Wearables in Logistics - Demo Facility

Final report of bachelor thesis

Submitted by Lukas Rolle

In fulfilment of the requirements for the degree
Bachelor of Science in Informatics
To be awarded by the
Fontys Hogeschool Techniek en Logistiek

Venlo, July 13, 2017

Information Page

Student Information

Name:	Lukas Rolle
Date of Birth:	23 March 1994
Place of Birth:	Moers, Germany
Student Number:	2310309
Study Course:	Informatics: Software Engineering

Thesis Information

Time frame:	01. February - 30. June 2017
Date of Delivery:	28 March 2017

Company Information

Name:	Fontys Hogeschool Techniek en Logistiek
Address:	Tegelseweg 255
Postal code:	5912 BG
City:	Venlo
Country:	Netherlands

Educational Institution

Name:	Fontys Hogeschool Techniek en Logistiek
Address:	Tegelseweg 255
Postal code:	5912 BG
City:	Venlo
Country:	Netherlands

Examination Committee

Company Supervisor:	Stefan Sobek
Supervising Lecturer:	Thijs Dorssers
Examinator:	Ferd van Odenhoven
External Representative:	J. Janssen

Statement of authenticity

I hereby solemnly declare for this submitted work, that:

- I myself wrote this internship report, without the assistance of any third party,
- I did not cut and paste any information (text, figures, diagrams, tables,..) from others without appropriate use of quotation marks and direct reference to their work,
- I did not re-word the ideas of others without proper and clear acknowledgement,
- I did not make use of ideas or suggestions that originated from others and claim these as my own,
- I did not include words from other's work without permission.

I am fully aware that any violation of the above will be declared fraud and may result in disadvantageous consequences for me (for example withdrawal of study credits and, in case of a repeated violation, withdrawal of complete study units). If fraud can be proved, I will be required to bear the costs of investigation into and sourcing of the original document.

Name: Lukas Rolle

Place / Date: Venlo, July 13, 2017

Signature:

Abstract

Many small- and medium sized enterprises have difficulties in trying out new technologies, as they often just do not have the needed resources to spend on the newest technology. The LOGwear project aims to give these companies the possibility to stay competitive in that area, it tries to take generalized logistics processes and combines these with wearables and make the results available to everyone.

This thesis is about the creation of a generalized reference model, that allows everyone a head start on creating their own application with a wearable in the area of logistics. Furthermore a demo facility is planned to be created to allow everyone that is interested in adopting the wearable technology to get hands-on experience. This demo facility is a physical area, where interested can come to visit and get a demonstration of how a process could be improved with the help of wearables. The environment for the demo facility is trying to mock a real world logistics company as closely as possible, using processes from logistics companies as a base.

This should allow companies, that are interested in adopting new technology, to inform themselves about these technologies easily, find technologies they think are interesting for their way of working, see how the technology actually works in a hands-on environment and then make an educated decision if the adoption of a wearable is something that could help improve their own processes.

Kleine- und Mittelständische Unternehmen haben oft Probleme damit neue Technologien auszuprobieren, da die dazu nötigen Ressourcen oft nicht vorhanden sind um die neuesten Technologien auszuprobieren. Das LOGwear Projekt versucht diesen Unternehmen die Möglichkeit zu geben in dieser Umgebung trotzdem wettbewerbsfähig zu bleiben. Es versucht standardisierte logistische Prozesse zu nehmen und Sie mit Wearables zu verbinden und stellt die Ergebnisse frei zur Verfügung.

Diese Abschlussarbeit beschäftigt sich mit der Erstellung eines standardisierten Referenzmodells, das eine schnellere Erstellung einer eigenen Anwendung mit wearables erlaubt. Weiterhin ist eine Testeinrichtung geplant die kleinen- und mittelständischen Unternehmen die daran interessiert sind, die Möglichkeit gibt Wearables selbst auszuprobieren. Die Testeinrichtung ist eine Umgebung zu der interessierte gehen können, um selbst sehen oder ausprobieren können ob ein bestimmtes wearable etwas für ihre Unternehmensstruktur ist. Es wird versucht mit der Testeinrichtung die tatsächliche Umgebung eines Logistikunternehmens nachzuahmen, in dem man wirkliche logistische Prozesse als Basis für die Testeinrichtung nimmt.

Damit sollte interesierten Unternehmen erlaubt werden, sich auf einfache Art und Weise über neue Technologien zu informieren, Technologien zu finden die in die Unternehmensprozesse passen könnten, zu sehen wie diese Technologien tatsächlich funktionieren und daraufhin eine informierte Entscheidung zu treffen, ob diese Technologie tatsächlich etwas ist, was Ihre Unternehmensprozesse verbessern könnte.

Contents

List of Figures

List of Tables

List of Listings

Glossary

battery pack a device used to store a bigger amount of power, generally able to charge a phone or similar devices multiple times.

cloud computing a technique that provides computing resources in a virtual server environment. Giving customers more or less computing resources depending on the demand of each specific user.

deserialize the reverse process of serialization, i.e. creating the state of an object from some format that has been stored in some way.

fat client a computer or program that is calculating everything on its own. A terminal computer, that is not relying on a server or cloud for any calculation is a fat client.

getter a method used to get the value of a variable, for purposes of encapsulation.

haptic feedback feedback given to a user via touch, most of the time being vibrations. Differences in haptic feedback can be the amount, place or intensity of the vibrations.

hot swap is the action of swapping a part of a device for a replacement, while the device is still powered on. For a phone it could be swapping the battery for another one, without turning the phone off, the phone has to support this feature.

package in software engineering terms, a package is a collection of classes that are in general grouped by the responsibilities they have.

parcel a package in the logistics sense, a physical package that is a part of an order.

push message a push message is a message that is send to a device without an initiating request from the target device. In general a message from a server was always only send to a device on request of that device, but push messages are send to a device on demand of a service running on a server.

reference architecture a template solution in a domain that gives multiple models, sets of functions and classes to describe in detail how something is supposed to work.

reference model an abstract model used to describe the general design of an application in a specific environment. A reference model is in general technology-agnostic and can be extended to ones wishes.

ring scanner a scanner that is worn on one or multiple fingers similar to a ring. This allows workers to lay the scanning device down when needing both hands for work.

sandbox an environment where something can be tested without having an influence on anything else. In software engineering, an environment where an application can be tested without damaging the live version.

serialize the process of serialization in computer science describes the translation of the state of an object into a format that can be stored.

setter a method used to set the value of a variable, for purposes of encapsulation.

smartglasses a set of glasses falling under the category of wearable technology, generally able to project something on the glasses to output information to the user.

smartwatch a watch falling under the category of wearable technology, generally able to communicate with a phone to have information directly available on the wrist.

thin client a computer or program that is relying on a server application to do computation for it.

wearable a piece of technology that can be worn on the body.

List of Abbreviations

API Application Programming Interface.

CASE Computer-Aided Software Engineering.

DB Database.

EF Entity Framework.

EU European Union.

FHTenL Fontys Hogeschool Techniek en Logistek.

HPU Holographic Processing Unit.

HSNR Hochschule Niederrhein.

ID Identifier.

IDE Integrated Development Environment.

JEE Java Platform, Enterprise Edition.

JSF JavaServer Faces.

JSON JavaScript Object Notation.

MSSQL Microsoft Structured Query Language.

MVC Model View Controller.

MVP Model View Presenter.

MVVM Model View ViewModel.

MWEIMH NRW Ministerium für Wirtschaft, Energie, Industrie, Mittelstand und Handwerk des Landes Nordrhein-Westfalen.

OMG Object Management Group.

QR Quick Response.

REST Representational State Transfer.

RFID Radio-Frequency Identification.

SDK Software Development Kit.

SME Small and Medium-sized Enterprises.

SOA Service-Oriented Architecture.

UML Unified Modeling Language.

VCS Version Control System.

WLAN Wireless Local Area Network.

WMS Warehouse Management System.

WP Work Package.

1 Introduction

This thesis is written as the completion of the study course software engineering at the Fontys Hogeschool Techniek en Logistiek in Venlo, Netherlands. The graduation project is conducted at the LOGwear research project, the research project itself will be explained in section 2.1. The thesis is written over the course of five months and is documenting the thought and creation process of the project, of creating a demo facility for the usage of a wearable in a logistics process.

As this thesis was written on a software engineering topic, pieces of code will occur throughout the thesis, single words, like variables, or classes will be written in a `mono-spaced-font`.

Furthermore the word package in this report is always meant as a software package and never as the package that is used in the logistics sector, the word parcel is used in this context.

Overview

The following chapters in this report will contain these topics:

Context and Scope
In chapter 2 the context of the project will be elaborated, naming the involved parties and what the research project is about. Furthermore the scope of the thesis will be defined, including demarcation. The general information, including project management details are explained.

Initial Analysis
In chapter 3 the analysis part of this project will be explained. Especially how the initial wearables information was acquired.

Research
In chapter 4 the general research part that was done during the thesis will be explained. The results of the research will be named and the chosen Computer-Aided Software Engineering (CASE) tools listed.

Reference Model
Chapter 5 contains the design process and the problems connected with the reference model.

Demo Facility
Chapter 6 will contain the infrastructure, design and implementation of the demo facility that is to be created to showcase the possibilities of wearables in the area of logistics.

Conclusion
Chapter 7 will contain the conclusion, recommendations and a look into the future, showing how the project is planned to develop.

2 Context and Scope

This chapter contains the context of the project, including an explanation of the research project LOGwear in section 2.1 and the parties involved in it. Furthermore the general project management strategies will be explained in the following sections: section 2.2 explains the stakeholders of the thesis, section 2.3 goes over the risks, section 2.4 explains the quality management for the project and section 2.5 describes the definition of done. Finally, section 2.6 is about the time planning and scheduling done for the given project.

2.1 LOGwear

LOGwear is a research project that aims to bring wearables to the area of logistics, especially to Small and Medium-sized Enterprises (SME). It is a German-Dutch research project were multiple parties are cooperating to create results. Involved in this are two Universities of applied sciences, Fontys Hogeschool Techniek en Logistek (FHTenL) in Venlo, Netherlands as the lead partner and Hochschule Niederrhein (HSNR) in Germany.

Further on there are also multiple partner companies involved in the project, namely KLG Europe bv, Helmut Beyers GmbH and imat-uve GmbH. These partner companies are there to give the knowledge about logistics processes, as well as to verify and test the results.

The project is backed within the scope of the INTERREG Deutschland-Nederland initiative. It is backed by the European Union (EU), Ministerium für Wirtschaft, Energie, Industrie, Mittelstand und Handwerk des Landes Nordrhein-Westfalen (MWEIMH NRW) and the Provincie Limburg as well. The official kickoff meeting for logwear was in september 2016 and the project will run until march 2018.

The logwear project is consisting of three main Work Package (WP)s. (Logwear.eu, 2017)

Knowledge Base (WP1)
The knowledge base is a platform that allows to exchange information between logistics companies which wearable can be used for which process. (Sander, 2017) (Canders, 2017)

Reference Architecture / Reference Model (WP2)
The expected result for WP 2 used to be a reference architecture, this has internally changed to a reference model, the differences about these two and what is expected from the reference model can be found in chapter 5.

Demo Facility (WP3)
The demo facility is the creation of a physical demo that allows SME to see the benefits of wearables for a demo process. The difference between the demo facility created during this thesis and WP 3 is, that the WP 3 is the implementation of a wearable solution at a pilot company, while the demo facility created during this thesis is in an enclosed environment that does not need to follow all constraints of deploying something at a company. Further details for the demo facility are described in chapter 6.

2.2 Stakeholder

Stakeholder Management involves identifying parties that are involved in the project. This ranges from people that are actively part of the development of the project or companies that might be interested in the end result. In the table 2.1 the most important stakeholders can be found. The stakeholders will themselves will be further explained in sections 2.2.1, which will explain the internal stakeholders, and 2.2.2 will explain the external stakeholders involved in the project.

Nr	**Stakeholder**	**Company / Institution**	**Internal / External**	**Level of Interest**	**Level of Influence**	**Potential management strategies**
1	Employer (Em)	FHTenL	Internal	Medium	High	Keep Satisfied
2	Student Workers (SW)	FHTenL	Internal	Medium	Low	Keep Informed
3	Graduation Student (GS)	FHTenL	Internal	High	High	Key Player
4	Project Manager (PM)	FHTenL	Internal	High	High	Key Player
5	Company Supervisor (CS)	FHTenL	Internal	High	High	Key Player
6	Project Team (PT)	FHTenL	Internal	High	High	Key Player
7	Partner University (PU)	HSNR	External	Medium	Low	Keep Informed
8	Pilot Company (PC)	KLG	External	High	High	Key Player
9	Examiner (Ex)	FHTenL	External	Low	High	Keep Satisfied
10	Supervising Lecturer (SL)	FHTenL	External	Medium	Low	Keep Informed

Table 2.1: Stakeholder Register

Figure 2.1 can be used to visualize the importance of the stakeholders. The color is used to emphasize the importance that this stakeholder is properly managed. The color is moving from blue to red over a yellowish green, the less blue and the more red a color has determines again the fact how important this stakeholder is.

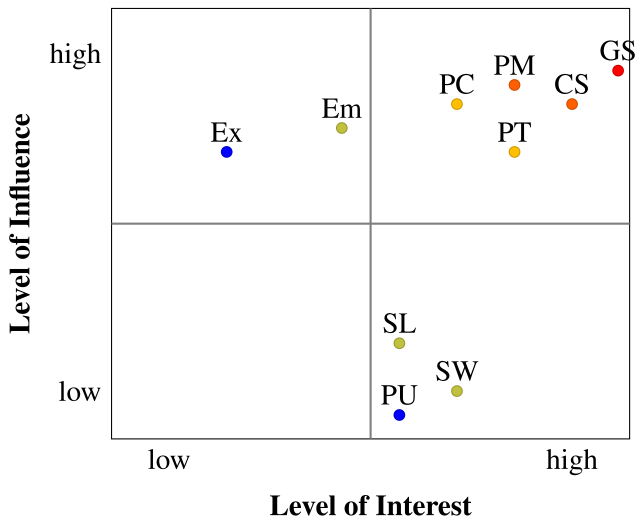


Figure 2.1: Stakeholder Graph

2.2.1 Internal Stakeholders

Internal Stakeholders are parties that are a part of the team that is working directly on the project in one way or the other. In this section, the internal stakeholders mentioned in table 2.1 will be listed again and explained.

