Software Requirements Specification Document

Project: Purchasing and Supplier Management System with Inventory Control

Historial de Revisiones

Fecha	Revisión	Descripción	Autor
dd/mm/aaaa	1.0	"Requerimientos de Interfaz"	<Nombre>

Documento validado por las partes en fecha:

Por el cliente	Por la empresa suministradora
Fdo. D./ Dña	Fdo. D./Dña

Contenido

1 Introduction **4**
1.1 Purpose 5
1.2 Project Scope **5**
1.3 Staff involved 5
1.4 Definitions, acronyms and abbreviations 6
1.5 References 6
1.6 Overview 6
2 General Description **6**
2.1 Product Perspective 6
2.2 Functionality of the product 7
2.3 User characteristics 7
2.4 Restrictions 8
2.5 Assumptions and dependencies 8
2.6 Foreseeable evolution of the system 8
3 Specific requirement **9**
3.1 Common interface requirements 10
3.1.1 User Interface 10
3.1.2 Hardware Interfaces 10
3.1.3 Interfaces de software 10
3.1.4 Communication interfaces 11
3.2 Functional requirements 11
3.2.1 Functional requirement: Validity Checking of inputs 11
3.2.2 Functional requirement: Exact Sequence of Operations 11
3.2.3 Functional requirement: Response to Abnormal Situations 12
3.2.4 Functional requirement: Parameters 12
3.2.5 Functional requirement: Generation Outputs 12
3.2.6 Functional requirement: Relationships Between Inputs and Outputs 13
3.2.7 Functional requirement: Specification of Logical Requirements for the Information to be Stored in the Database 13
3.3 Requisitos no funcionales 14
3.3.1 Performance requirements. 14
3.3.2 Restrictions on team management permissions 14
3.3.3 Restrictions on team management permissions 14
3.3.4 Reliability 14
3.3.5 Availability 15
3.3.6 Maintainability 15
3.3.7 Portability 15
4 Attachments **16**
4.1 Diagrams 16
4.1.1 Entity Relation Diagram 16

4.1.2 Relational Model 17
4.1.3 Activity Diagram 18
4.1.4 Component Diagram 18
4.1.6 Deploy Diagram 19
4.1.7 Use cases Diagram 20
4.1.8 Sequences diagram 21
4.2 Preprocess 21
4.3 Content Modeling 21
4.4 Preconstruction 21
4.4.1 Mockup 21
4.4.2 Wireframes 21
4.5 Construction 23

1 Introduction

This program focuses on optimizing the process within the purchasing area of a certain company, making use of various technologies and tools, such as MySql database, PHP, JavaScript, among others, for the development of a web page with which you can manage and perform various actions within the purchasing area.

1.1 Purpose

The purpose of this document is to be able to provide and correctly report all kinds of information that may seem inconclusive about the software specifications.

Target audience:

Companies seeking to optimize the management from suppliers.

1.2 Project Scope

The "Purchasing and Inventory Management System for Industry" will streamline the management of supplier purchases and inventory control. Key features include:

1. **Purchasing Management:** Handle supplier orders, approve or reject requests, and report deliveries to HR.
2. **Inventory Control:** Supervisors track shortages and report to HR for material requests.
3. **Receiving Management:** Verify deliveries and report to Purchasing.

The system aims to improve efficiency, communication, and supply chain management within the organization.

1.3 Staff involved

Name	Martinez Bustamante Daniel
Rol	Programmer
Professional Category	Student
Responsibilities	Operative Systems
Contact information	0323105919@ut-tijuana.edu.mx
Approval	

Name	Torrez Iñiguez Ariel
Rol	Programmer and documentor
Professional Category	Student
Responsibilities	Database
Contact information	0323105868@ut-tijuana.edu.mx
Approval	

Name	Garcia Manga Jared
Rol	Documentor and Sales
Professional Category	Student

Responsibilities	Integrator
Contact information	0323105868@ut-tijuana.edu.mx
Approval	

Name	Lara Lopez Josue Isaac
Rol	Programmer
Professional Category	Student
Responsibilities	Web App
Contact Information	0323105937@ut-tijuana.edu.mx
Approval	

1.4 Definitions, acronyms and abbreviations

1.5 References

Reference	Title	Route	Date	Autor

1.6 Overview

This project aims to develop a system to manage industrial purchases and their reception through a web application. The application will use technologies such as HTML, CSS, JavaScript and PHP, integrating a MySQL database. It is designed to run on the Ubuntu operating system, optimizing efficiency in the process of purchasing and receiving goods. The document is organized using IEEE830 software requirements specifications (ERS).

2 General Description

With our project we propose a comprehensive management system that facilitates the relationship with suppliers, improving the purchasing and inventory control.
The system optimizes the request and receipt of materials, ensuring efficiency and quality, as may become the main factors that influence the development of this solution, such as key functions, user characteristics, limitations and dependencies.

2.1 Product Perspective

The application is intended for the purchasing area within the industry, where strict control of raw material purchase order is required.

2.2 Functionality of the product

- User management: user registration, login, profile creation.
- Creation of purchase requisitions.
- Approve or reject purchase requisitions.
- Placing orders to suppliers.
- Verify delivery of products.

2.3 User characteristics

User Type	Supervisor
Education	Technical or university education, in production or operations management.
Skills	Ability to generate reports and initiate material requests in the system.
Activities	Sends material reports to the administration.

User Type	Human Resources
Education	University education in business administration or logistics.
Skills	Competence in report verification and managing internal communications.
Activities	Reviews reports, creates material requests, and sends them to the purchasing department.

User Type	Purchasing
Education	University degree in supply chain management or logistics.
Skills	Proficiency in supplier management, order creation, and tracking.
Activities	Views and approves requests, sends orders to suppliers, and tracks deliveries.

User Type	Store
Education	Technical or secondary education, often with specific training in inventory management.
Skills	Skills in verifying deliveries, updating inventory, and confirming receipt of orders.
Activities	Verifies the delivery of goods and sends confirmation back to the purchasing department.

2.4 Restrictions

- Database Connection in MySQL
- Connecting to Apache
- Developed for Ubuntu
- Use HTML, CSS, JavaScript and PHP
- Exclusive use for employees registered in the database

2.5 Assumptions and dependencies

Assumptions and dependencies are key factors that affect the success of the system. These describe the conditions necessary for its operation, such as the availability of resources, instigation with other systems.

