows the sequence. E.g.: 1, 2, 3, …
Question	A free text stating the question to be answered. E.g.: "How willing are you to try different cosmetic products?"
Instruction	Optionally a text with some answering instruction might be provided.
Type	Defines the kind of answer and available options (cf. 5.1.2).
Obligatoriness	It might be one of "mandatory", "optional" or "condition dependent". If the latter is selected, a condition needs to be set.
Extra Info	Additional structured information depending on the question type.

5.1.2 Question' Types and Examples

Questions adopted in any questionnaire are envisioned to be of one of the following types:

- Free-Text: it means the person answers the question by typing some text (cf. Figure 3). In some case, it is necessary to limit the amount of text (e.g.: single line, multiple lines setting a maximum number of chars per line).
- Numeric: it means the person answers the question by typing a numeric value. By default, no decimals are allowed. When no decimals are allowed, the question might be used to determine the repeatability of a section.
- Single-Choice: it means the person answers the question by selection one (and just one) of the provided options (cf. Figure 4). Each option consists of a value (not showed to the person) and a description. At least to option must be specified.
- Single-Choice with input value: very similar to the "single choice" but the last option, if selected, implies that the person must type a numeric value or a free text.
- Multiple-Choice: very similar to the "single choice", but instead of selection just one, the answering person might select more than one (cf. Figure 5). Sometimes, it is necessary to set a limit (e.g., maximum 3 option can be selected). By default, there is no limit.
- Multiple-Choice with input value: very similar to the "multiple choice", but the last option, if selected, implies that the person must type a numeric value or a free text (cf. Figure 6).
- Sorting Options: given two or more option the person answers the question by sorting the options as desired and in accordance with the instructions provided. By default, no order is assumed.
- Scaling Options: it means the person answers the question by selecting a value of a given scale (e.g.: unimportant, neutral, important) to each of the specified options (cf. Figure 7). The adopted scale is the same for all the options of a given question.

Some example questions of distinct type are provided in Figure 3 to Figure 7[2]. Yet, the proposed solution must have in mind that other question types might be required in a near future.

[2] The questions shown as example were extracted from: (i) https://www.smartsurvey.co.uk/s/QOVC0/ , (ii) https://www.surveymonkey.com/mp/skin-care-products-survey-template/ and (iii) https://www.questionpro.com/a/showSurveyLibrary.do?surveyID=130197&mode=1&render=old

Any question of a type comprehending options specification might be used to express a condition either to determine sections repeatability or section/question obligatoriness.

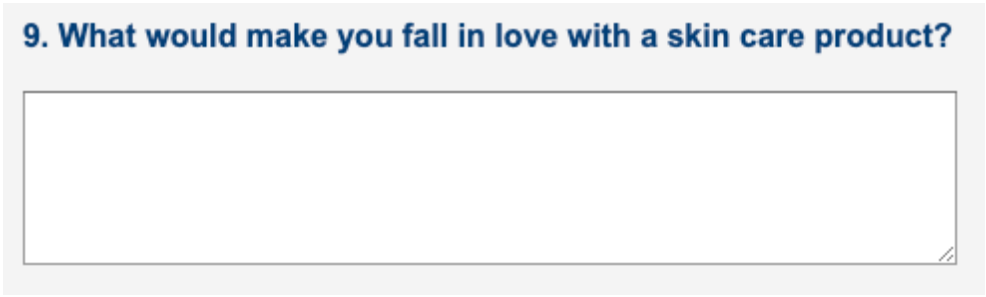

FIGURE 3. EXAMPLE OF A FREE-TEXT QUESTION.

FIGURE 4. TWO EXAMPLES OF A SINGLE-CHOICE QUESTION.