Employer

The employer in this project is an institution and not a single person. This does not change the fact, that the employer is interested in the project, as he is financing the project. Also the employer could change the outcome, if he is not accepting the proposed plans. It is important, that this party is kept satisfied as more work could be created when the plan has to change.

Student Workers

Student Workers are employed to help in the LOGwear project in general. While they currently are not involved with the process of the creation of the demo facility that might change in the future, therefore they should be kept informed.

Graduation Student

The graduation student is the person mainly responsible for the development of the demo facility and therefore has a lot of responsibility and interest towards the project.

Project Manager

The project manager is responsible for the general planning of the project. Planning meetings with the different parties and coordinating them.

Company Supervisor

The company supervisor is looking over the progress of the graduation student and is giving advice if needed.

Project Team

The project team are the members of the team actively developing the application prototype and are involved in building the demo facility afterwards.

2.2.2 External Stakeholders

External Stakeholders are parties that are involved in the project, but are not a part of the team actively developing the project. In this section, the external stakeholders mentioned in table 2.1 will be listed again and explained.

Partner University

The partner university is also working on the LOGwear project, but on a different aspect. They might be interested in project of creating a demo facility, but probably will not interfere with it.

Pilot Company

The pilot company involved is the logistics company KLG. They bring in the highest amount of domain knowledge and are interested in the project to improve their own processes. They could influence the project easily by not approving the planned demo facility due to problems with how the logistics process is modelled.

Examiner

While the examiner is not involved in the project itself, the examiner will finally assesses the performance of the graduation student.

Supervising Lecturer

The supervising lecturer is there to answer questions and support the student from a software engineering standpoint. While not having a lot of influence on the project itself, the supervising lecturer is interested in what the student is doing and especially how he is doing it.

2.3 Risks

Risk management is about identifying risks and finding solutions to problems before they can occur. The list of risks can be found in table 2.2. The identified risks will increase as the project moves forward. Especially when a decision is made for the wearable and the process.

Nr	Risk Name	Description	Probability	Impact	Root Cause	Potential Responses	Risk Owner
1	Wearable unavailable (W)	The wearable desired to be used in the demo facility is unavailable.	Low	Medium	The desired product is a prototype or similar.	Choosing a different wearable that is already readily available.	Graduation Student
2	Demo Area (D)	A demo area is in mind that could potentially be rented, but that could not be possible.	Medium	Medium	The owner of the place does not rent the area.	Researching possible places where the demo facility could be created.	Graduation Student
3	Unusable wearable (U)	A wearable is chosen that does not have the capabilities to fulfill the things that were planned with the demo facility.	Low	High	Too little research done on the wearables, or the researched material was wrong.	Altering the demo scenario to accommodate the problems with the wearable.	Graduation Student
4	Vocabulary unclear (V)	The vocabulary used in the logistics branch, especially abbreviations and acronyms might cause problems in communication.	High	Low	The graduation student has too little knowledge of the logistics branch, at the beginning of the project.	Asking questions if a word's or sentence's meaning is not clear.	Graduation Student
5	Communication Problems (C)	When explaining a task, a phrase or word is understood differently from different parties.	High	Medium	The proper definition is not known to everyone and are expecting a different meaning from a given phrase or word.	When the misunderstanding is discovered the different parties talk out what is expected from that term and come to a common understanding.	Project Team

Table 2.2: Risk Register

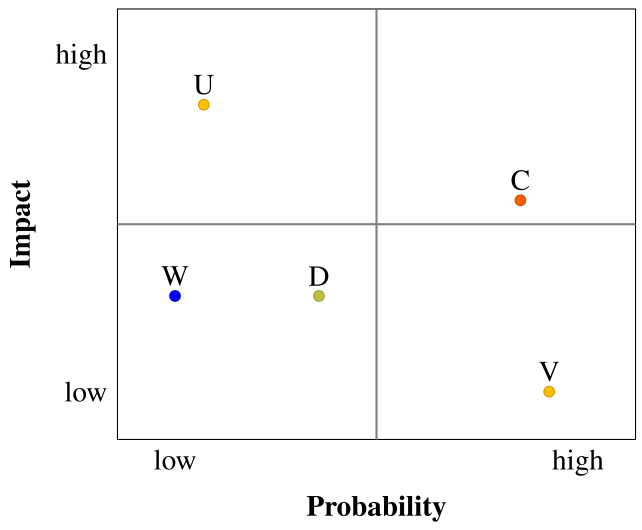


Figure 2.2: Risk Graph

In figure 2.2 the risks can be seen in a graph that shows their Probability and Impact again. The color emphasizes the amount of attention a risk should get, in order for the project to continue smoothly. The color is moving from blue to red over a yellowish green, the less blue and the more red a color has determines again the fact how important it is to handle this risk. The figure also shows more fine grained the probability and impact of the risks than just low, medium and high.

2.4 Quality Management

Metrics used to determine the quality of the project:

Code Coverage
Code coverage is a useful metric showing the amount of code covered by unit tests. While they should not be the only way of testing an application of this size, they are still useful to see if a single components works on their own. Due to time constraints code coverage of 100% is most likely not achievable and a coverage of about 80% to 90% is aimed towards.

Language Conventions
Programming languages usually have conventions on how the semantics of the code should look like to be considered code. These conventions will most likely be followed, and if there are any changes to that they will be listed here.

Documentation
Documentation of a method will include the parameters involved, the return result, usage and general how something works. Furthermore documentation explaining the class and package will be created in a similar fashion. Furthermore created diagrams will be made available and also documented.

Performance Testing
The wearable part of the application might need to be performance tested, as wearables tend to be less powerful than most computing devices normally used. This will be decided when the application is creating problems regarding to performance.

2.5 Definition of Done

A part of the software project is done, when it is fully designed, implemented, documented and tested. When that part of the application is passing all of these criteria it is added to a repository where the result is build. When that build is successful that part of the application is done, for the moment. When that part needs to be changed in the future the same procedure will be used again.

2.6 Planning

The schedule for the project can be seen in figure 2.3. The project will be executed in a scrum-like way.

01.Feb - 17.Feb	Initial Logistics Processes and Wearables Research
02.Mar	Project Plan
20.Feb - 24.Mar	Reference Model
28.Mar	Mid-Term Report
27.Mar - 31.Mar	Research Demo Facility
11.Apr	Mid-Term Presentation
3.Apr - 14.Apr	Setting up Infrastructure for Demo Facility
13.Jun	Final Report
17.Apr - 17.Jun	Design and Implementation Demo Facility Application
19.Jun - 23.Jun	Setting up physical Demo Environment
27.Jun	Final Presentation
26.Jun - 30.Jun	Preparing for Handover

Figure 2.3: Schedule

The schedule is divided in work packages that are to be executed and milestones that are a part of the project. It is to be noted, that each work package could be split into multiple sprints in the future. The milestones are the entries in the schedule that are just having a single date and not a range of dates. In the following subsections the work packages will be explained.

2.6.1 Initial Logistics Processes and Wearables Research

This work package includes research about the given processes and wearables in general, as well as already choosing potential wearables that could be used to improve the process. The end result for this should be a decision on a process and a wearable. The result for this could potentially take longer than this task is scheduled, the reason for this is that some wearables are ordered as part of the process, and a decision can then just be made when those wearables are delivered. The wearables should be ranked after getting hands-on experience on them, therefore some of them have to be ordered first.

2.6.2 Reference Model

A reference architecture should be created for a sample wearable application. What this work package contains is, the creation of diagrams which show the communication from a wearable to the Warehouse Management System (WMS) or a similar system. What should not be created is a full reference architecture for a process that is implemented with a concrete wearable.

It is about creating the always needed layers when using a wearable in a way that supports most wearable solutions.

2.6.3 Research Demo Facility

This task includes researching what physical objects and what systems would be needed to create a demo facility that could showcase a single process with a single wearable. This also includes the gathering of knowledge of where the demo facility should be created and where to get the needed objects.

2.6.4 Setting up Infrastructure for Demo Facility

The infrastructure for the demo facility includes multiple things, like setting up a server to run a database for the demo facility and creating an interface to connect to it.

2.6.5 Design and Implementation Demo Facility Application

The work package includes the creation of the software design and implementation for the wearable and further aspects that are needed to fully showcase a process.

2.6.6 Setting up physical Demo Environment

This task includes the physical creation of the demo facility. This means setting up shelves with parcels to scan and put on a hand pallet truck. Setting up barcodes on the packages to scan. Setting up an environment that can showcase what is happening better to an audience.

2.6.7 Preparing for Handover

Since this project is part of the bigger logwear research project, the results that were created during this project will be used in the future, therefore all the resources will be made available to a future developer. Furthermore additional documentation and a handover document are part of this work package.

3 Initial Analysis

The first questions asked in this project were, what logistics processes are existing and which wearables can be used for them. At the beginning of the project not a lot of information was available. The only information that was available from the start, were the logistics processes that have already been researched beforehand. The processes themselves will be discussed in section 4.1. This chapter will discuss the initial list of wearables that could be considered and which requirements are existing for these wearables and why these requirements.

3.1 Requirements towards Wearables

Before talking about wearables themselves, the base requirements needed to be set to start looking for wearables. One of the processes can be seen in the appendix, figure A.1. With the aid of this process the requirements a wearable needed to be able to fulfil can be set. Following will be short descriptions of requirements and how they are existing in the process. For further information the whole process itself can be looked at, but the parts that are interesting for the wearables will be listed here. First the functionality the wearable needs to have will be named, then the proof from the process why it needs to have that functionality and furthermore what this means for the wearable concretely.

The wearables needs to be able to:

- scan some kind of barcodes. This functionality is needed due to the multitude of areas that involve scanning in the given processes. Since the new wearable should replace the hand scanner that is currently used, it needs to be able to scan barcodes. For a wearable this means there needs to be some kind of visual input, this could mean just a simple camera included or a dedicated barcode scanner included in the wearable.

- provide the user with information. This functionality is needed to support the database connections displayed in the processes that should show some kind of data to the user afterwards. What this means for a wearable is, that it should have some kind of way to provide the user with information. This could be a screen, a speaker, haptic feedback or some other method of providing someone with information.

- send confirmations to the WMS. This is often done in the processes and every worker needs to do it multiple times during each process. The wearable should therefore have the functionality to connect to an outside computer system with a wireless connection.

These requirements were talked about with the company supervisor and accepted as main requirements towards wearables that absolutely need to be fulfilled. One more requirement was added at this point after the agreement on the already mentioned requirements. The ability to use the wearable hands-free for most of the time, meaning that hands should not be needed for operating the device at most times. For example the scanning process itself should leave the worker without the need to occupy his hands. But when just walking through the warehouse the wearable could be operated using the hands and it would not be something that immediately disqualifies a wearable.

3.2 Wearables

The initial search for wearables started with wearables that have already been used in projects similar to this. One of the projects that was looked at for this was the pick-by-vision project conducted by Schwerdtfeger (Schwerdtfeger, 2009). That is focusing on the application of smart glasses for the order picking process. A case study by DHL shows the usage of the Vuzix M100 and the Google Glasses and is resulting in improved times per task in the picking process (DHL, 2015).

The results of these previous researches led to the inclusion of a lot of general purpose smartglasses. A lot of the smartglasses mentioned were just part of a short internet research, checking the general functionality of the devices and if it would be able to fit the things needed for the processes given in section 3.1. The initial list of smartglasses will be listed below:

- Vuzix M100
- Vuzix M300
- Vuzix Blade 3000
- Daqri Smart Glasses
- Epson Moverio BT-300
- Epson Moverio BT-200
- Microsoft HoloLens
- ODG R7 AR
- Sony SmartEyeGlass
- Google Glass

Apart from smartglasses one other type of wearables has been mainly searched and that being a combination of a ring scanner and a wrist-mounted computer. This complies to the hands-free requirements mentioned earlier by only needing interaction with the wrist-mounted computer, after a part of the process has successfully been executed and therefore leaving the worker with empty hands anyway. But if that is still not enough, since some processes might need more interactions than the ones planned for here, a headset can be added to the combination, allowing for voice commands when needed. Wearables in this category are:

- Zebra WT6000 + RS6000
- Honeywell Dolphin 75e + 8620 Wearable Ring Scanner

Furthermore there are not a lot more wearables that do fit the given requirements. Wearables like smartwatches or various fitness trackers are existing, but do not fit the given requirements at all. A smartwatch could be used to replace the wrist-mounted computer, but for the given ringscanners above, the connection to the wrist-mounted computer is proprietary and therefore can not be used with any current smartwatches.

4 Research

This chapter includes the general research that has been done for the project, which includes the research about some given processes and wearables that could be appropriate for the processes. The CASE Tools chosen at the current state of the project will also be discussed and what they are used for.

4.1 Processes

There were process already modelled and made available in the LOGwear project. They were created by working students at the FHTenL and revised through customer meetings. The naming of the processes was also done by the working students. The processes existing at the time of this report were:

- Order Picking High Rack
- Goods Receipt and Put away
- Order Picking W3-5

The decision for a single process was quickly made in cooperation with the company supervisor. The processes will just be briefly explained here, as it is not as important to fully understand the processes that were not chosen.

Order Picking High Rack is a process where either a special vehicle is used to reach especially high racks or the rack itself is mechanized and the items needed for the order picking process is moving to oneself. This process has been discarded due to the nature of what should be created. A simulation that should model a real environment. Modelling a High Rack and usage of that would be too hard and would take too much time.

Goods Receipt and Put away was another process to be considered. The process itself is about getting an order from a customer, waiting for it to arrive and then afterwards putting the incoming goods away in a spot that was dedicated for it beforehand. This process was discarded as well. The modelling of it would be problematic due to the time delays in the tasks. Furthermore the process showed less potential for improvements with wearables.

Order Picking W3-5 was the process chosen to improve with wearables as multiple possibilities to improve the process were discovered. Also the processes seemed rather easy to model with a demo facility by setting up one or two racks and placing multiple parcels on there. This process was selected together with the company supervisor. This process can be seen in the appendix, in figure A.1 but the demo scenario will also be further explained in section 6.2.

4.2 Wearables

Researching wearables was a more problematic task, as the amount of wearables that could be used potentially is a lot higher than the amount of given processes. Some criteria were set in place for a wearable to be considered in the first place.

4.2.1 Criteria

Further criteria are more specific towards the chosen process. The criteria were divided into requirements and quantifiable criteria. The requirements towards a wearable, or a combination of wearables, are the following:

Scan ID
The ability to scan an Identifier (ID), could be a barcode, Radio-Frequency Identification (RFID) code, Quick Response (QR) code or something different.

Informing User
The ability to give information to the user, this can be done by audio, visual or haptic feedback.