Assumptions:

- **Accessibility**: It is assumed that all users will have access to devices with internet connectivity (PCs, tablets or Smartphones) to interact with the system
- **Supplier Collaboration:** Suppliers are expected to adopt the system smoothly and be willing to participate in evaluations and order status updates.
- **Seamless interaction:** It is assumed that the integration with other company systems will be fluid and that there will be no major interoperability problems between the inventory, accounting and finance systems**.**

Dependencies:

- **Data dependency**: The system will need accurate and up-to-date data on products, inventories, and suppliers to function properly.
- **Integration with external systems:** The success of the system will depend on its ability to integrate with other industrial and financial management systems that are already in use within the company.
- **Technical support:** The correct operation of the system will depend on the technical support team in charge of performing updates, maintenance and problem resolution.

2.6 Foreseeable evolution of the system

In the future, we want to implement extensions to cover many more areas of the company and not only focus on certain specific areas, but in the areas that we touch briefly to be able to deepen them more and thus have much more area of why we are requesting or / and requesting certain things such as purchase requests, cover more specific requests.

3 Specific requirement

Requirement number	REQ-001
Requirement name	Log in
Type	Required Restriction
Requirement font	Requirements requested by the client
Requirement priority	Alta/Esencial Media/Deseado Baja/ Opcional

Requirement number	REQ-002
Requirement name	Automatic user creation
Type	Required Restriction
Requirement font	Requirements requested by the client
Requirement priority	Alta/Esencial Media/Deseado Baja/ Opcional

Requirement number	REQ-003
Requirement name	Have different work spaces
Type	Required Restriction
Requirement font	Requirements requested by the client
Requirement priority	Alta/Esencial Media/Deseado Baja/ Opcional

3.1 Common interface requirements

3.1.1 User Interface

The user interface will consist of a web application that after login will send you to different work pages depending on your position where it will allow you to carry out different activities such as sending documentation.

3.1.2 Hardware Interfaces

The system shall interact with various hardware components to provide a seamless experience for users. Hardware interfaces include input devices, output devices, storage, and network connectivity. The logical characteristics for each of these interfaces, including the necessary configurations, are detailed below.

Mouse and keyboard

- **Protocol:** USB 2.0/3.0, Bluetooth 4.0 or higher.
- **Configuration:** Must allow input of textual data and navigation **commands.** Plug-and-play configuration.
- **Compatibility:** Compatible with Linux (Ubuntu) operating system

Minimum processor requirements

- **Intel**: Intel Pentium 4 or superior
- **AMD**: AMD Athlon 64 or superior

Additional requirements

- **Software compatibility**: The processor must be compatible with Linux distributions such as Ubuntu 18.04 or higher.
- **Adequate performance**: Although these processors are older and basic, they are capable of running office applications and small programs without significant problems.

3.1.3 Interfaces de software

Relational Database for Data Storage

- **Product Description Software Used**
  - **Product: MySQL**
  - **Purpose of the Interface:** Storage and management of program data, including users, stock, suppliers, etc.

Web application

Product Description Software Used

Product: Web Application in HTML, CSS, JavaScript and PHP
Purpose of the Interface: Allow the application to interact with the MySQL database and other necessary services.

3.1.4 Communication interfaces

The servers, users and computers will be able to communicate with each other through standard Internet protocols, in order to communicate between the database and the computing equipment.

3.2 Functional requirements

- Validity checking inputs.
- Exact sequence of operations.
- Response to abnormal situations (overflows, communications, error recovery).
- Parameters.
- Generation of outputs.
- Relationships between inputs and outputs (input and output sequences, formulas for information conversion).
- Specification of logical requirements for the information to be stored in the database (type of information required).
- Functional requirements can be divided into subsections

3.2.1 Functional requirement: Validity Checking of inputs

This ensures that all data entered into the system meets strict accuracy and completeness standards, which prevents errors that can propagate across modules, leading to inventory mismanagement, purchasing issues, or supplier miscommunication. By validating fields such as order quantities, product codes, supplier contact details, and other essential data, the system minimizes the risk of errors and discrepancies, enhancing the reliability of every action performed within the system. Input validation includes format checking, mandatory field requirements, and real-time feedback to users for correcting potential issues before submission

3.2.2 Functional requirement: Exact Sequence of Operations

Establishing a precise sequence of operations is crucial for ensuring that processes within the system are executed in the correct order. This sequence impacts workflows like creating and verifying purchase orders, updating

inventory records, and coordinating with suppliers. For instance, the system could require that inventory levels are checked and confirmed before a new purchase order is placed. This sequencing prevents errors by enforcing logical steps in each process and allowing seamless integration between modules. By following an exact operational order, data integrity is maintained, and staff efficiency is optimized as tasks are completed systematically.

3.2.3 Functional requirement: Response to Abnormal Situations

The system is designed to handle unexpected situations gracefully, such as data overflows, network communication disruptions, or input errors. Automated error detection and response mechanisms mitigate risks associated with these issues, minimizing data loss and maintaining operational continuity. For instance, if there is a network failure during a transaction, the system might save the process state to allow smooth recovery once connectivity is restored. In cases of input overflow or invalid data, the system can alert users immediately and suggest corrective actions, which promotes resilience and reliability within daily operations.

3.2.4 Functional requirement: Parameters

Parameters are predefined values or thresholds that control system behavior, which can be adjusted according to operational needs. This includes defining minimum stock levels to trigger reorder alerts, setting purchase order limits for budget management, and specifying lead times for supplier deliveries. By establishing and refining these parameters, the system can automate tasks like notifying users of low stock levels or flagging orders that exceed spending limits. Parameters allow for customization and scalability, enabling the system to adapt to changing requirements or business growth.

3.2.5 Functional requirement: Generation Outputs

The system provides various outputs, including purchase order confirmations, supplier performance reports, inventory summaries, and alerts for low-stock or overdue deliveries. These outputs are formatted and generated based on real-time data from the database, making them valuable for decision-making. For instance, regular supplier performance reports can help assess reliability, while inventory summaries allow quick visibility into stock levels. Each output is tailored to provide actionable insights, facilitating more informed decisions by managers and staff and supporting continuous improvement in purchasing and inventory processes.