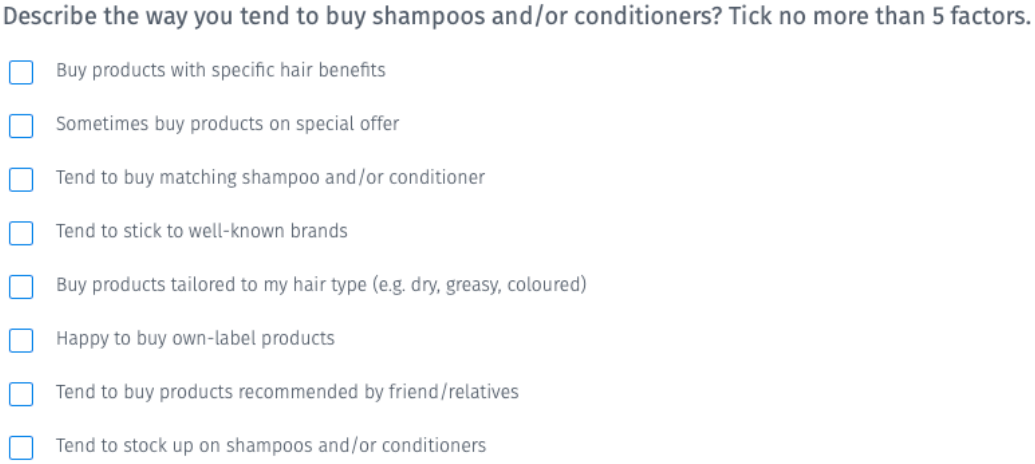

FIGURE 5. EXAMPLE OF A MULTIPLE-CHOICE QUESTION.

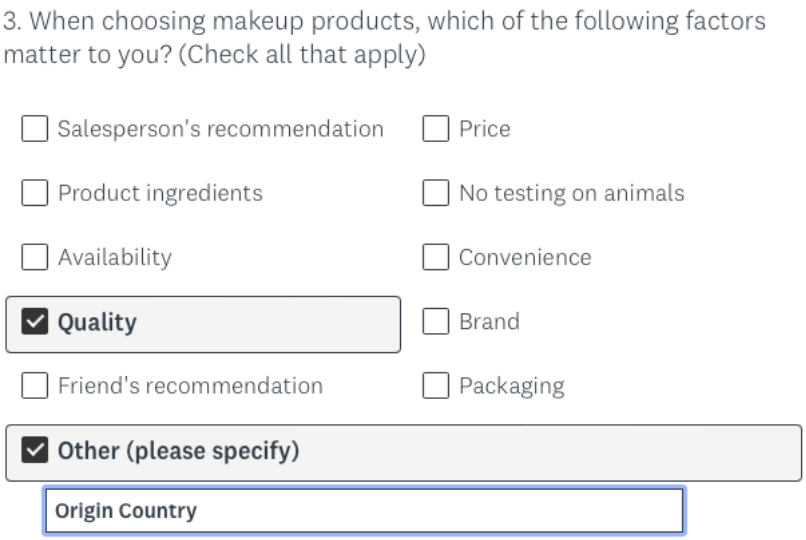

FIGURE 6. EXAMPLE OF A MULTIPLE-CHOICE QUESTION WITH INPUT VALUE.

FIGURE 7. EXAMPLE OF A SCALING OPTIONS QUESTION.

5.1.3 Outlining a Questionnaire

For demonstration purpose, a draft of possible questionnaire for market prospection regarding cosmetic products is briefly outlined.

Regarding the primary information (i.e.: id, title, welcome message, and final message), one can considered the examples provided on Table 1. Concerning the questionnaire sections, one could organize the questions in eight distinct sections as follows:

- Section 1: with a title “About You” and consisting of three questions: (i) “Q1” about the gender (cf. Figure 4 left); (ii) another one “Q2” about the age group (s)he belongs to (cf. Figure 4 right) and (iii) an optional one “Q3” regarding the country region where the person lives in (e.g.: “North”, “Center”, “South”, “Madeira” and “Azores”) such that one can be selected. This section is mandatory.
- Section 2: focusing on knowing the categories of cosmetics products the person is used to buy. For that, one “Multiple-Choice” question “Q4” with the options/values “Skin Care”, “Hair Care” and “Make-Up” is enough.
- Section 3: it would consist of several questions concerning “Skin Care” products as, for instance, (i) “Q5” the skin type of the person, (ii) “Q6” how often facial creams are used, and (iii) “Q7” which

factors are considered as relevant to buy. The latter question ("Q7") becomes mandatory (or not) depending on the answer given to question "Q6" (e.g.: it is mandatory if any value other than "never" is selected). This section is presented for answering just in the case that "Skin Care" option is selected on question "Q4".