Send Confirmations
The ability to send confirmations to the WMS, that part of the process has been completed.

Hands-free
The ability to operate the wearable without the need to take a device and put it back all the time. Operations should not need user hand input while parcels are being handled or the hands are otherwise occupied.

On top of these criteria that the quantifiable requirements were:

Performance
The performance of a device is important as some wearables considered are not necessarily intended for the tasks in a warehouse and therefore some tasks like scanning different kinds of barcodes might be more problematic and need a lot of time if the device is not performant.

Cost
The cost is a factor to consider when the company possibly needs to order a large amount of these wearables.

Battery Life
A wearable should be able to sustain a whole day of working without the need to exchange batteries or swap wearables during a break. Worst case scenario would be to have the need to change the wearable device multiple times each day due to an empty battery.

Durability
The warehouse is not an environment where devices can be used that could break if a parcel graces or hits it.

Most of the quantifiable criteria were hard to actually quantify without having hands-on-experience with these devices. The reason behind this is, that most of the devices are either just released to the public or niche products leading to a small amount of information available besides information published by the manufacturers of the devices, which is generally not a good source of unbiased information.

4.2.2 Devices

Smartglasses seem to be the most promising type of wearable for the task, due to the possibilities it does give its user. The possibility for indoor navigation, scanning, exactly displaying the item location, constant display of information and further possible features that could be implemented using an always-on camera that is implemented in most devices (Schwerdtfeger, 2009).

The smartglasses most interesting for this topic are:

- Epson-Moverio BT-300
- Microsoft HoloLens
- Vuzix M100

They are the newest publicly available glasses from some of the biggest manufacturers of smartglasses, that are currently publicly available. This list has been created by using the initial list of smartglasses given in chapter 3 and then removing the ones that were either no longer available to the public, or were not released to the public at the start of the project.

The combination of a wrist mounted-computer and a ring scanner does fulfil the requirements and both items listed in chapter 3 are also available at the time this project was executed. Therefore the list here is the same as given already, being the following wearables:

- Zebra WT6000 + RS6000
- Honeywell Dolphin 75e + 8620 Wearable Ring Scanner

4.2.3 Decision

The wearables that were tested hands-on have been:

- Microsoft HoloLens
- Vuzix M100
- Honeywell Dolhin 75e + 8620 Wearable Ring Scanner

The other named wearables were also intended to be tested, but were not able to be delivered in a time that was considered acceptable to continue the project. Therefore the final weighted decision will only take these three wearables in consideration. The combination of Honeywell Dolphin 75e and the 8620 Wearable Ring Scanner will be counted as a single wearable for this comparison as they are intended to be used together. The general weighted decision for these wearables can be found in table 4.1.

The weights are distributed to sum together to the value of ten. The weights themselves were decided together with the company supervisor and were accepted. The value for a wearable is given from a scale from one to ten and the weighted value is the original value multiplied with the weight given for a criteria. For the criteria performance, battery life and durability, the higher the value of a wearable, the higher the performance, battery life and durability of the wearable are. For the cost of a wearable it is reversed, the higher the cost of a wearable the less points the wearable is getting in that criteria.

		HoloLens		Vuzix M100		Honeywell	
Criteria	Weight	Value	Weighted	Value	Weighted	Value	Weighted
Performance	4	6	24	2	8	8	32
Cost	2	2	4	5	10	3	6
Battery Life	2	4	8	5	10	9	18
Durability	2	3	6	3	6	8	16
	10		42		34		72

Table 4.1: Weighted Decision Wearables

The wearables were given a value between one and ten depending on how well they performed in the hands-on test and the information available beforehand.

From the weighted decision we can see that the Honeywell combination comes out on top, which is not as surprising, as it is a device that was specifically designed for this and similar purposes. The performance of the Honeywell combination is rated as highly, because the ring scanner is handling the scanning, which can be a more expensive task for ordinary computing devices. The HoloLens is in front of the Vuzix M100 mainly because of the integrated Holographic Processing Unit (HPU) that is handling the mapping of the environment and most of the data that is incoming from the different cameras. This allows the other components of the HoloLens to do different tasks with higher priority. Another factor here is that the HoloLens in general is using newer components that have a higher performance by themselves. The cost or price of the device is just the price of the devices.

The battery life between the HoloLens and the M100 is relatively similar, but the Vuzix solution has an included battery pack to increase usage time, which is also something that could be done with the HoloLens, but charging and using a device at the same time is not the nicest solution. Therefore the score is pretty low for both. When using a battery pack, both devices might get through a workday, but without both need to be recharged multiple times a day. The Honeywell combination is able to sustain through a day of work, and if needed, the battery here can be hot swapped. The durability of both smartglasses is similar in points, as both were not necessarily designed in the warehouse. While the HoloLens might break a bit faster than the Vuzix M100, at least from how rugged both look and feel, the M100 just needs a small push to no longer be in the right position to view the monitor. The Honeywell combination is by far the most rugged and also can not disposition as easily, the biggest problem might be the connection between the ring scanner and the wrist-mounted computer. While not likely that could tear or be plugged out.

But that might not be the only things that could lead to a decision, an aspect that is not explored here is possibilities. But the possibilities that a wearable has, are not that easily measured and therefore are ignored for the wearable comparison at this point in time. Therefore the Honeywell Dolphin 75e + 8620 Wearable Ring Scanner has been chosen as a wearable to implement the demo scenario explained in section 6.2.

4.3 CASE Tools

The CASE tools are used to implement parts of the project and have been decided on with the company supervisor.

Program	**Usage**
Azure Cloud	A cloud computing environment by Microsoft. In this project used to deploy the database for the mock WMS, the database connector and the Representational State Transfer (REST) interface to connect to it.
Bitbucket	Bitbucket is a tool developed by Atlassian. It is a Version Control System (VCS) solution that uses git as an underlying system and is aiming to give companies a secure and private git repository to use for proprietary projects. During the thesis Bitbucket is used as a VCS for the different parts of the project.
JIRA	JIRA is a tool developed by Atlassian. It is an issue tracking and project management tool. During the thesis it is used as a tool to plan sprints and track bugs and issues, as well as having a backlog of features that should still be implemented.
Mobility Software Development Kit (SDK) for Android	A SDK used to develop for the Honeywell Dolphin 75e. This is provided by Honeywell to allow usage of the unique components in their devices.
Netbeans Integrated Development Environment (IDE)	Is an IDE developed by the oracle corporation for Java development. Netbeans is also extensible by third party developers. In this project it is used to implement a web application using the JavaServer Faces (JSF) framework.
UMLet	UMLet is a free, open-source Unified Modeling Language (UML) tool. It is used to create UML diagrams and is extensible by user created, custom elements. (Umlet.com, 2017) It was used to create the diagrams that were created during the thesis.
Visual Studio	IDE developed by Microsoft for programming languages managed by them. For this project used to implement the database connection and the REST Application Programming Interface (API).

Table 4.2: CASE tools

5 Reference Model

The reference model of the logwear project is a model that shows how the communication from a wearable to the WMS or some other system might work. This does not mean that every communication necessarily needs to be implemented like displayed in the reference model, but that in general every wearable is able to communicate with an underlying system in the way it is displayed. The model can be seen in figure 5.1.

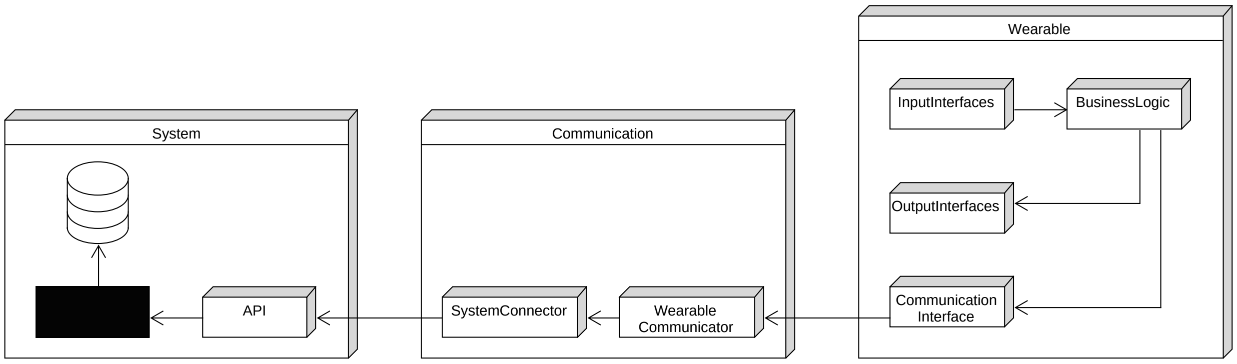


Figure 5.1: Reference Model LOGwear

5.1 Definition

A reference model is an abstract design used to help others understand the general concept and the relationships between existing entities of a specified environment. Furthermore a reference model, in general, does not model anything with a specific technology in mind and rather models everything as general as possible. This is done so that when creating an architecture around the reference model. The reference model can be used as a template to start working with. Not as a constraint, that holds the architects back. When implementing the reference model with a specific technology, it is needed to change existing parts or add new parts to fit the given constraints. A reference model as is, is not directly implementable due to the abstract nature.

The aim for a reference model is to standardize the way how developers in the future implements an application in the given domain, regardless of the used technologies.(Oasis-open.org, 2017)

5.2 Requirements

The reference model is an abstract construct, therefore the functional requirements towards it are taken as example to prove the validity of the model. The functional requirements are listed in the form of use cases The non-functional requirements are towards the model itself and not towards an implementation of it.

5.2.1 Functional Requirements

The functional requirements represents example cases that should be possible to implement with the reference model. The cases that will be given do not need to be fulfilled all at once, but all of them should be possible with the reference model. Given in the following is a single use case in figure 5.2, furthermore a short list of descriptions for further use cases, while the other full use cases themselves can be seen in the appendix, section A.2. The example cases given here are based on the demo scenario chosen, but more use cases can be defined for different scenarios.

Use Case	**Order Confirmation**
Code	UC-W2
Package	Wearable
Actors	Picking Worker
Description	A picking worker equipped with a wearable is using an input interface to confirm an order.
Precondition(s)	The order picker has finished picking the order and wants to confirm that everything is correct.
Scenario	1. The order picker uses an input interface on the wearable to confirm the order. 2. The wearable is processing the request. 3. The wearable is sending the confirmation to the WMS. 4. The WMS sends, that the confirmation was successfully received. 5. The wearable displays the user, that the confirmation was received by the WMS.
Extensions	1.1 The wearable is able to check if the confirmation is allowed to be send. See UC-W3.
Exceptions	3.1 The wearable got an error when processing the request. The order picker is informed about the cause of the error and can try to fix it. 4.1 The WMS does not answer. The order picker is informed about that and can try contacting the IT.
Result	The order picker has confirmed his order.
Version	1.1 | Author | LUR

Figure 5.2: Use Case: Order Confirmation

Get Order
An order document is fetched and in some way displayed to the worker.

Order Control
When an order is being picked, the wearable is supporting the worker in counting the right number of parcels and picking the correct article in the first place.

5.2.2 Non-Functional Requirements

The non-functional requirements towards the reference model are the most important details that were taken in consideration when the reference model was designed. They can be seen in the following list:

Documentation
The reference model needs to be properly documented. The advantages and disadvantages of the design

need to be properly explained to a person that potentially wants to implement a system based on it.

Extensibility
The reference model needs to be extensible since some companies might have some more requirements towards their system than is intended for a general case. Adding functionality to the reference model should be possible.

Modifiability
The existing ideas of the reference model should be modifiable. Companies are going to use different wearables and infrastructure, and the reference model should be adjustable to fit a lot of possibilities without the need to create a completely new system architecture.

5.3 Design

As can be seen in figure 5.1 the reference model is divided into three different packages that all fulfil different responsibilities. First it will be described what this reference model can help to create. Therefore the general thought behind the whole model will be explained and afterwards the three packages `system`, `communication` and `wearable` will be explained on their own.

The concept is simple, a wearable is connected to a communication layer, that then again connects to a system, that could be anything, as long as it has an API. The arrows are not indicating an information flow but rather an instruction flow. The box that the arrow leaves does invoke an instruction in the box that the arrow points to. The sources of information are generally the `InputInterfaces`, and the incoming information is spread from there. Depending on the incoming information, actions are invoked.

The reference model is purposely minimalistically designed, to allow most infrastructures and wearables, to apply the model, with as few changes as possible. If for example a new WMS was used the only component that needs to be changed is the `SystemConnector` in the `communication` package. In that case, the wearable that are in use, can also continue to work, just like they normally would, without the need for a new version to be deployed on all of them.

5.3.1 Wearable

The `wearable` package is representing the actual physical wearable, or a set of wearables, that a worker is using. This could be either smartglasses, smartwatches, a ring-scanner or some other wearable. It could also be a combination of wearables that is used in order to fulfil a certain task. In such cases it is still possible to stick to the reference model either by changing the existing model, with multiple `BusinessLogic` classes in the different wearables and a manager that could handle that. Or it could be, that even when using multiple wearables at once, a single wearable is handling the business logic of all wearables. Then the other wearables could be addressed by just their input- and output- interfaces.

Further on the wearable package is again held as simple as possible. The `InputInterfaces` are there to get information from the outside world and the `OutputInterfaces` are there to somehow represent information to the outside world. The `BusinessLogic` is there to process the incoming data and invoke the appropriate actions, that could be either displaying some information to the user or making a call to the underlying system through the `CommunicationInterface`.

The `CommuncationInterface` is the means of communication with an outside computer source, this could be radio, bluetooth, REST via Wireless Local Area Network (WLAN) or some other way of communication. The `WearableCommunicator` in the `Communication` package just needs to be able to receive and understand the messages.

5.3.2 Communication

The `communication` package is existing due to the different technologies used from wearables to connect to other computing devices. The communication layer might also be the only place, where information can be fully controlled by the developer, this is especially interesting when the `system` is controlled by a third-party. Also needed actions can be taken, if the incoming data has to be transformed into a different format, before either the `API` or the `wearable` can interpret it.

The `communication` package could potentially be removed, if the wearable has the needed technology in place to directly connect to the API of the given system. But this is in general not recommended, as the communication layer allows the developer to create a more standardized flow of information.

5.3.3 System

The `system` package here can be something like a WMS of the company, a system that is mostly a black box with an API and a database. Most of the time, the `system` cannot be changed by the developer, therefore the developer needs to use the possible ways to connect to the given API.

5.4 Variations

Multiple variations can be made to the reference model and some of the most likely variations of the reference model will be listed in this section.