3.2.6 Functional requirement: Relationships Between Inputs and Outputs

Inputs and outputs within the system are interconnected through structured formulas and logical relationships. For instance, a low-stock input might trigger a reorder alert, or a purchase order input may link directly to inventory updates upon delivery. Formulas for information conversion also assist in interpreting data; for example, calculating order fulfillment rates based on purchase and delivery records. This design ensures that each input has a meaningful and logical impact on system outputs, promoting cohesive data flow and reducing errors in tracking procurement and inventory metrics.

3.2.7 Functional requirement: Specification of Logical Requirements for the Information to be Stored in the Database

The database is structured according to the types of information it will manage, such as supplier profiles, inventory details, order histories, and user activities. Each piece of information stored meets specific logical criteria, allowing quick retrieval, reliable analysis, and secure management of data. For example, supplier data includes contacts, pricing, and delivery history, while inventory records track quantities, locations, and reorder points. By clearly specifying these requirements, the database design supports the complex relational queries necessary for efficient data handling and operational reporting across modules.

3.3 Not functional requirements

3.3.1 Performance requirements.

- **Availability.** The system is expected to be available 98% of the time.
- **Response Time.** A response time of maximum 3 seconds is expected 94% of the time.
- **Processing.** The system should be able to handle several requests at the same time.
- **Resource Usage**. The CPU usage of the server should not be excessive, greater than 80% in most cases.
- **Network latency.** Critical operations, such as order approval and inventory updates, should have a maximum latency of 100 milliseconds.
- **Initial load times.** The system should be available for use within 5 seconds of system startup.
- **Memory usage.** The system should not exceed 65% of the amount of memory allocated to the system in normal situations.

3.3.2 Restrictions on team management permissions

Specification of elements that will protect the software from malicious access, use and sabotage, as well as from malicious WDor accidental modifications or destruction. The requirements may specify:

- **Security measures implemented in the code**
- **Registering and declaring variables privately**
- **Assignment of certain functionalities to certain modules.**

3.3.3 Restrictions on team management permissions

- **Authentication. The system must use authentication through secure credentials, the user and his password.**
- **Role-based access control. The system must implement access control by levels, based on user roles.**
- **Audit logging. The system should record all relevant activities, including order changes, inventory updates, and access to sensitive data. Logs should be retained for a minimum of 12 months.**
- **Account lockout policy. The system must temporarily lock an account after 10 failed login attempts within a 10-minute period, with a lockout time of at least 15 minutes.**
- **Data backup and recovery: The system must perform automatic backups of all critical data, including inventory, orders and suppliers, at least once a day.**

3.3.4 Reliability

- **Fault tolerance.** The system must be able to continue operating without interruption in the event of minor hardware or software failures, through the implementation of automatic recovery mechanisms or redundancy in critical services.
- **Availability.** The system must be available at least 99% of the time during working hours and 98% of the time outside working hours.
- **Data consistency.** The system must ensure 100% data consistency, especially in high-priority operations such as order entry, inventory updates, and purchase approvals.
- **Stress testing.** The system must be able to pass stress tests with 50% more than the maximum expected load, without significant performance degradations or service drops.

3.3.5 Availability

- **System availability.** The system must be available at least 99% of the time during working hours and at least 98% of the time outside working hours.

- **Recovery from critical failures.** The system must be able to recover from critical failures in no more than 30 minutes, ensuring that data is restored to the last backup without significant loss of information.

- **System monitoring.** Constant monitoring of the system should be implemented to detect and fix availability problems before they affect users. Notifications should be sent to administrators in case of outages or failures.

3.3.6 Maintainability

- **Identification of the type of system maintenance required.** The system will require preventive and corrective maintenance, focused on security updates and improvements to the purchasing and supplier database.

- **Specification of who should perform the maintenance tasks**. Maintenance tasks will be performed by the company's internal IT team or by an external developer with restricted access to the system.

- **Specification of when maintenance tasks should be performed.** Maintenance shall be performed on a scheduled basis once a month, outside working hours (weekends), and should include performance reporting and system auditing.

3.3.7 Portability

- **Percentage of server-dependent components.** 85% of the system will be centralized in the company's main server, and 15% will depend on external web services for validations and authentications.

- **Percentage of server-dependent code.** Approximately 90% of the code will be designed to run on the company's local server, with 10% running on distributed systems or cloud services.

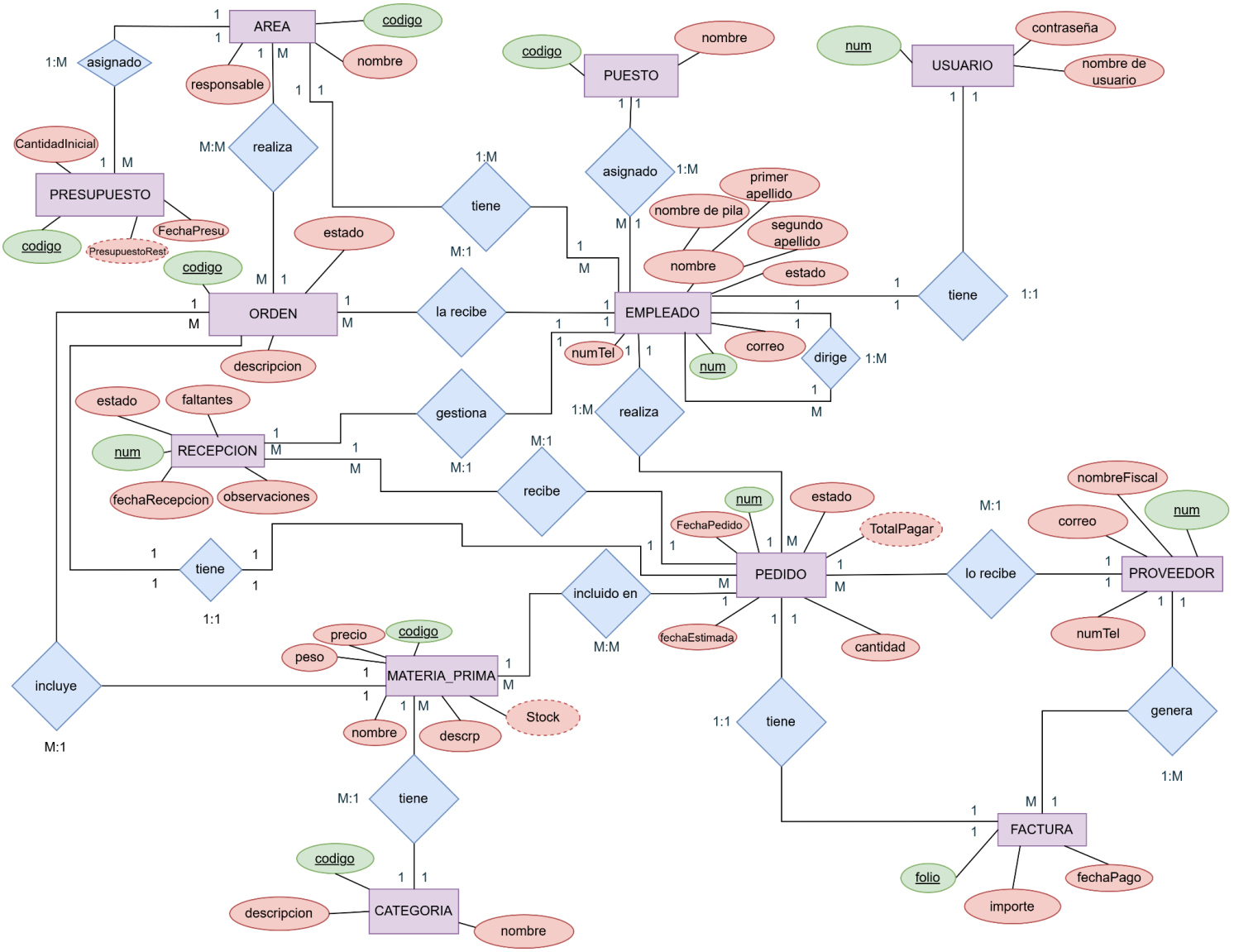
- **Use of a specific compiler or development platform.** The system will be developed using VS Code to ensure compatibility with servers and production environments.