- Section 4: like section 3, it would consist of several questions concerning the "Hair Care". As so, this section is presented for answering just in the case that "Hair Care" option is selected on question "Q4".
- Section 5: like section 3 and 4, it would consist of several questions concerning the "Make-Up". As so, this section is presented for answering just in the case that "Make-Up" option is selected on question "Q4".
- Section 6: focusing on knowing (i) "Q8" how willing the person is to try new products of a selected category (cf. question "Q4") and (ii) "Q9" how much (s)he is available to spend on such buy (per category). As it is easily perceivable, the questions of this section should be made once by each category that was select on the "Q4" answer.
- Section 7: it could consist of several questions aiming to know the level of (dis)agreement of the person answering the questionnaire with some statements (cf. Figure 7 example).
- Section 8: a fully optional section aiming to know more about the person answering the questionnaire as, for instance, (i) "Q10" the working status (e.g.: Full-Time, Part-Time, Unemployed, Student, Retired), (ii) "Q11" the marital status, (iii) "Q12" if (s)he has children and (iv) "Q13" how many children (s)he has.

5.2 Warehouse Plant

A warehouse can be thought as a matrix where each cell represents a small portion of the warehouse (i.e., a square) that has relevance for the warehouse structure (e.g., aisles location) and, therefore, also for its graphical representation as depicted in Figure 8. A similar representation it is intended to get on the dashboard will present the AGVs position in the warehouse.

The warehouse depicted on Figure 8 has 20 squares length and 18 squares width. It has (i) six AGV docks depicted as yellow squares and (ii) four aisles depicted as gray and light orange squares. The arrows are depicting the direction through which AGVs must access the aisle. Each aisle has distinct rows depicted as "merged" squares. At this respect, it is worth noticing that although the two aisles located in the middle of the warehouse are close together (i.e.: aisles are back-to-back), AGVs access both aisles by different directions. The remainder squares (i.e., not colored ones) represent free space through which AGVs can move around.

By simplicity, an AGV can only move horizontally or vertically. I.e.: at the same time, it can only be changing its position regarding length or width respectively.

Information regarding the warehouse plants is available through JSON files. Figure 9 shows an excerpt of the JSON file describing the warehouse depicted in Figure 8.

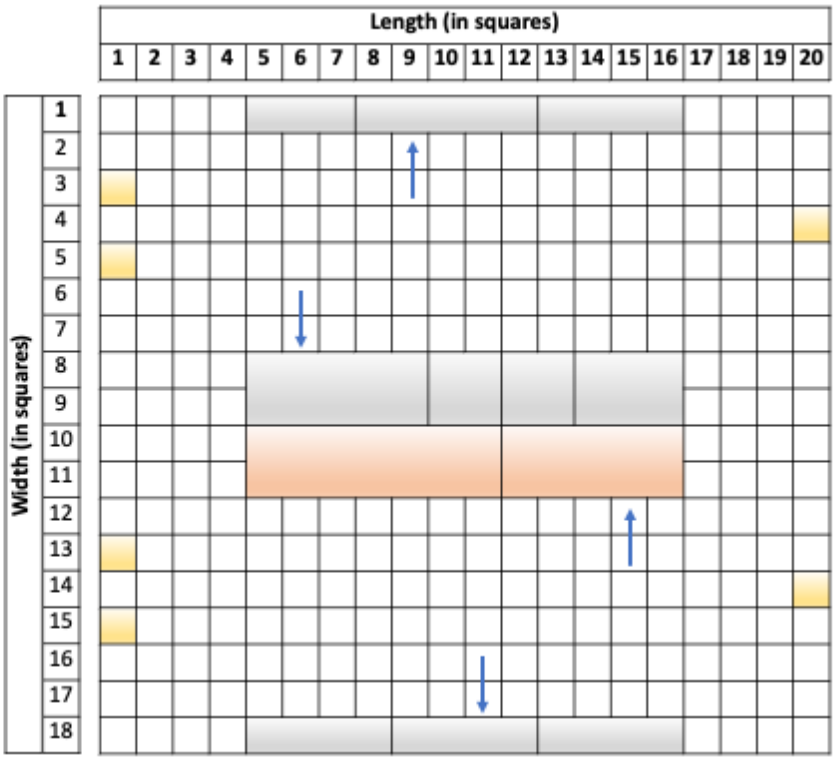

FIGURE 8. AN EXAMPLE OF A WAREHOUSE REPRESENTATION.