5.4.1 No Communication Layer

As already mentioned in subsection 5.3.2 about the communication layer, the communication layer might be skipped in situations where the wearable is able to directly connect to the WMS. How the model changes in such situations can be seen in figure 5.3.

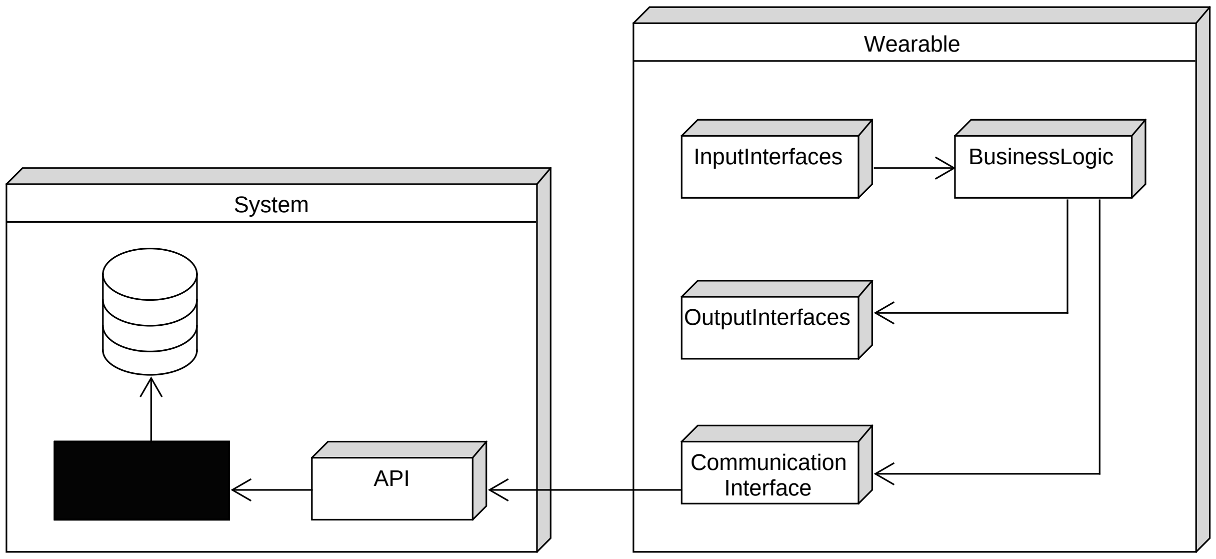


Figure 5.3: Reference Model - No Communication Layer

Apart from the removal of the communication layer the reference model still works exactly the same. This change is rather likely since a lot of newer wearables are able to connect to a rest interface or something similar.

5.4.2 Push Messages

The possibility to send push messages is also a very likely addition to the existing reference model. An example case for the usefulness of this is, an urgent order is coming in and a coordinator wants to see which person might be the most responsible for that order. Afterwards the coordinator can just contact the worker through his wearable and give him the new order directly through a push message. A possible configuration for this can be seen in figure 5.4.

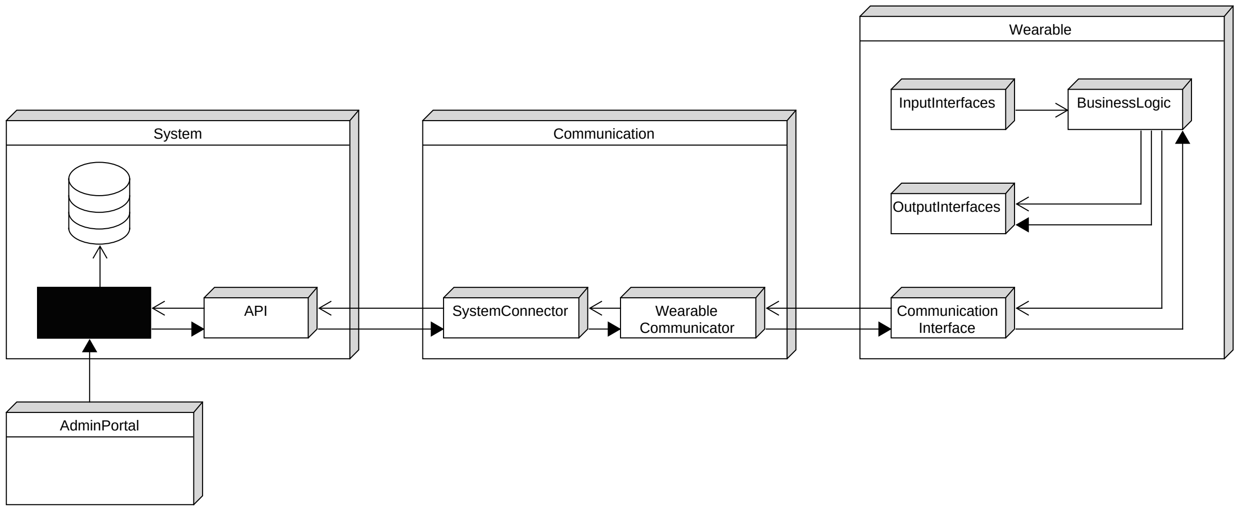


Figure 5.4: Reference Model - Push Messages

Like in the normal reference model, the normal arrows show an instruction flow that is originating from the worker. The arrows with the filled arrow head are the instruction flow from the coordinator and therefore the push messages. The AdminPortal is added here as a possibility for a coordinator to check on the state of current orders and what workers are currently most occupied. This helps the coordinator choose a worker that is available and used to the customer that ordered something with a higher priority.

5.4.3 Web Application

Another likely variation of the reference model is a scenario in which a web application is replacing the communication layer. This reduces the load on the wearable and allows a more uniform handling of multiple wearables. This variation of the reference model can be seen in figure 5.5.

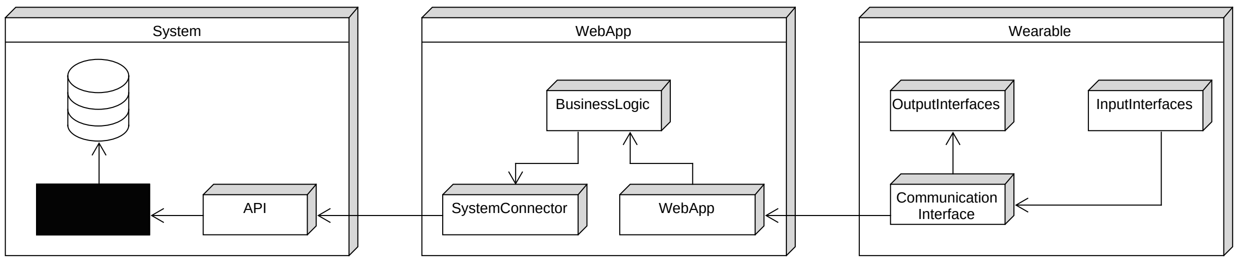


Figure 5.5: Reference Model - Web Application

The `WebApp` package here is just held as simple here, since the way to implement a web application can vary a lot. If a web application is implemented using the Model View ViewModel (MVVM) pattern, Model View Controller (MVC) pattern, Model View Presenter (MVP) pattern or some other pattern to implement web applications, is not relevant for the model and is therefore ignored here. The other parts of the reference model are mostly similar, the business logic is moved from the wearable to the web application. This is done to reduce the load on the wearable, which might be a problem for some wearables.

5.5 Problems

The main problem that occurred during the creation of the reference model was a problem in communication, regarding the names of reference model and reference architecture. The initial task was understood to create a general-purpose reference architecture, that would connect a wearable to some kind of system. During the process of creating the different diagrams for the reference architecture and trying to validate them using code. The problem became obvious that trying to go deeper than the now given reference model, as seen in figure 5.1, was impossible to do for a general-purpose implementation.

This is the case because of the nature of the given problem. Being able to use any wearable with any system, for any task. Given that two wearables might have completely different sets of input- and outputinterfaces those could not be defined. The communication interface could be different, which leads to being unable to define a communication standard, while the possibility of any task gives no single action that will always be the same.

6 Demo Facility

Physical creation of the demo facility that showcases the possibilities of the chosen wearable in the process of order picking in a warehouse. This chapter will therefore focus on the planned infrastructure, the existing design and what is planned for the demo facility in the future. As the implementation would mostly just be a repetition of what is discussed in the design section, there is no particular focus on the implementation details.

There is also a major difference between the demo facility task that will be explained in this report and the task given in the logwear website. (Logwear.eu, 2017) The task that will be executed here will be creating a demo facility in a physical sandbox environment and the task explained in the work package on the logwear homepage is about implementing the improved process, using a wearable, at a pilot company and observing the results.

6.1 Infrastructure

The infrastructure of the demo facility is divided into multiple areas:

Mock WMS
A mock WMS that allows to store different orders and edit them to have a more realistic demo. While not all functionality has to be given, the data has to be persistent and easily resettable to allow showcasing a demo case multiple times. The mock WMS is deployed on an azure server and is using a Microsoft Structured Query Language (MSSQL) database.

Rest API
A REST API that is used to connect to the mock WMS from an outside perspective, in this case from the communication layer, see subsection 5.3.2.

Communication Layer
The communication layer will be deployed on a server with a connection to the Database (DB) API and a connection to the wearable.

Wearable
The wearable will need to connect to the communication layer and process input and output.

The mock WMS and the REST API are insignificant parts of the implementation, therefore not a lot of thought was put into the decision on choosing these technologies was done out of curiosity for these technologies or being the most comfortable with them respectively.

6.2 Demo Scenario

The demo scenario explains the process that will be executed during the demo facility. Therefore describing the actions taken in detail, but ignoring how they will be executed, e.g. with or without a wearable. The original process can be seen in the appendix, figure A.1. For the actual scenario used for the demo facility an activity diagram was created, that can be seen in figure 6.1.

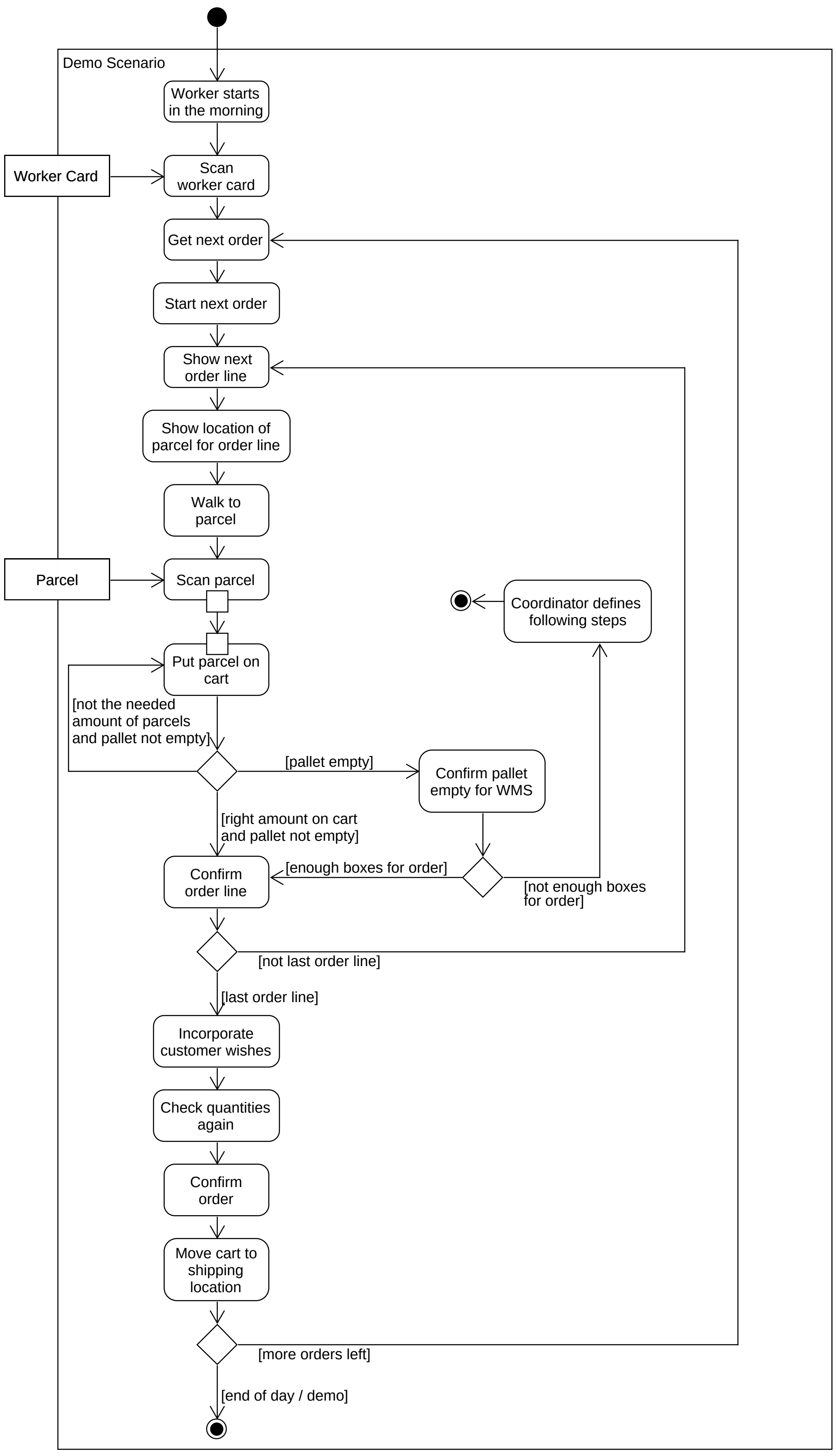

Demo Scenario
Worker starts in the morning
Worker Card
Scan worker card
Get next order
Start next order
Show next order line
Show location of parcel for order line
Walk to parcel
Parcel
Scan parcel
Coordinator defines following steps
Put parcel on cart
[not the needed amount of parcels and pallet not empty]
[pallet empty]
Confirm pallet empty for WMS
[right amount on cart and pallet not empty]
Confirm order line
[enough boxes for order]
[not enough boxes for order]
[not last order line]
[last order line]
Incorporate customer wishes
Check quantities again
Confirm order
Move cart to shipping location
[more orders left]
[end of day / demo]

Figure 6.1: Activity Diagram Demo Scenario

The model shows the original process in a simplified form. It also shows a complete work day and not just the process of picking a single parcel. The worker arrives in the morning and scans their worker card, this is done to log in the worker into the system and be able to get the orders that are allocated to that worker. The next thing is to get the order document, this is currently done on paper, but the intent for this is, that in the end this can be displayed on the wearable. The worker can then at first look at the order document and then continues to start working on that order.