- **The Ubuntu Linux distribution will be used for the creation of the software.** The development and production environment will be based on Ubuntu 22.04 LTS servers, ensuring stability and security for the application.

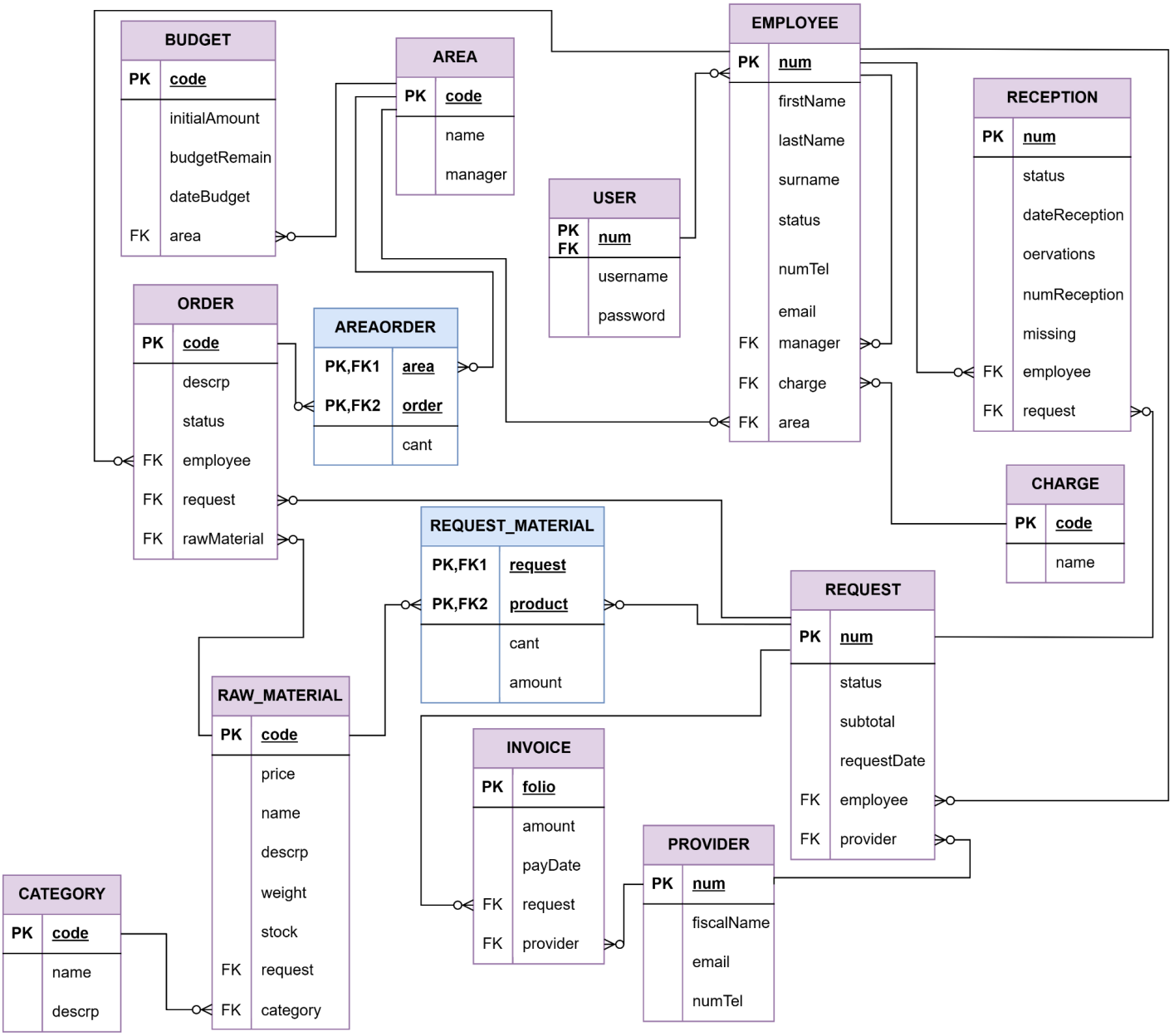
4 Attachments

4.1 Diagrams

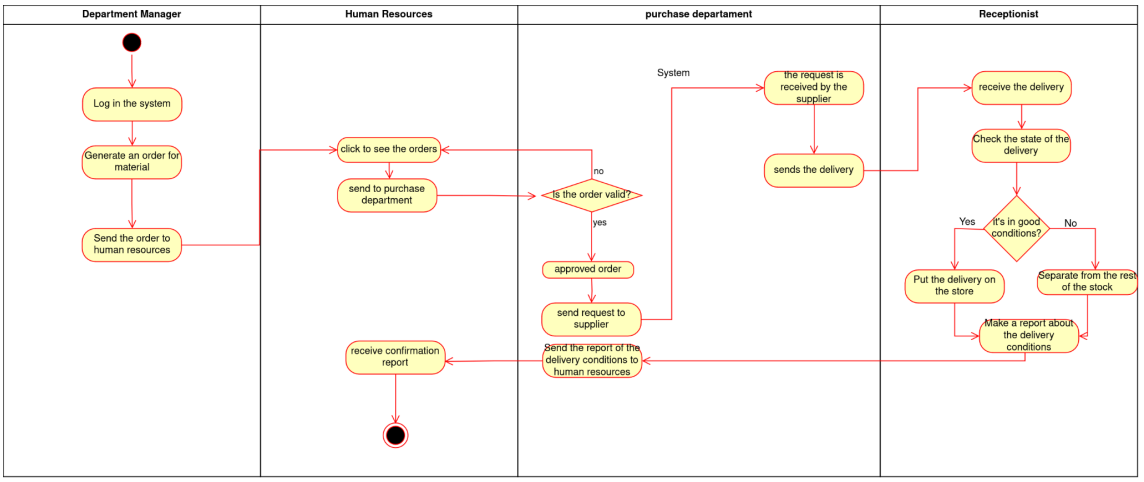
4.1.1 Entity Relation Diagram



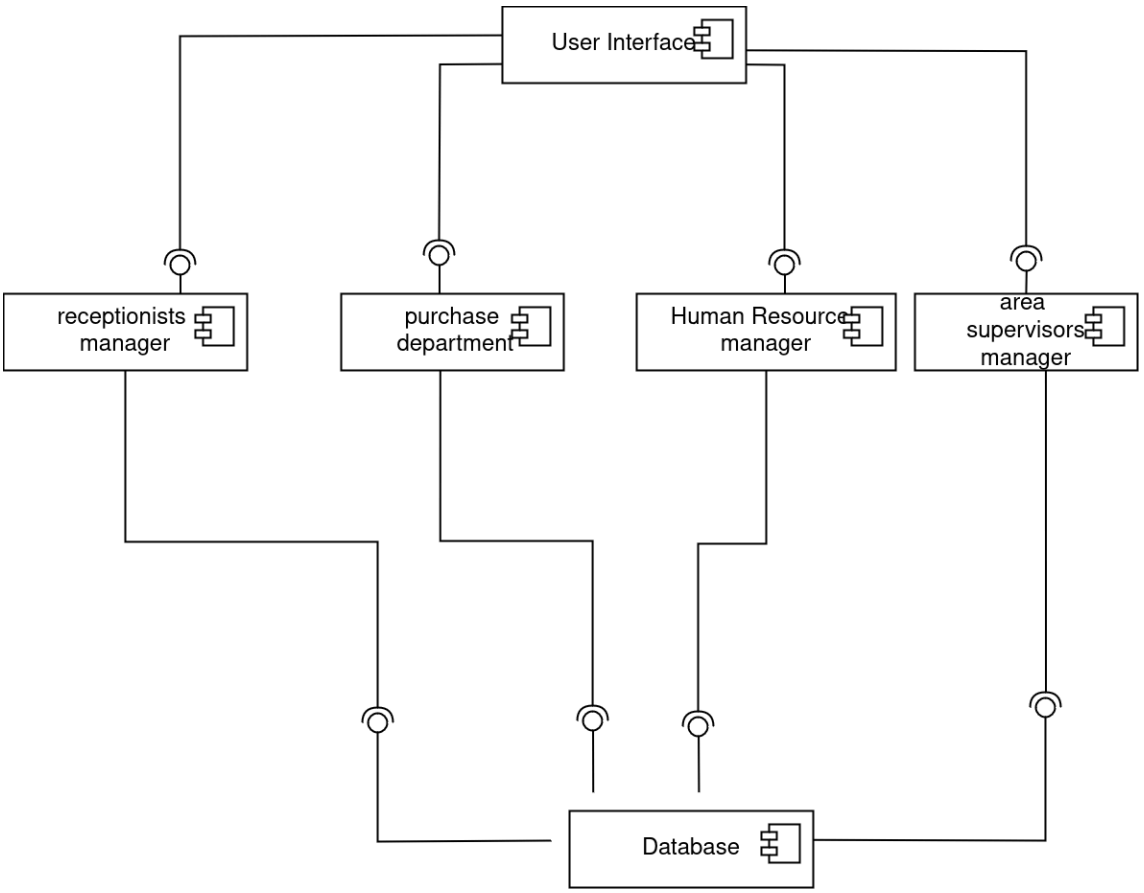
4.1.2 Relational Model