The JSON structure is as follows:

- *Warehouse*: a textual description of the warehouse to which the information pertains.
- *Length*: a numeric value specifying the length of the warehouse.
- *Width*: a numeric value specifying the width of the warehouse.
- *Square*: a numeric value specifying the length/width of the smallest square relevant for the warehouse.
- *Unit*: the metric unit (e.g.: centimeters, millimeters) in which the values of the *Length*, *Width* and *Square* fields are expressed.
- *Aisles*: a list of the aisles existing in the warehouse. For each aisle, it is specified an identifier (*Id*), three squares (named *begin*, *end* and *depth* respectively) to state/determine the aisle location, the *accessibility* direction, and the aisle' *rows*. Squares are identified by two coordinates: *lsquare* and *wsquare* denoting the number of the square regarding length and width respectively. The accessibility direction consists of two chars: the first char (either "w" or "l") denotes that accessibility is made varying the **w**idth or the **l**ength regarding the aisle; while the latter char denotes if such variation is computed by incrementing ("**+**") or decrementing ("**-**") width/length. Similarly, for each row, it is specified its identifier (*Id*) and the location of the row in the aisle is given by two squares: *begin* and *end*.
- *AVGDocks*: a list of the AVG docks existing in the warehouse. Like aisles, a dock has an identified (*Id*), a location determined by means of three squares (*begin*, *end* and *depth*) and its *accessibility* direction.

```
{
    "Warehouse": "Warehouse Example 1",
    "Length": 800,
    "Width": 720,
    "Square": 40,
    "Unit": "cm",
    "Aisles":[
        { ... },
        {
            "Id":2,
            "begin": {"lsquare":5, "wsquare":8},
            "end": {"lsquare":16, "wsquare":8},
            "depth":{"lsquare":16, "wsquare":9},
            "accessibility":"w-",
            "rows":[
                { ... },
                { ... },
                {
                    "Id":3,
                    "begin": {"lsquare":12, "wsquare":8},
                    "end": {"lsquare":13, "wsquare":8},
                    "shelves":4
                },
                { ... }
            ]
        },
        { ... },
        { ... }
    ],
    "AGVDocks":[
        { ... },
        {
            "Id":"D2",
            "begin": {"lsquare":1, "wsquare":5},
            "end": {"lsquare":1, "wsquare":5},
            "depth":{"lsquare":1, "wsquare":5},
            "accessibility":"l+"
        },
        { ... },
        { ... },
        {
            "Id":"D5",
            "begin": {"lsquare":20, "wsquare":4},
            "end": {"lsquare":20, "wsquare":4},
            "depth":{"lsquare":20, "wsquare":4},
            "accessibility":"l-"
        },
        { ... }
    ]
}
```


FIGURE 9. EXCERPT OF A JSON FILE DESCRIBING A WAREHOUSE PLANT.

5.3 AGV Digital Twin

This section presents some tips/guidelines for the AGV Digital Twin development, namely its decomposition in several sub-components/modules. Accordingly, at least seven distinct modules are being foreseen:

- Communications: responsible for receiving/sending information from/to other external components such as the "AGV Manager" (cf. Figure 2). For instance, it can receive the warehouse plant, the location of the other AGVs as well as the command to pick-up some product(s). On other hand, it can send information about its current location, status and so on.
- Control Unit: responsible for processing/executing the received commands, keeping the AGV internal information (e.g.: its position, detected obstacles, battery status) updated and inform (through the communications module) the "AGV Manager" about its global status.
- Simulation Engine: responsible for knowing the warehouse plant and the current location of all AGVs and based on that, transmit information to the AGV sensors at, for instance, every 100 ms.
- Sensor: receives information from the "Simulation Engine" and emits obstacles alerts to the other modules. For instance, emits "1" if an obstacle is detected at one square distance and, therefore, indicates that AGV should stop; emits "2" if an obstacle is detected at two squares distance and, therefore, indicates the AGV should reduce its speed; and emits "0" for obstacles detected at three or more squares distance and, therefore, indicates the AGV can keep going. Typically, an AGV has, at least, 8 sensors (visualizing an AVG as a rectangle, it has two sensors at each corner). Each sensors emits signals considering its position on the AGV.
- Positioning: responsible for calculating the new AGV position in the warehouse based on the AGV (i) speed, (ii) direction, (iii) last position and (iv) current route. Reads/updates information from/on the "Control Unit" module. Speed might be expressed in squares per second.
- Route Planner: responsible for (re)computing routes based on a source and target location on the warehouse considering the warehouse plant only. It is worth noticing that AGV can only move horizontally or vertically.
- Battery Management: responsible for monitoring the battery status and simulating battery consumption and recharge according to the AGV status. For instance, when the AGV is moving battery consumption might be computed based on the travelled distance, but when the AGV is waiting/stopped on its dock battery consumption might be computed based on time.