The order is split up into multiple order lines. An order line contains a single article, location, quantity and a number to identify the order line. The worker identifies the location and starts moving to the article location. At the article location the parcel is scanned and afterwards put on the handcart of the worker. This is repeated until the quantity listed in the order line is met. When the quantity of the order line is met, the order line is confirmed, but there are multiple things that might happen during this process:

right amount on cart and pallet not empty
The process is working as explained above and the order line is confirmed without a problem.

right amount on cart and pallet empty
The pallet is confirmed empty for the WMS, but otherwise the process is working as explained above and the order line is confirmed.

too little amount of parcels on cart and pallet empty
The pallet is confirmed empty for the WMS, afterwards a coordinator is contacted to define the following steps, the process as planned is ending at this point and the steps as defined by the coordinator are executed.

When the order line is successfully confirmed the next order line can be worked on, if the current order line was not the last one of the order. When the order line was the last order line the customer wishes can be incorporated. The quantities for the order will then be checked again and afterwards the order is confirmed. The handcart is then moved to the shipping location. When the order is delivered to the shipping location the next order can be started. If this is the end of the working day for the worker, the process ends.

This diagram was created for multiple reasons:

Readability
The scale of the original process diagram made it hard to process. One goal was to create a more compact diagram. This was realized by designing it with the UML 2.5 notation as specified by the Object Management Group (OMG). (OMG, 2015)

Purpose
The original diagram had a different purpose, it was supposed to model the process of one of the pilot companies in its entirety. The purpose of the activity diagram is to model a demo case, that does not need to show every single detail of the process in the first place.

Focus point
The demo scenario focuses on the general tasks an order picking worker is doing, but is just focussing on the main points for this. The original diagram also includes the connections to the database and also includes tasks outside of the actual order picking.

6.3 Design

The general design for the demo facility application is based on the reference model described in chapter 5. The more concrete model for the demo facility can be seen in figure 6.2.

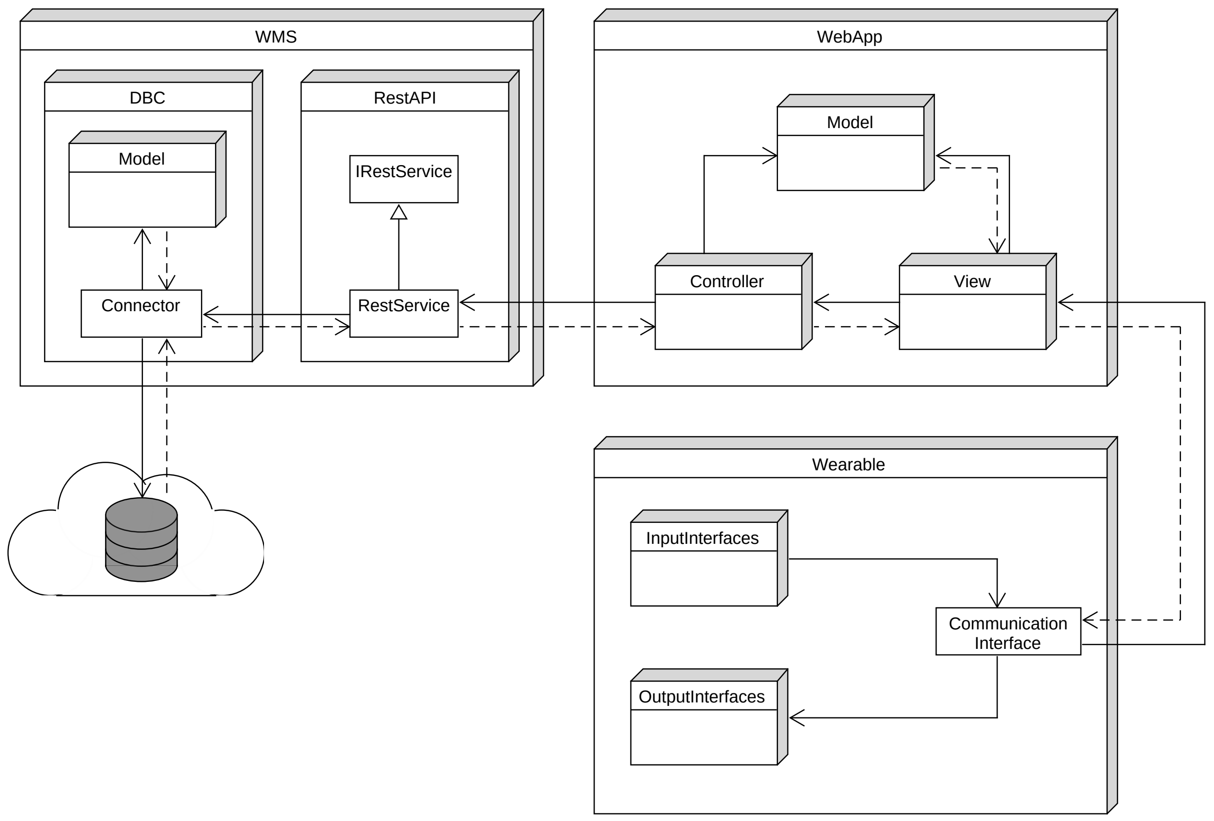


Figure 6.2: Package Diagram Demo Facility

The package diagram displayed is a high level view on the demo facility. The biggest difference to the reference model is the wearable, which in this case uses a thin client to connect to a web application that is doing most of the work for the wearable. The web application then in turn connects to the WMS and is replacing the communication layer that is existing in the reference model.

The normal lines represent function calls and the dashed lines represents returned information in the diagram. The following sections will go into detail for each of the displayed packages and the demo facility design will be elaborated further.

6.3.1 WMS

The Warehouse Management System consists of two parts, the database that is going to contain the data for the demo facility and the database connector. This also defines the interface, with which to connect to the WMS.

Database

Figure 6.3 shows the relations and fields in the database. It can be seen that the database just contains a small amount of information, due to being a demo, an actual WMS would contain a lot more data. The most important item in the model is the order, as that is the key piece, where most relations lead together. An order has a number that is connecting it to one or multiple workers that are working on them. Furthermore an order consists of multiple `OrderLines`. An `OrderLine` is describing the different lines that would appear on an order, that specify the item and the amount for an order. For a warehouse it is also important to add the pallet where to find the item and if the current line is already acknowledged or not. A pallet has a location in the warehouse and how much of an article stored on the pallet is still available in the warehouse. The article corresponds to a name for the article number. Finally an order is ordered by a customer, a customer might have additional wishes for their orders and an address where that customer wants things delivered to.

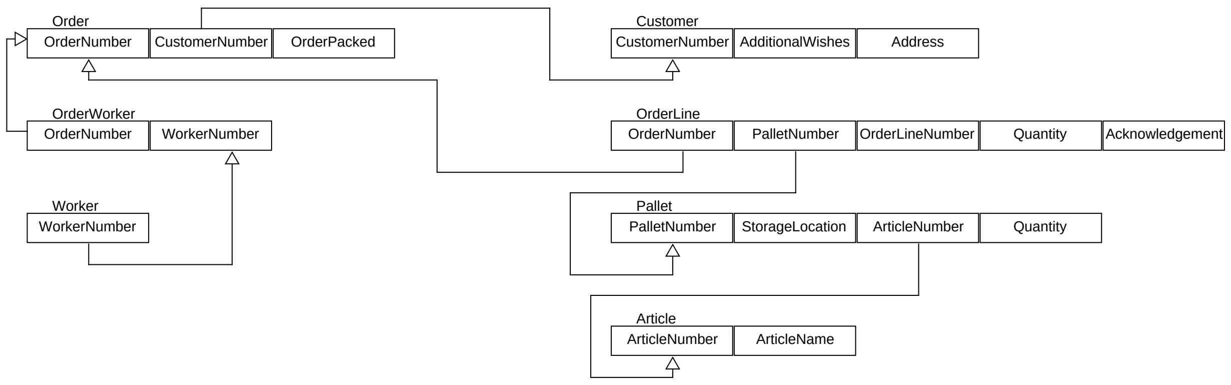


Figure 6.3: Relational Schema Warehouse Database

For an actual warehouse the customers attributes would change completely, as companies might have multiple addresses, therefore that might change for every order of that customer, making it easier to add an address field to the order and not the customer. Also a customer might want to add additional wishes just to a specific order or dependent on what might be ordered, therefore the additional wishes might also be moved to the order, but for a demo case, that is not executed at a pilot company, the model is sufficient.

Interface

The interface, that is exposed by the WMS, is a REST interface. The options available through the REST interface are the following:

GET Order
Returns the order with the id that is given as an argument. Takes an ID as an argument.

NextOrder
Returns the order with the smallest number for a specific worker. Takes an ID as an argument.

POST ConfirmOrderLine
Confirms that an order line in an order has successfully been picked. Takes an order ID and an orderline ID as arguments.

ConfirmOrder
Confirms that a complete order has been picked. Takes an ID as an Argument.

PUT ResetDatabse
A function that exists purely to have an existing and repeatable demo case. This resets the database to a state it was in at the beginning, before the demo was executed. Takes no arguments.

6.3.2 Wearable

The wearable is implemented with a thin client. This has multiple advantages, especially for such a demo case where potentially multiple wearables are supposed to be tested. One advantage is, because wearables tend to be less powerful devices just due to their size, the thin client allows to ease the load that would normally be computed on the wearable. But the bigger advantage is, that this allows to employ multiple wearables more easily. A thin client is implemented faster and easier than a fat client, therefore multiple demos for multiple wearables are more easily implemented. This allows to deploy multiple demos a lot easier. At the same time this allows a company a smoother upgrade process, meaning if they ever want to replace their existing wearables to another set of wearables, the transition could be realized a lot cheaper and faster.

Therefore apart from the usage of the input and output information to get information from the user and return information to the user only a communication interface is used to

6.3.3 Web Application

The web application is split into multiple web pages that leads a worker through the day. The first window shown is a page, where the worker is prompted to log in by scanning their worker id. Then the next order for that worker can be started. Then order lines are displayed in more detail and can be confirmed. If every order line is confirmed the order can be confirmed and the next order can be started. The mockups for these web pages can be found in the appendix in section A.3.

From a technical perspective, the web application will be implemented using Java and the JSF framework. JSF is a part of the Java Platform, Enterprise Edition (JEE) platform and is the current standard to implement web applications with Java. Furthermore JSF is a framework that is using the MVC pattern to deploy web applications.

This was chosen as the framework to implement the web application due to familiarity with the technology and it being a framework that allows rather easily to implement multiple web pages and connect that to a controller in the back end.

6.4 Implementation

This section will not discuss everything that is a part of the implementation, but will focus on points of the implementation that did not necessarily went as expected or are additions on top of the initial design due to problems that occurred later on. Also further information on the quality management will be discussed in later parts of this section.

6.4.1 Implementation Details

Most of the implementation is working as planned and is allowing the worker to go through his workday. The most interesting parts here are currently the connection between the web application and the database in the backend. As the database connector and the rest service there are implemented in C# and the web application is implemented using Java. When order information is to be send to the web application, the database backend is first creating an order model object. This model has been created using the Entity Framework (EF) provided by microsoft that is automatically creating the classes for the model from the given database information. This is creating the classes according to the official microsoft C# language specifications. (Microsoft, 2017)

The classes are afterwards serialized using the `JavaScriptSerializer` to serialize an order object into the JavaScript Object Notation (JSON) format. The JSON message is then send to the client that asked for the order information. The simple implementation for this can be found in listing 6.1.

```
var serializer = new JavaScriptSerializer();
var order = Connector.GetOrder(value);
HttpContext.Current.Response.ContentType = "text/HTML";
HttpContext.Current.Response.Write(serializer.Serialize(order));
```

Listing 6.1: C# Object Serialization

The client is then implemented in Java in the web application and trying to deserialize the JSON received from the backend. In Java the jackson library has been used to create the state of the order object from the received message (FasterXML, 2017). The first problem here is that the JSON that is expected from jackson is different, than the `JavaScriptSerializer` is creating. Therefore some transforming needs to be done to the JSON, otherwise the objects state can not be reproduced. Class and variable names that jackson expects need to be in camel case, otherwise it does not accept them, a method needs to be created that is automatically transforming those to camel case. When receiving a JSON message it is in the format displayed in listing 6.2 and the expected format can be seen in listing 6.3. The message displayed in these listings has been simplified and is just there to show the different formats.

```
{
  "Customer" : {
    "CustomerNumber" : 1
  }
}
```

Listing 6.2: Received JSON format

```
{
  "customer" : {
    "customerNumber" : 1
  }
}
```

Listing 6.3: Expected JSON format

The method used to transform the JSON message is quite simple and could be improved in the future to increase performance, but for the testing purposes it is enough. Code for this can be seen in listing 6.4. For further information about what the method does the included javadoc can be read.

```
    /**
     * This method is taking the input string and is converting the beginning of
     * every string inside of the json into a lower case letter. This
     * effectively is transforming every attribute and object into camel case.
     * What this also does is making every string attribute, that is representing
     * the state of an object, start with a lower case letter, which might not
     * be wanted by the user.
     * @param input The initial json string that should be transformed.
     * @return Returns the input json with every attribute in camel case.
     */
    public String jsonToCamelCase(String input) {
        StringBuilder stringBuilder = new StringBuilder(input);
        for (int i = 0; i < (input.length() - 1); i++) {
            char c = input.charAt(i);
            if (c == '"') {
                char nextChar = input.charAt(i + 1);
                if (nextChar >= 65 && nextChar <= 90) {
                    stringBuilder.setCharAt(i + 1, (char) (nextChar + 32));
                }
            }
        }
        return stringBuilder.toString();
    }
```

Listing 6.4: Java Code to Transform JSON

A few further things need to be changed in the json format, but that can be seen in listing A.1. That cod could also be improved a lot especially in regards to modifiability, but it is sufficient for a simple test case that should be created here.

For other cases it is just executing the commands exposed by the rest interface, that is detailed in section 6.3. The other implementation details needed are mostly simple web pages and setters and getters which do not need further explanation. The most interesting other design choice that was made for the model in the web application part was to implement a huge class `Order` that is containing multiple inner classes for each of the components. This can be seen in figure 6.4. This has been done since the jackson library is only available to deserialize an object if all of the nested classes are inner classes of the wrapping class.