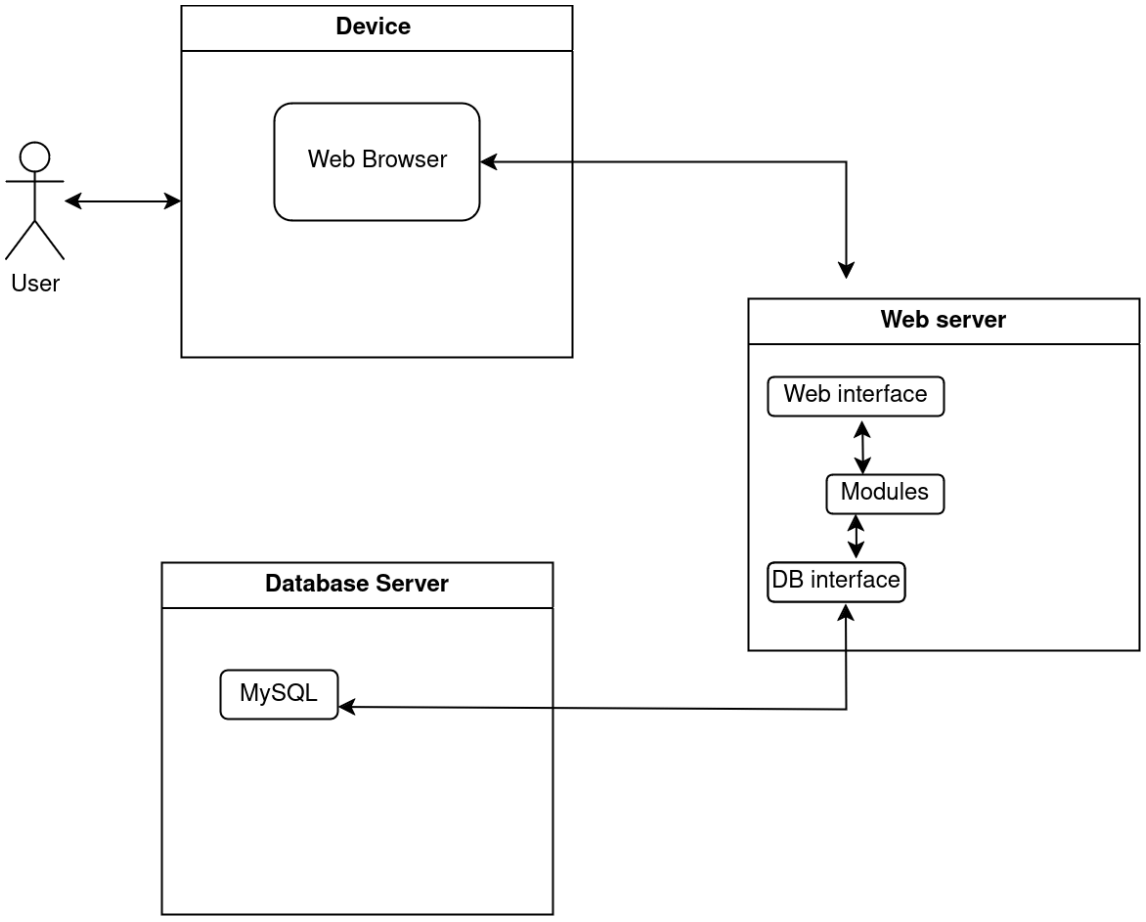
4.1.3 Activity Diagram



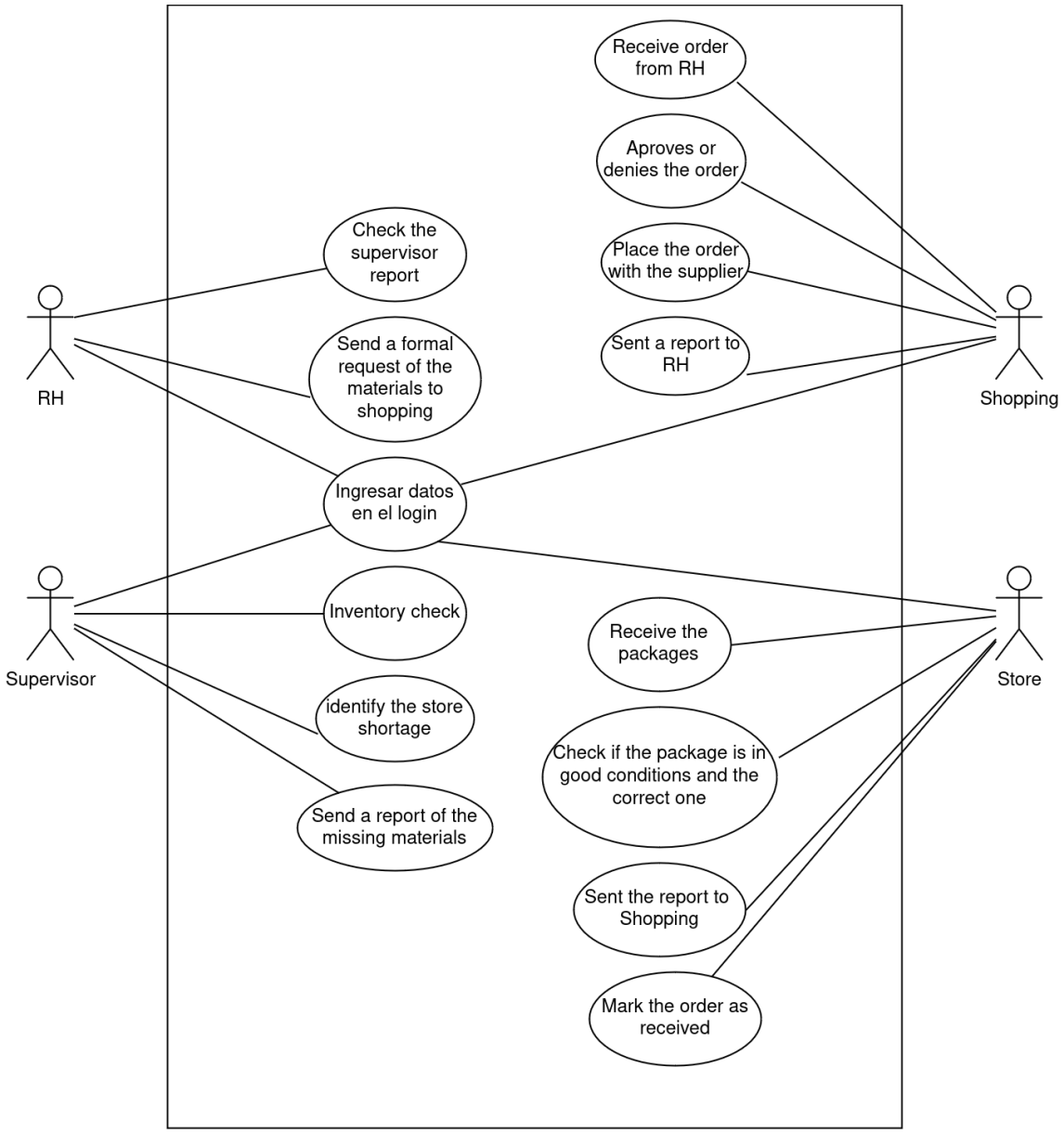
4.1.4 Component Diagram



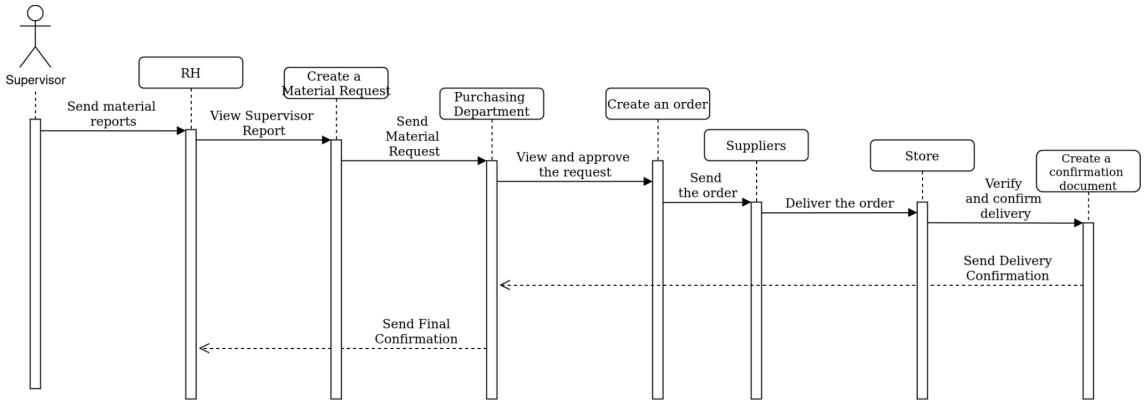
4.1.6 Deploy Diagram



4.1.7 Use cases Diagram



