U. PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

NERDWIRED

Sistemas de Informação

Mestrado Integrado em Engenharia Informática e Computação

Ana Macedo - up201909572@fe.up.pt
André Moutinho - up201707291@fe.up.pt
Leonor Sousa - up201705377@fe.up.pt
Luís Fernandes - up201706910@fe.up.pt
Sérgio Dias - up201704889@fe.up.pt

Project Overview

Our project consists in the development of a **Mobile Application** to support the **management** of all the **logistic activity** related to **NerdWired**, a company that sells **computer components** to final customers.

The app contemplates the **purchase** and **sales** process, handles **stock replenishment** and **allocation** to **warehouse** and assists the **picking** process of a sales order.

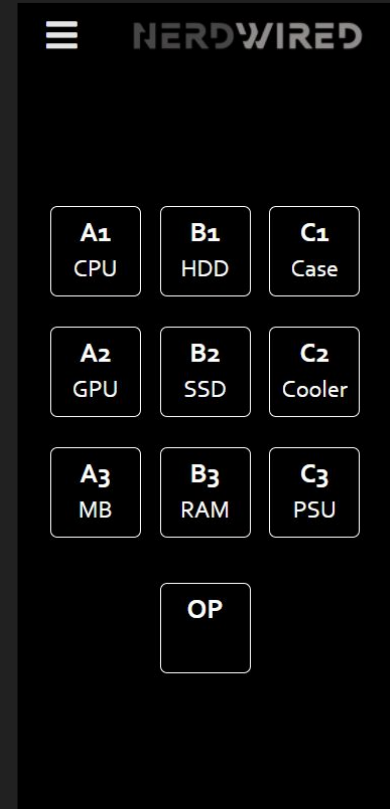It has mainly 2 types of users: **managers** and **pickers**.

Fig 1 - Home page (warehouse overview)

Context

When orders from **suppliers** arrive at the **warehouse**, their products are immediately stored in their respective **subsections**. As the customers make orders, the picker picks the respective products, based on **picking waves** that can aggregate products from different orders.
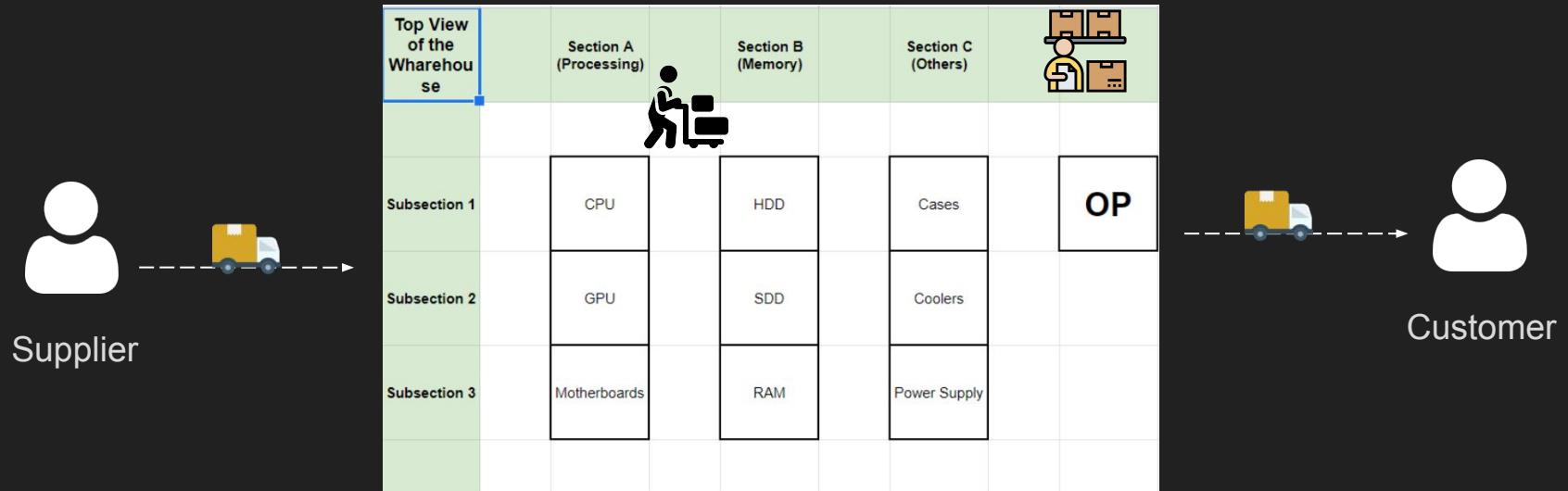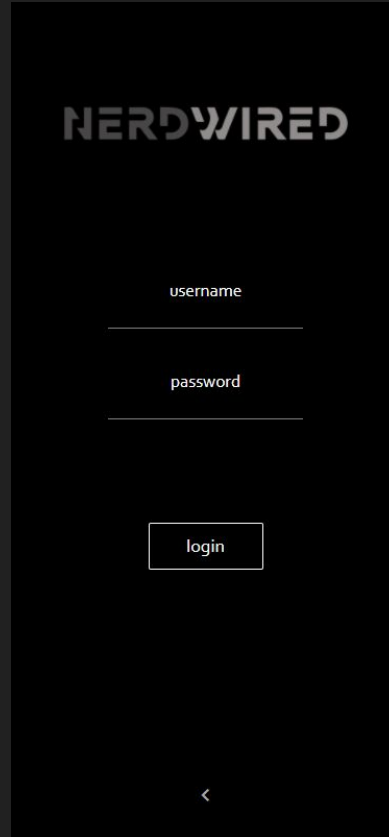