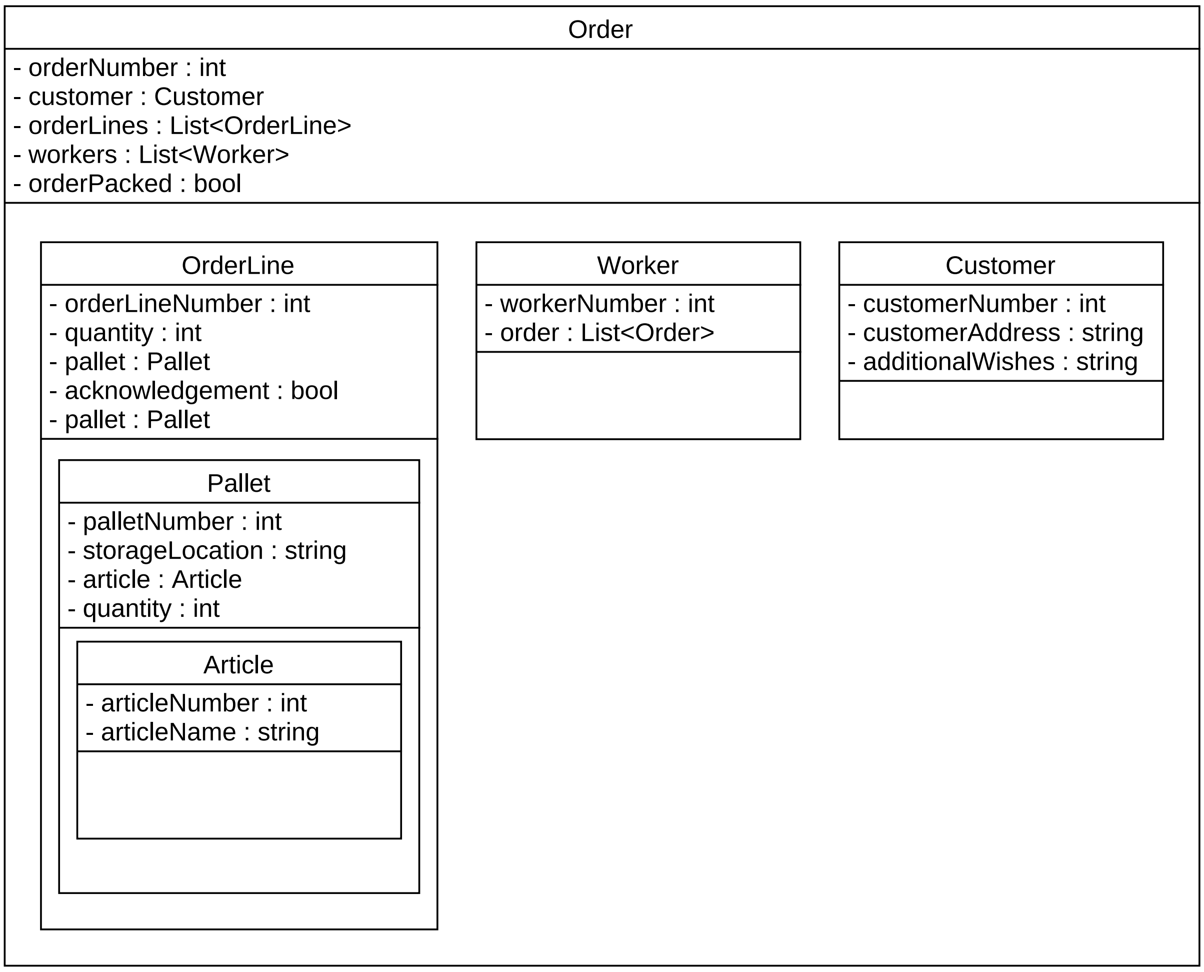


Figure 6.4: Class Diagram Web App Model Package

6.4.2 Quality Management

The quality management aspects have been described in section 2.4 and following this it will be explained how the different aspects have been looked at during the implementation.

Code Coverage

The code coverage percentage that has been aimed for during the project has been around 80% to 90%. The problem with code coverage is, that for java the coverage plugin is creating an error that could not be resolved during the time of the project. But every method is being called multiple times during the tests and therefore the coverage should be relatively high.

Language Conventions

The language conventions have been checked at their respective sources, in this case Microsoft for C# (Microsoft, 2017), and Oracle for Java (Oracle, 2017). Furthermore they have been followed in all parts of the application.

Documentation

The documentation for methods can already be seen in the listings 6.4 and A.1, furthermore the documentation has been created in similar fashion for all methods.

Performance Testing

Since the web application has already been created to reduce the load on the wearable there will most likely not be any problems of the performance of the wearables. Therefore this aspect of the quality management has not been conducted, as already explained in the description in section 2.4. Performance testing would have only been done if an issue would be likely or there would be some problems existing in regards to performance, but none of that has revealed itself during the implementation phase.

7 Conclusion

Up to this point, the reference model has been fully designed and is documented with multiple variations for different situations. The database for the demo has been deployed on an azure cloud service. The REST interface to that database is deployed on the same azure instance. The web application is still being developed at this point, as well as the connection from the wearable to this web application. The facility area has also not been rented and filled at this point.

Concluding the whole tasks that are to be finished at the end of this project is, that the research process for the available wearables is finished, but more wearables are going to be available in the future for the logwear project. Those wearables will be tested and compared in the future, but that will most likely not be a part of this project anymore. The requirements analysis and design phase for the reference model and the demo facility are also completely finished at this point. The infrastructure surrounding the demo facility is fully implemented as well. The two parts that are not fully implemented at this point are the web application and the wearable application itself. While there are still tasks to complete when that is finished as well, these are the main tasks that are a part of this project.

7.1 Recommendations

The recommendations are towards possibilities of the project in the future when the initial project of creating a demo facility is done.

There is still work to be done in the area of the demo facility after the initial one is done. There might be an internship or a bachelor thesis that is concerned with creating demos for each of the wearables available to the logwear project. The focus here could be for the research part on the possibilities of the different wearables for the same task. The information gained from the different demos can be used to compare the different wearables more effectively and on a level that was not possible beforehand. The design part could go more into detail on how the different wearables and environments are requiring different design patterns and ideas. The implementation is rather interesting again due to the multitude of environments and technologies used in each wearable. This type of project could be especially interesting for SME so that they get the ability to test a lot of different wearables out themselves.

Another interesting project for an internship or bachelor student could be the implementation of a process improved by wearables at a pilot company. The emphasis there could be on the differences between a sandbox style demo and a demo that has all the constraints that are imposed by the environment of the pilot company.

7.2 Further Planning

Apart from the recommendations given in section 7.1, this project still has a few tasks left, that will be handled. Further functionalities that might still be added to the web application:

- Dividing a room in multiple sectors to help new picking workers to more smoothly get used to their work. And help a worker to find the most optimal route through the warehouse.
- Implementing a mechanism that is helping the picking worker to confirm the quantity.

Apart from different functionalities that can still be added to the demo facility, there is still other tasks to be done. Further documentation for a handover and a document on how to set up the existing system on other systems. The area for the demo environment needs to be prepared before the demo can actually be showcased.

When all that is done, the demo can still be actually showcased to interested SME, to show the advantages of wearables in the order picking process.

Bibliography

Canders, Sascha (2017). “Development of a web application backend in a volatile environment”. Bachelor Thesis. Fontys Hogeschool Techniek en Logistiek.

DHL (2015). *DHL successfully tests Augmented Reality application in warehouse*. [online] Available at: `http://www.dhl.com/en/press/releases/releases_2015/logistics/dhl_successfully_tests_augmented_reality_application_in_warehouse.html`. [Accessed 25 March 2017].

FasterXML (2017). *Jackson Project*. [online] Available at: `https://github.com/FasterXML/jackson`. [Accessed 3 July].

Logwear (2017). *Order Picking Process Diagram*. [online internal].

Logwear.eu (2017). *LOGwear Arbeitspakete / Werkpakkete*. [online] Available at: `http://logwear.eu/`. [Accessed 25 March 2017].

Microsoft (2017). *C# Language Specification*. [online] Available at: `https://msdn.microsoft.com/en-us/library/ms228593(v=vs.80).aspx`. [Accessed 22 March 2017].

Oasis-open.org (2017). *OASIS SOA Reference Model*. [online] Available at: `https://www.oasis-open.org/committees/soa-rm/faq.php`. [Accessed 22 March 2017].

OMG (2015). *Object Management Group (OMG) Unified Modeling Language Version 2.5*, 371ff.

Oracle (2017). *Code Conventions for the Java TM Programming Language*. [online] Available at: `http://www.oracle.com/technetwork/java/codeconvtoc-136057.html`. [Accessed 3 July].

Sander, Oliver (2017). “Development of a web application in a volatile environment”. Bachelor Thesis. Fontys Hogeschool Techniek en Logistiek.

Schwerdtfeger, Björn (2009). “Pick-by-Vision: Bringing HMD-based Augmented Reality into the Warehouse”. PhD thesis. Technische Universität München.

Umlet.com (2017). *UMLet Website*. [online] Available at: `http://www.umlet.com/`. [Accessed 25 March 2017].

Appendices

A.1 Order Picking Process

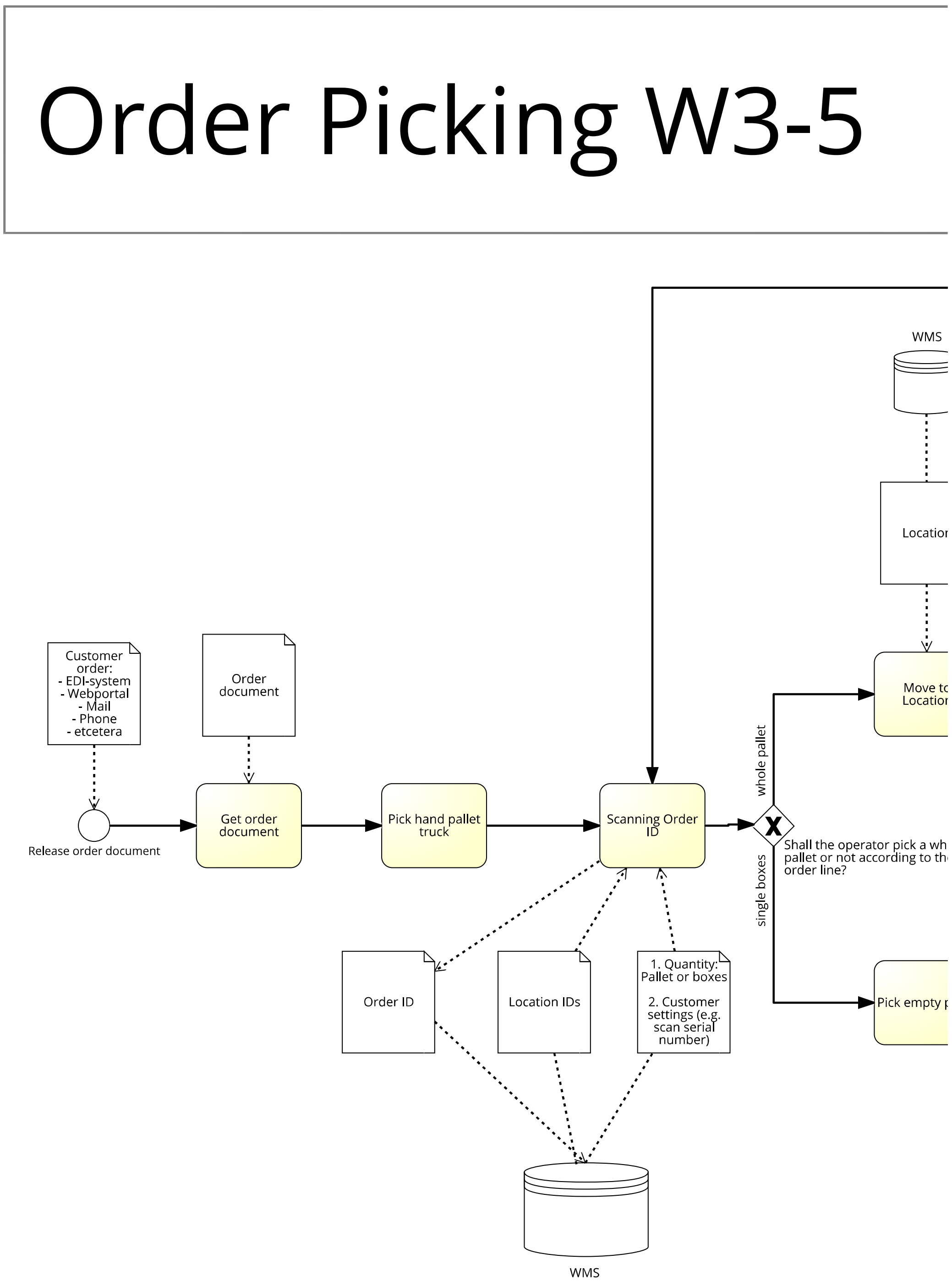

Order Picking W3-5
WMS
Location
Move to Location
Customer order: - EDI-system - Webportal - Mail - Phone - etcetera
Order document
whole pallet
Release order document
Get order document
Pick hand pallet truck
Scanning Order ID
Shall the operator pick a wh pallet or not according to th order line?
single boxes
Order ID
Location IDs
1. Quantity: Pallet or boxes 2. Customer settings (e.g. scan serial number)
Pick empty p
WMS

Figure A.1: Order Picking Process Diagram (Logwear, 2017)