Notice that some modules need to (intensively) interact with each other and/or need to read/update the same piece of information concurrently. As so, it is essential applying the proper concurrency mechanisms to ensure data consistency effectively.

INTEGRATIVE PROJECT OF THE 4TH SEMESTER OF LEI-ISEP

2021-2022 (version III.a)

PART III – Communication Protocol

1 Context

In the 2021-2022 academic year, the fourth semester (i.e. 2nd year, 2nd semester) of the Degree in Informatics Engineering (LEI) of the Instituto Superior de Engenharia do Porto (ISEP) adopts a teaching-learning process based on the development of a single project that enhances the integration and application of knowledge, skills and competencies of all course units taught through the semester: Applications Engineering (EAPLI), Laboratory and Project IV (LAPR4), Languages and Programming (LPROG), Computer Networks (RCOMP) and Computer Systems (SCOMP).

The project, common to all course units, requires the use of an application communication protocol, called Smart Products Order Management System Protocol (SPOMSP 2022), for interaction between the applications composing the system being developed (cf. document with the same designation whose subtitle is " PART II – System Specification").

2 SPOMS Protocol (SPOMSP)

The SPOMS Protocol aims to facilitate data exchange between the applications comprising the system being developed and in line with previous recommendation (cf. Figure 1).

2.1 SPOMSP Description

The SPOMS Protocol (SPOMS) has the following characteristics/features:

- It is TCP (Transmission Control Protocol) based, therefore, prior to any actual data exchange, a TCP connection must be established.
- It uses the client-server model. The **client application** is the one that takes the initiative of requesting a TCP connection establishment with the counterpart **server application**, which accepts the connection request.
- Once a TCP connection is established, the client-server relation persists and only the client application is allowed to take the initiative of sending data, i.e.: **a request**. The server application must be passively waiting for a request and only then is authorized to send data, i.e.: **a response,** to the received request.
- Every request (sent by the client) has a mandatory response (sent by the server), both share a same message format described ahead (cf. section 2.3).
- Once established, the TCP connection between client and server is kept alive and used for all requests needed while the client application is running.