4.1.8 Sequences diagram

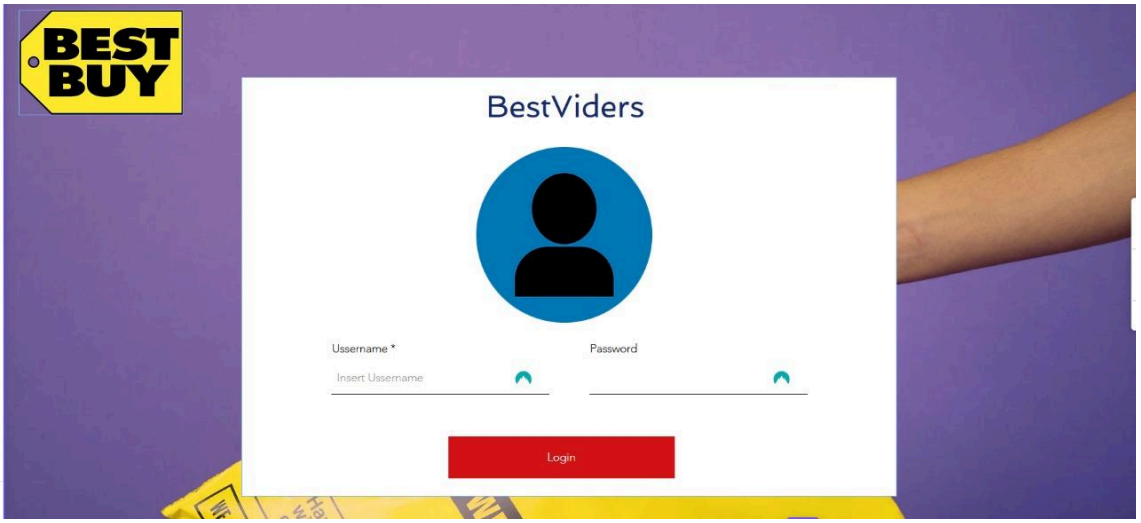


4.2 Preprocess

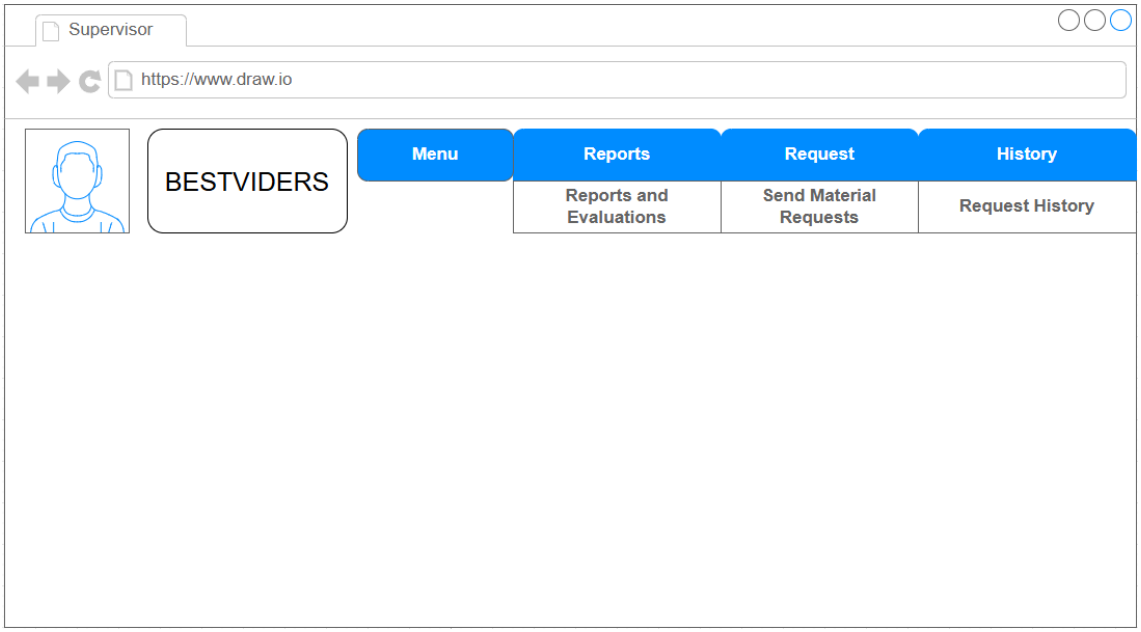
4.3 Content Modeling

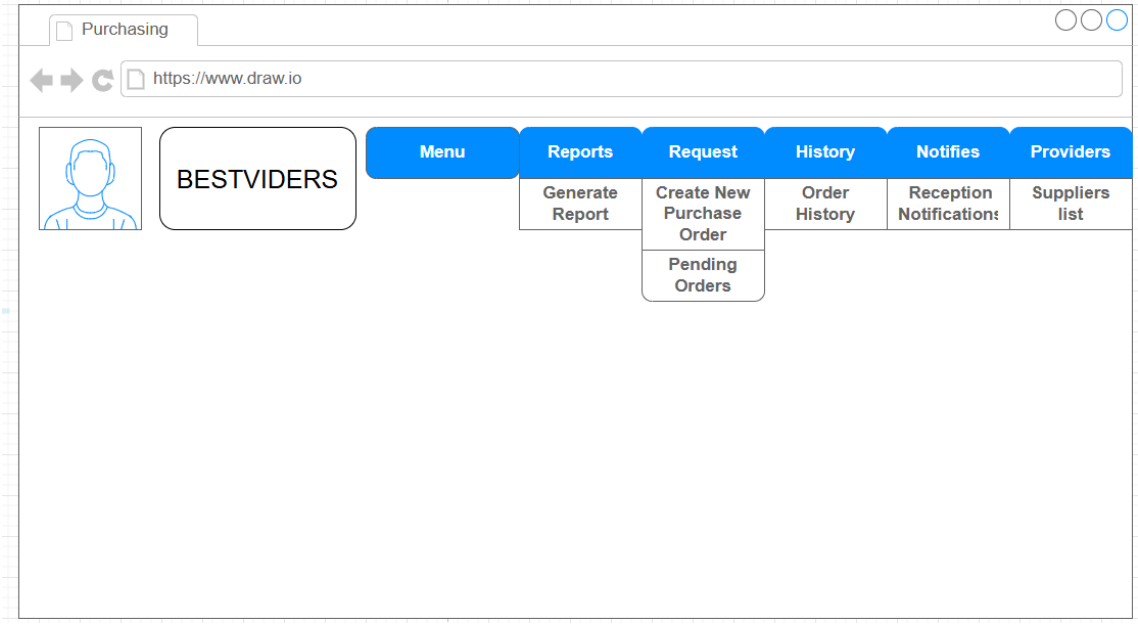
4.4 Preconstruction

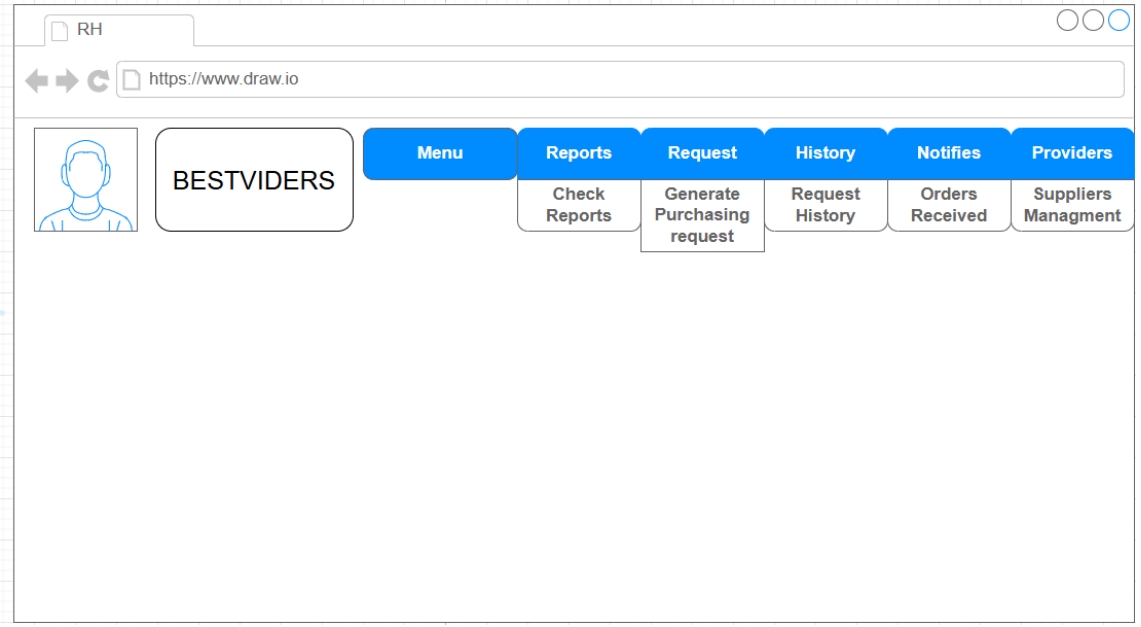