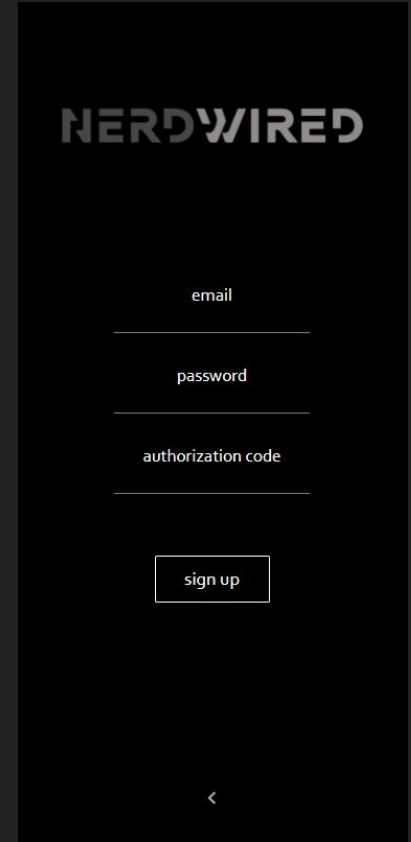Fig 2 - Business process scheme of the target company

Features

Authentication

Login - signup

- Email / password based;

- Firebase Authentication.

Fig 3 - Login screen

Fig 4 - Signup screen

Features

Authentication

User permissions - 2 different roles

Picker

Executes picking waves, **collecting** products that are ordered by clients and bringing them to the Out Point.
Inputs feedback throughout the **picking** process.

Manager

Triggers the generation of **picking waves** and assigns them to the pickers. Guides the picking process when necessary and reads the feedback provided by pickers, taking actions accordingly.

Features

Stock management

Listing

- List of **products** stored in each **warehouse section**;

- Information about **reference** and **stock** level of each product.

Fig 5 - Stock listing (per section)

Features

Stock management

Adjustment

- Once a **Picking Wave** is **finished**, the **stock** of each picked item is **updated**;

- **Supplier orders**' note are read and **stock** increases automatically.

Features

Orders

Listing

- List of **client orders**;

- List of **supplier orders**.

Fig 6 - Client orders listing

Fig 7 - Supplier orders listing

Features

Orders

Details

- Visualization of details such as **products** that are part of an **order** and their respective **ref** and **quantities**.
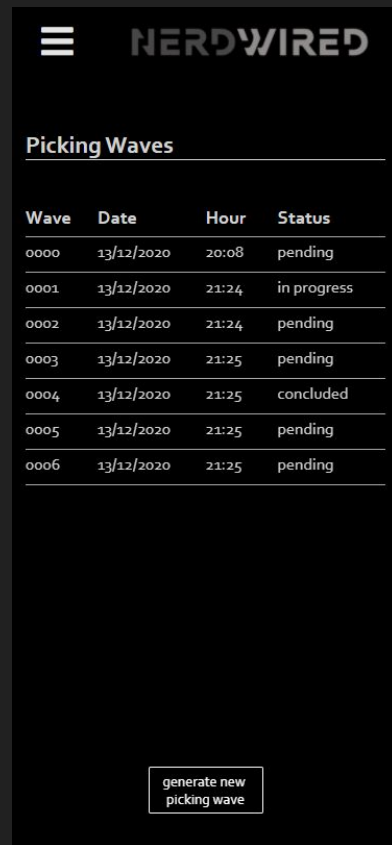

Fig 8 - Order details view

Features

Picking waves

Listing

- Visualization of the generated **picking waves**;
- **Manager** can visualize all **picking waves**;
- A **picker** can only visualize the **picking waves** that were **assigned** to them.

Fig 9 - Picking waves listing

Features

Picking waves

Assigning

- A **manager** can assign a specific **picker** to carry out a **picking wave**.