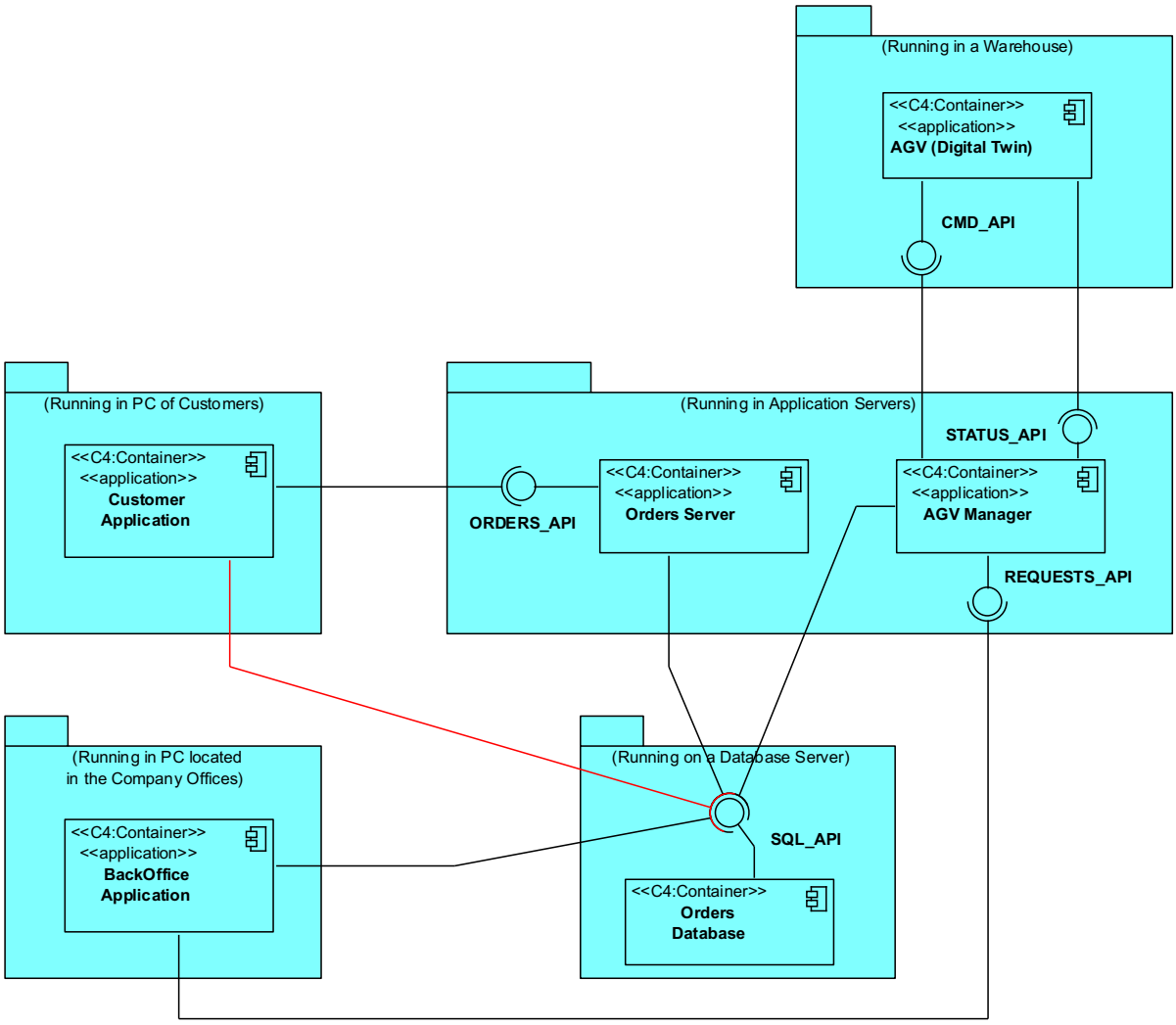

FIGURE 1. SET OF ENVISIONED APPLICATIONS.

2.2 Applications and SPOMSP Roles

Considering the envisioned applications (cf. Figure 1) and the general protocol description presented in section 2.1, Table 1 highlights the relationships between applications and their SPOMS role.

TABLE 1. SPOMS APPLICATIONS USING SPOMSP.

SPOMS Application	SPOMSP Role(s)
Orders Server	Server application
AGV Manager	Server application
Customer Application	Client application
Backoffice Application	Client application
AGV (Digital Twin)	Server application and client application

2.3 SPOMSP Message Format

Every data exchange through the TCP connection (requests and responses) must comply with the bytes sequence description in Table 2. This structure corresponds to the message format version one, which is not expected to change during the SPOMS development.

TABLE 2. SPOMSP MESSAGE FORMAT

Field	Offset (bytes)	Length (bytes)	Description
VERSION	0	1	SPOMSP message format version. This field is a single byte and should be interpreted as an unsigned integer (0 to 255). The present message format version number is one.
CODE	1	1	This field identifies the type of request or response. It should be interpreted as an unsigned integer (0 to 255).
D_LENGTH_1	2	1	These two fields are used to specify the length in bytes of the DATA field. Both these fields are to be interpreted as unsigned integer numbers (0 to 255). The length of the DATA field is to be calculated as follows: D_LENGTH_1 + 256 x D_LENGTH_2 The length of the DATA field may be zero, meaning it does not exist.
D_LENGTH_2	3	1	
DATA	4	-	Contains data to meet the specific needs of the participating applications.

2.4 SPOMSP Message Codes

Table 3 contains a list of some message codes that must be implemented by every application using SPOMSP. Additional unique message codes will be established by the development teams as needed.

TABLE 3. MESSAGE CODES.

Code	Meaning
0	**COMMTEST** - Communications test request with no other effect on the server application than the response with a code two message (ACK). This request has no data.
1	**DISCONN** - End of session request. The server is supposed to respond with a code two message, afterwards both applications are expected to close the session (TCP connection). This request has no data.
2	**ACK** - Generic acknowledgment message. Used in response to requests with codes zero and one but may be used for other requests. This response has no data.

Notice that, as long as the message format is the same, adding new message codes doesn't change the message format version.