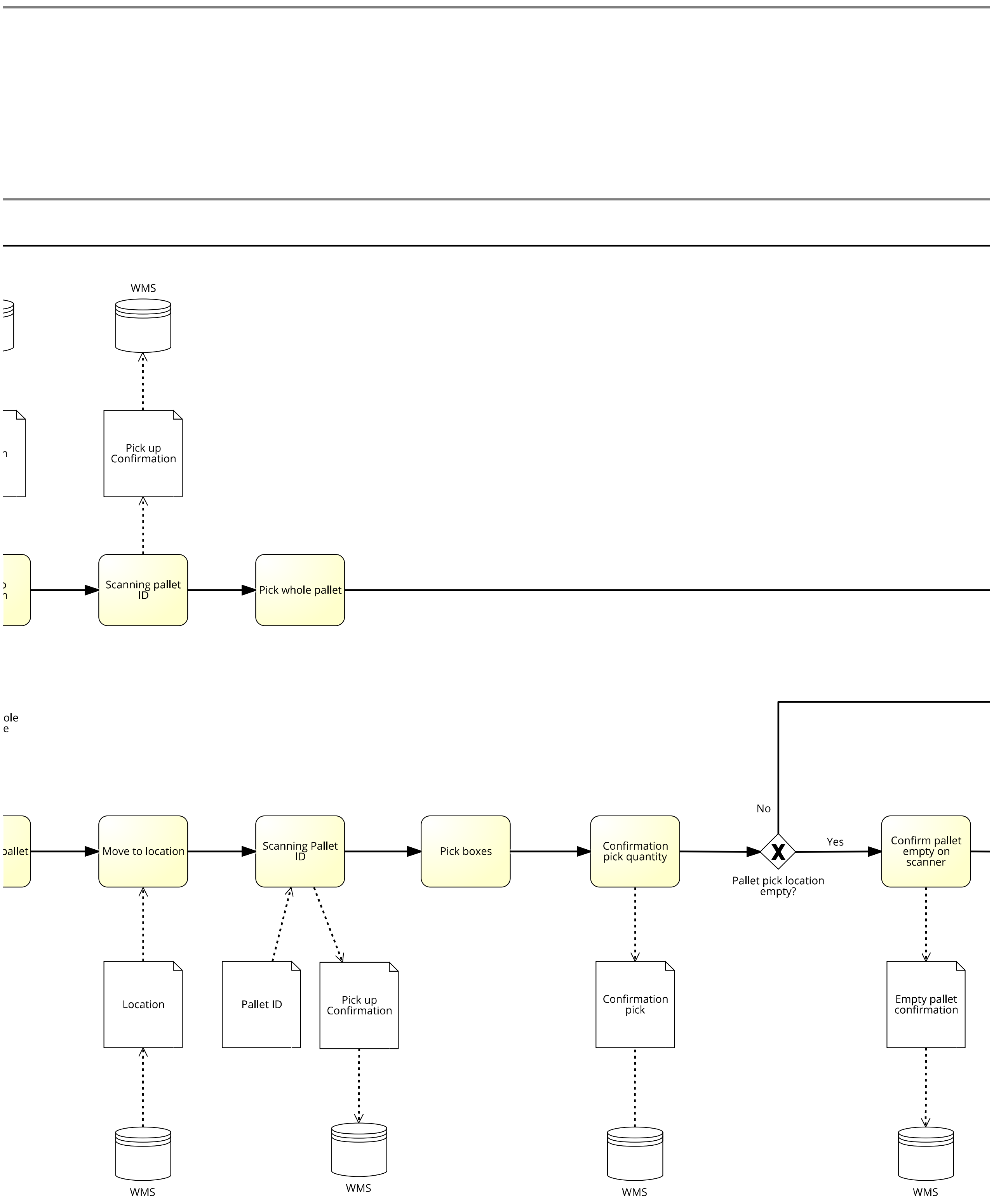


Figure A.1: Order Picking Process Diagram (Logwear, 2017)

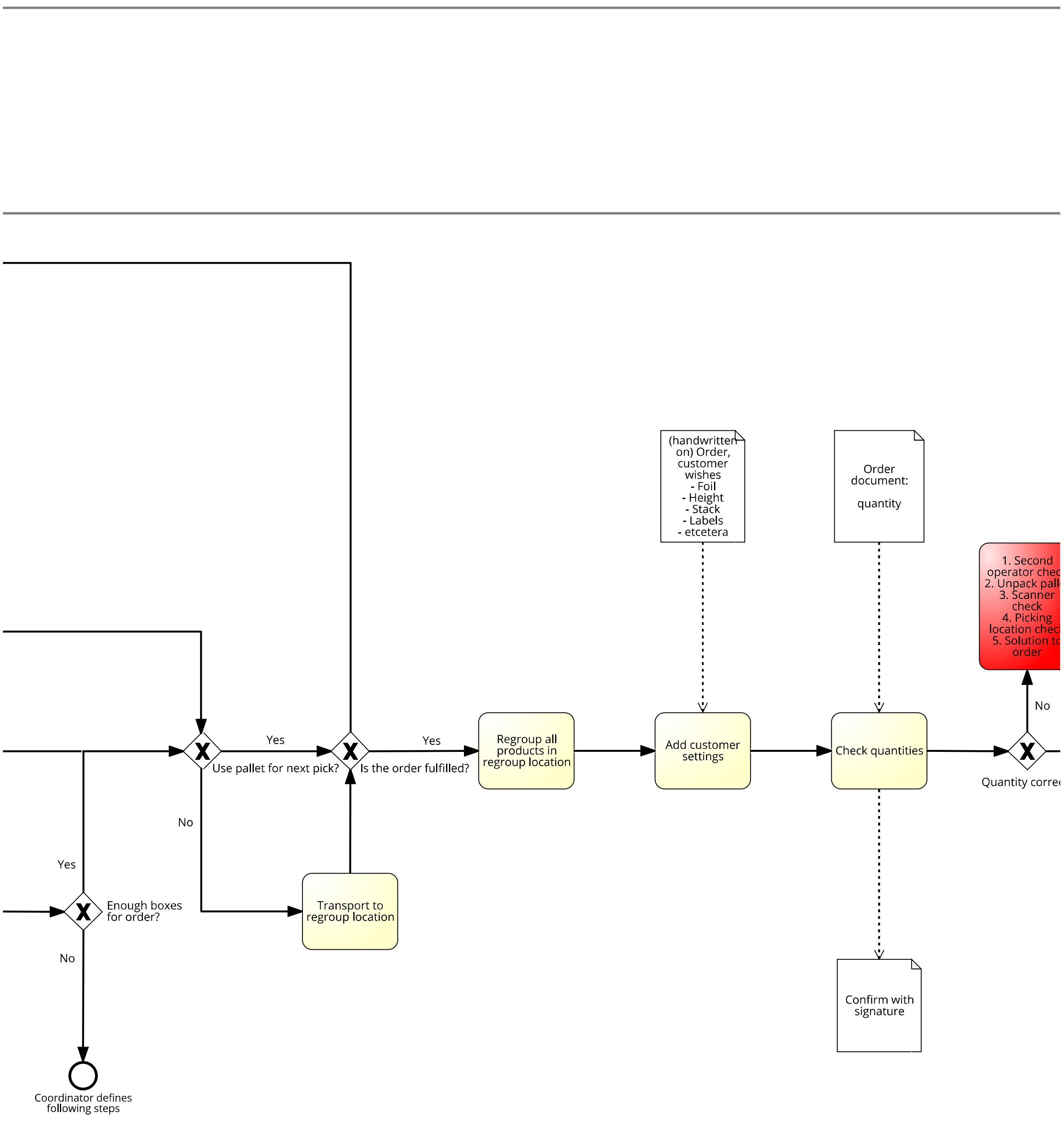
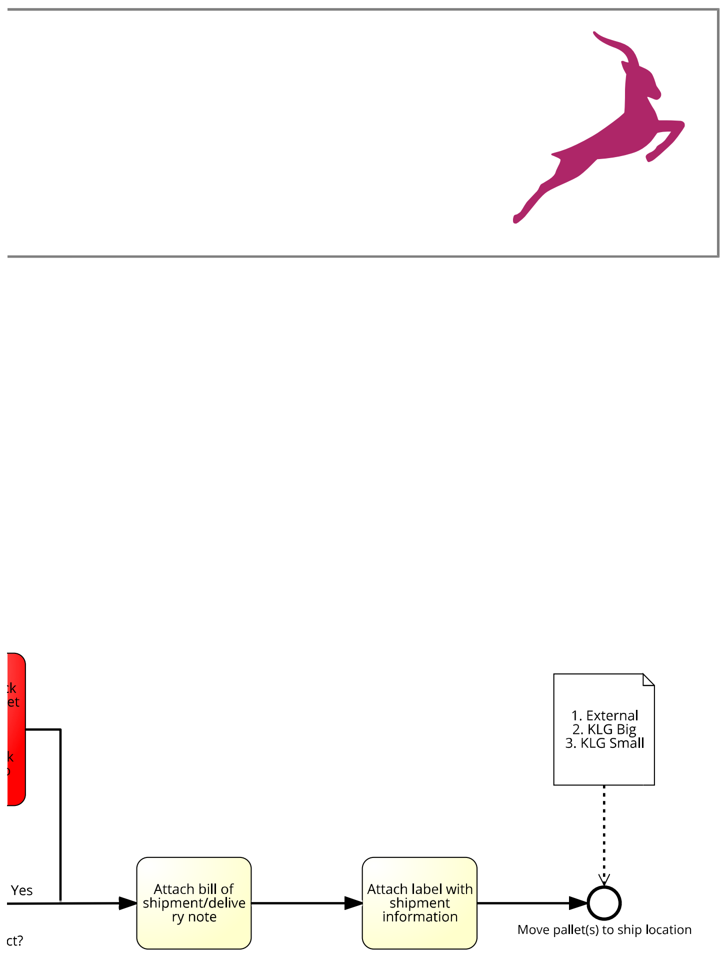


Figure A.1: Order Picking Process Diagram (Logwear, 2017)



Yes
ct?
Attach bill of shipment/delivery note
Attach label with shipment information
1. External
2. KLG Big
3. KLG Small
Move pallet(s) to ship location

Figure A.1: Order Picking Process Diagram (Logwear, 2017)

A.2 Use Cases Reference Model

Use Case	**Get Order (Voice)**		
Code	UC-W1		
Package	Wearable		
Actors	Picking Worker		
Description	A picking worker equipped with a wearable is using a voice command to get the next order displayed.		
Precondition(s)	The wearable has to be equipped with an Input Interface accepting voice and and an Output Interface that is able to return information to the user.		
Scenario	1. The Picking worker gives the voice command "Next Order". 2. The Wearable asks the WMS for the next order for the specific worker. 3. The WMS sends the Data to the Wearable 4. The Wearable displays the Data to the order picker.		
Extensions	-		
Exceptions	1.1 The Voice command could not be properly understood. The Wearable does nothing. 1.2 The Order Picker is already on an Order and that one is unfinished, the command is ignored. 1.3 Another Voice command is understood, that one is executed. 3.1 The WMS did not find a next order. The order picker is informed about that. 4.1 There is an error in the format that was received. The data that could be understood is still displayed and the order picker is informed, that there might be an error with the order.		
Result	The Order Picker has received the next order.		
Version	1.0	Author	LUR

Figure A.2: Use Case: Get Order(Voice)

Use Case	**Order Control**
Code	UC-W3
Package	Wearable
Actors	Picking Worker
Description	A picking worker is in the process of picking his order.
Precondition(s)	The wearable has a vision interface or is in another way able to control what the order picker is doing. Furthermore the order picker is currently in the process of picking an order and is picking a specific item of that order.
Scenario	1. The order picker scans an item. 2. The wearable checks what is being packed. 3. The order picker is putting the item on his hand pallet truck. 4. The wearable is counting the amount of items put onto the hand pallet truck. 5. The order picker scans a new item. 6. The wearable detects a new item and ends the counting process for the last item. 7. The wearable informs the order picker that everything went correctly with the last item.
Extensions	6.1 When the order has no next item, the worker tries to confirm the order and the wearable can then proceed to check if the amount of picked parcels was correct.
Exceptions	7.1 The order picker could have counted wrongly and the wearable is informing him about it. After the order picker has checked the quantity on the Truck, go back to 1. with the item started with. 7.2 The wearable could have counted wrongly and the wearable is informing the order picker as if he counted wrong. The order picker can check the hand pallet truck for the item and confirm the right quantity. 7.3 The order picker could have picked the wrong quantity and the wearable could have counted wrong, resulting in the wearable saying the order picker has picked the right amount. An error is made.
Result	The Order Picker completed a part of the order and the wearable confirmed the quantity of that part.
Version	1.1 | Author | LUR

Figure A.3: Use Case: Order Control

A.3 Web Application Mockups

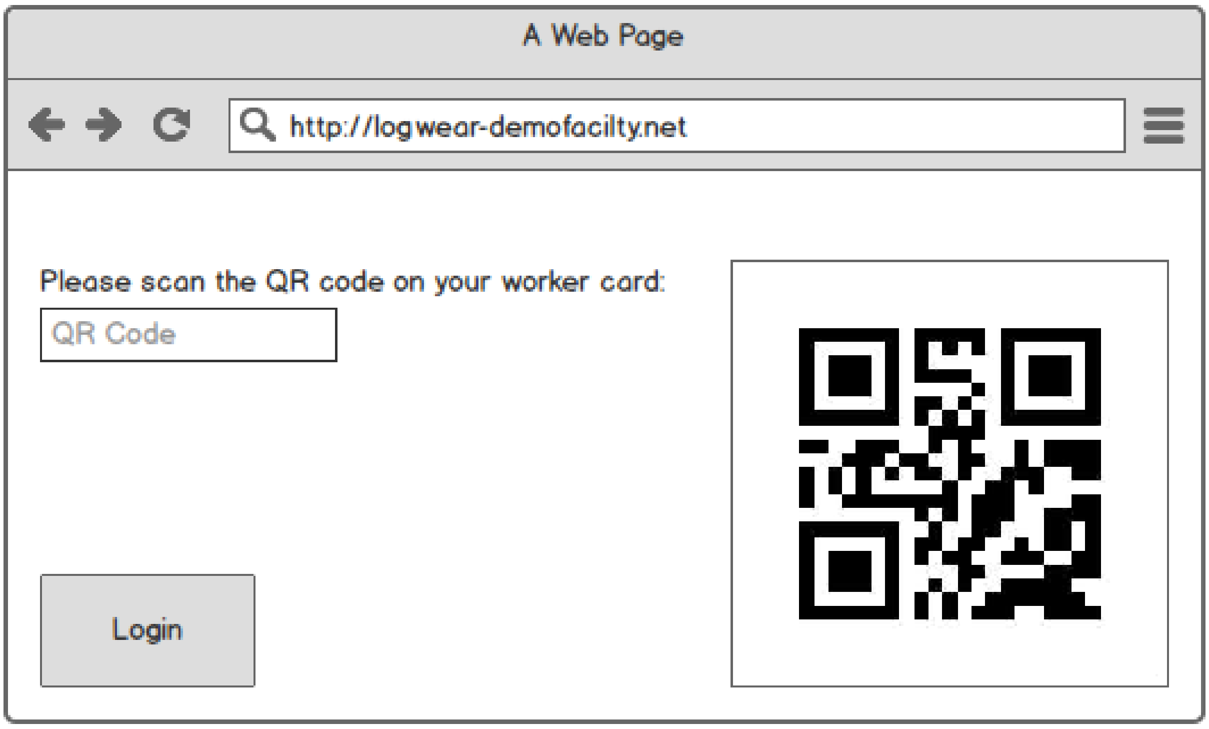

A Web Page
http://logwear-demofacilty.net
Please scan the QR code on your worker card:
QR Code
Login