Fig 10 - Picking wave details (manager view)

Features

Picking waves

Details

- List of **products** that are part of the **picking wave**, grouped by warehouse sections;
- **Quantity** to pick for each item;
- The order that warehouses appear in indicates the **route** to follow.

Fig 10 - Picking wave details (manager view)

Features

Picking waves

Processing

- **Picker** can update the picked amount of each product as they execute the **picking wave**;

- **Picker** can submit a **textual note** if additional information needs to be provided regarding a **picking wave** (e.g. exceptions on inventory).

Fig 11 - Picking wave input screen

Features

Picking waves

Delivery note

- A **delivery note** is automatically generated when all items of an order are picked.

Goods receipt

- A **good receipt** is automatically generated when an **order** from suppliers arrives to the warehouse.

Features

Picking waves

Automatic generation

- The **managers** only have to input the amount of products for the **Picking Wave**. The **composition** of the Picking Wave is created **automatically**!

  It is done based on multicriteria, aiming to:

  - Maximize the number of **finished orders** after the **Picking Wave** is executed;
  - Process **older orders** faster;
  - Select a set of products that allow for an **efficient route**.

Features

Picking waves

Route calculation

- Based on the generated product set, we calculate the shortest route to visit every collection point, starting at the **Entry Point** to the warehouse and ending at the **Out Point**, where orders are dispatched.

- Takes into account the warehouse shape, exploiting this knowledge during the **shortest path** searching;
- Always returns a possible **route** even if a certain amount of time is reached before the algorithm executes completely;

Features

Picking waves

Warehouse plant

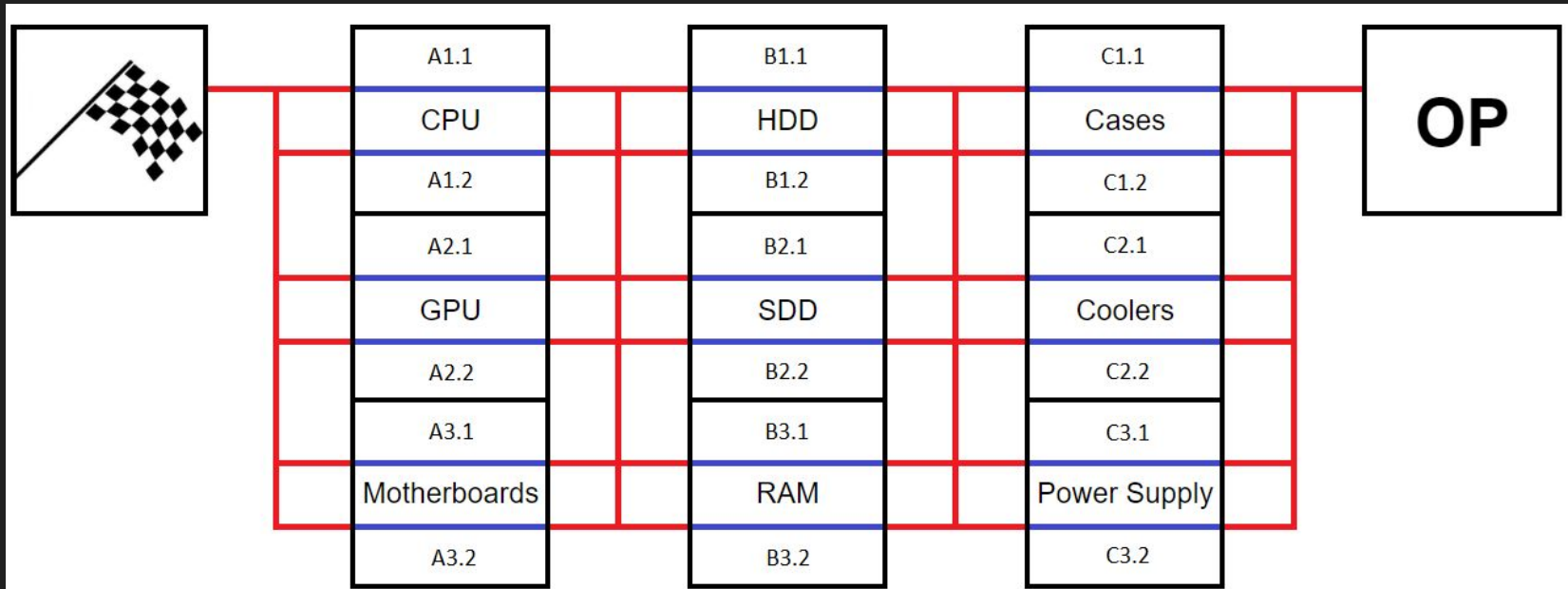

Fig 12 - Warehouse plant

System Architecture & Technologies

Jasmin (cloud management software)

React Native (framework used for building the app)

Firebase (database provider)

Demo

Navigation through the app