Figure A.4: Mockup Worker Login

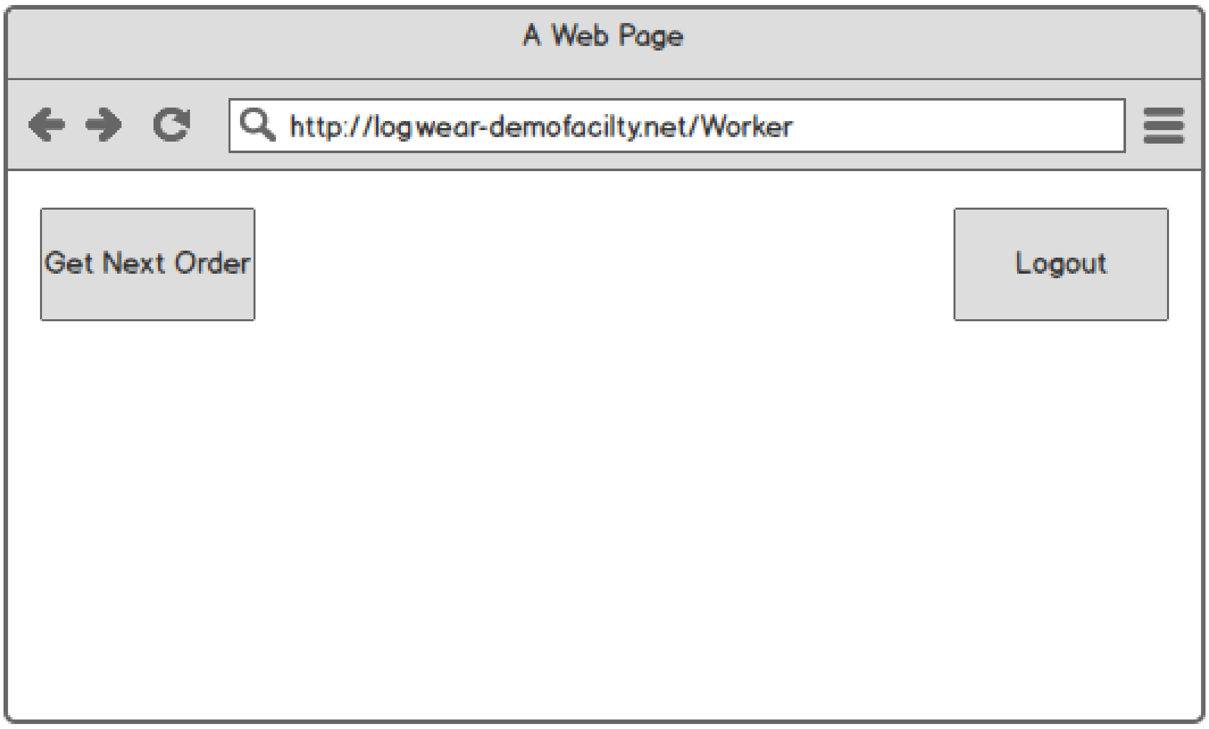
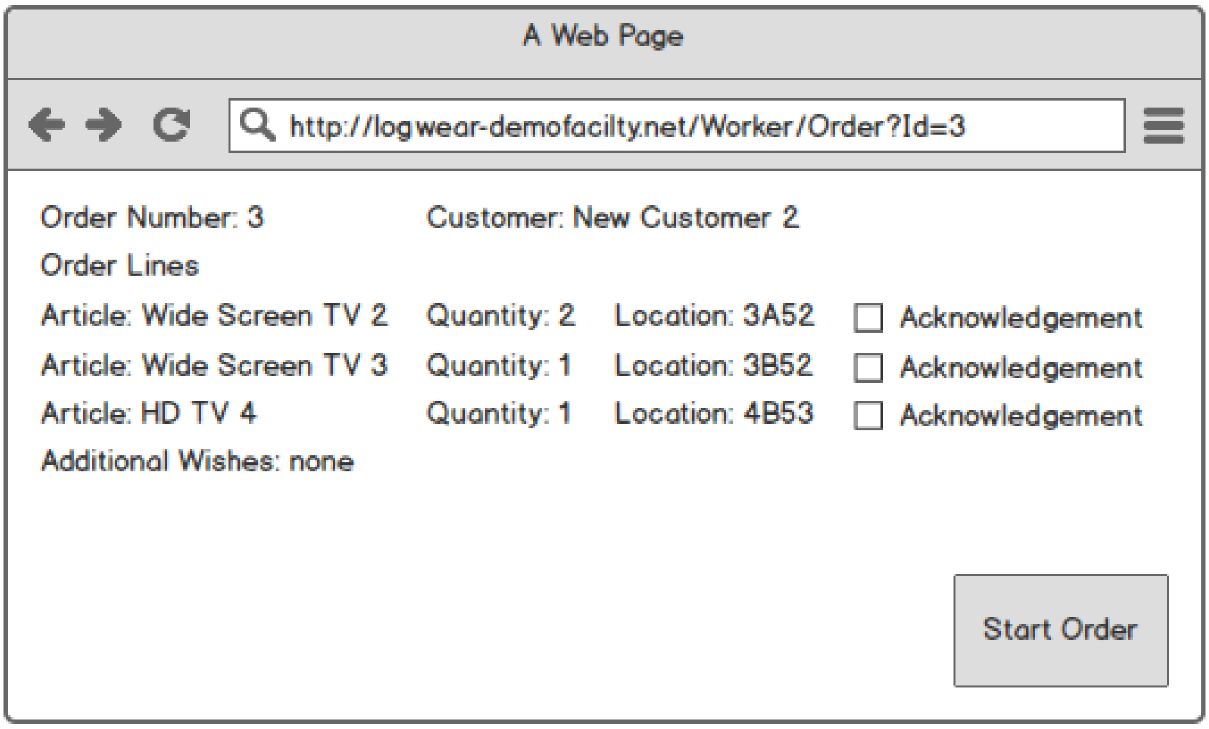


Figure A.5: Mockup No Order Started



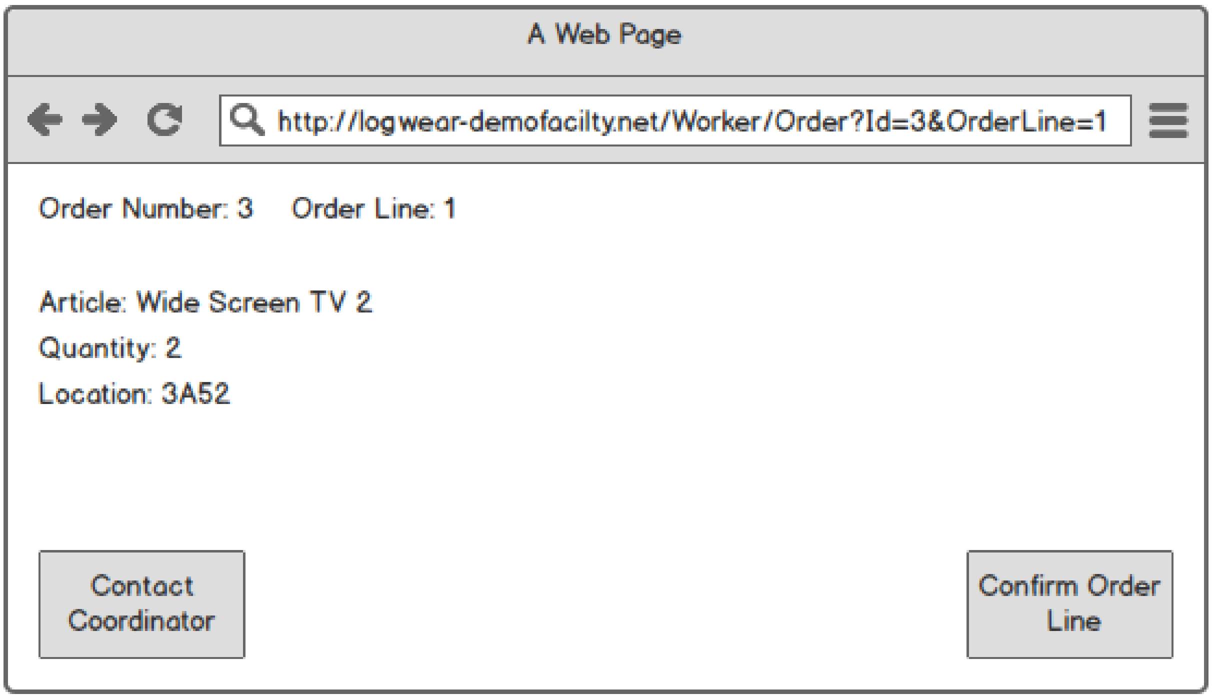
Order Number: 3 Customer: New Customer 2

Order Lines

Article: Wide Screen TV 2	Quantity: 2	Location: 3A52	☐ Acknowledgement
Article: Wide Screen TV 3	Quantity: 1	Location: 3B52	☐ Acknowledgement
Article: HD TV 4	Quantity: 1	Location: 4B53	☐ Acknowledgement

Additional Wishes: none

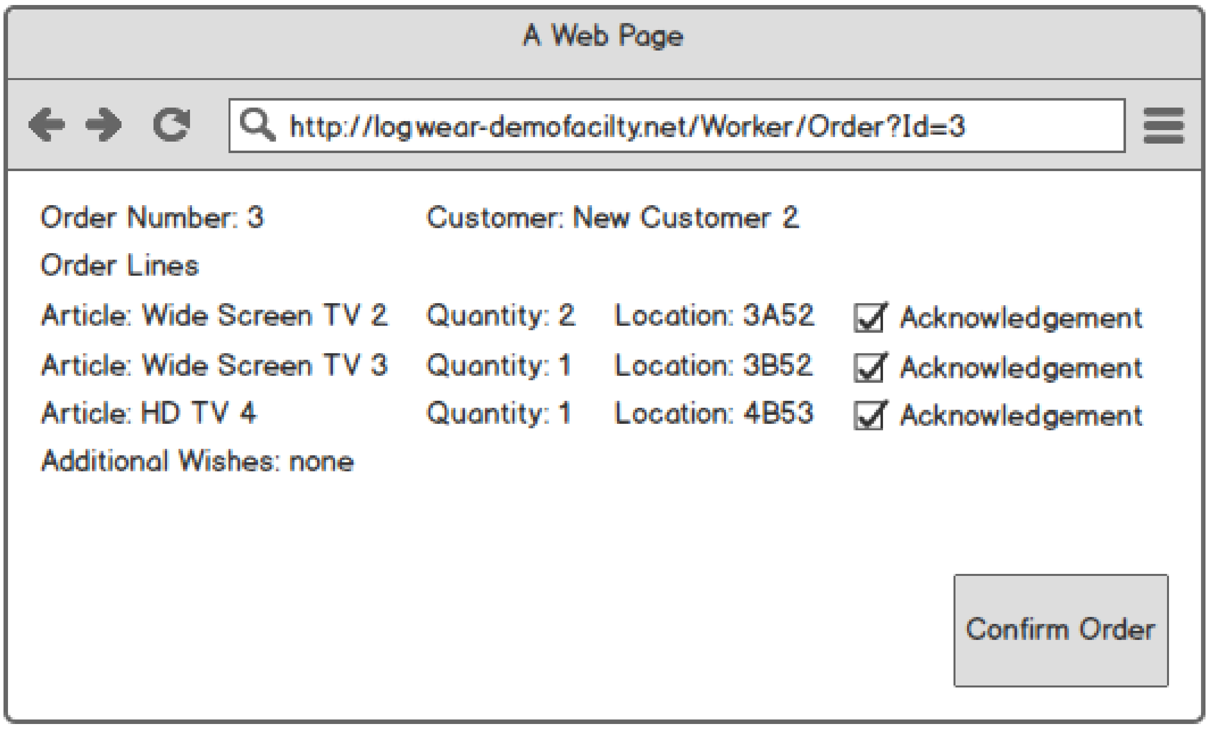
Figure A.6: Mockup Start Order


A Web Page
http://logwear-demofacilty.net/Worker/Order?Id=3&OrderLine=1
Contact Coordinator
Confirm Order Line

Order Number: 3 Order Line: 1

Article: Wide Screen TV 2
Quantity: 2
Location: 3A52

Figure A.7: Mockup Confirm Order Line



Order Number: 3 Customer: New Customer 2

Order Lines

Article: Wide Screen TV 2	Quantity: 2	Location: 3A52	☑ Acknowledgement
Article: Wide Screen TV 3	Quantity: 1	Location: 3B52	☑ Acknowledgement
Article: HD TV 4	Quantity: 1	Location: 4B53	☑ Acknowledgement

Additional Wishes: none

Figure A.8: Mockup Confirm Order

A.4 Implementation Details

```
/**
 * Is creating a connection to the rest service and is getting the next
 * order that the current worker should work on.
 * @return The next order for the current worker.
 */
public Order getCurrentOrder() {
    HttpURLConnection connection = null;
    URL restUrl;
    String orderJson = "";
    try {
        restUrl = new URL(baseUrl.toString() + "NextOrder/?id=" + workerID);

        connection = (HttpURLConnection) restUrl.openConnection();
        connection.setRequestMethod("GET");

        //Get Response
        InputStream is = connection.getInputStream();
        BufferedReader rd = new BufferedReader(new InputStreamReader(is));
        StringBuilder response = new StringBuilder();
        String line;
        while ((line = rd.readLine()) != null) {
            response.append(line);
            response.append('\r');
        }
        rd.close();
        orderJson = response.toString();
        orderJson = jsonToCamelCase(orderJson);
        orderJson = "{\"Order\":" + orderJson + "}";
    } catch (JsonMappingException ex) {
        Logger.getLogger(WorkerBean.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(WorkerBean.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        if (connection != null) {
            connection.disconnect();
        }
    }

    if (!orderJson.equals("")) {
        ObjectMapper mapper = new ObjectMapper();
        mapper.configure(DeserializationFeature.UNWRAP_ROOT_VALUE, true);

        try {
            currentOrder = mapper.readValue(orderJson, Order.class);
        } catch (JsonMappingException ex) {
            Logger.getLogger(WorkerBean.class.getName()).log(Level.SEVERE, null, ex)
;
        } catch (IOException ex) {
            Logger.getLogger(WorkerBean.class.getName()).log(Level.SEVERE, null, ex)
;
        }
    }
    return currentOrder;
}
```

Listing A.1: Java Code to Receive JSON from REST service