4.4.1 Mockup

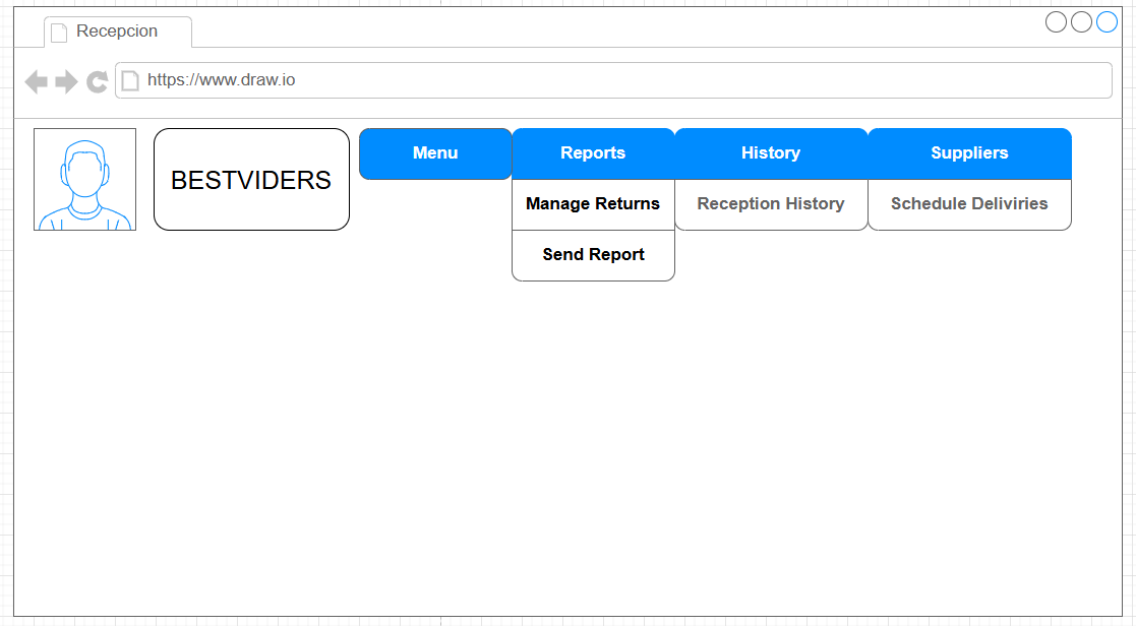


4.4.2 Wireframes


Supervisor
https://www.draw.io
BESTVIDERS
Menu
Reports
Request
History
Reports and Evaluations
Send Material Requests
Request History

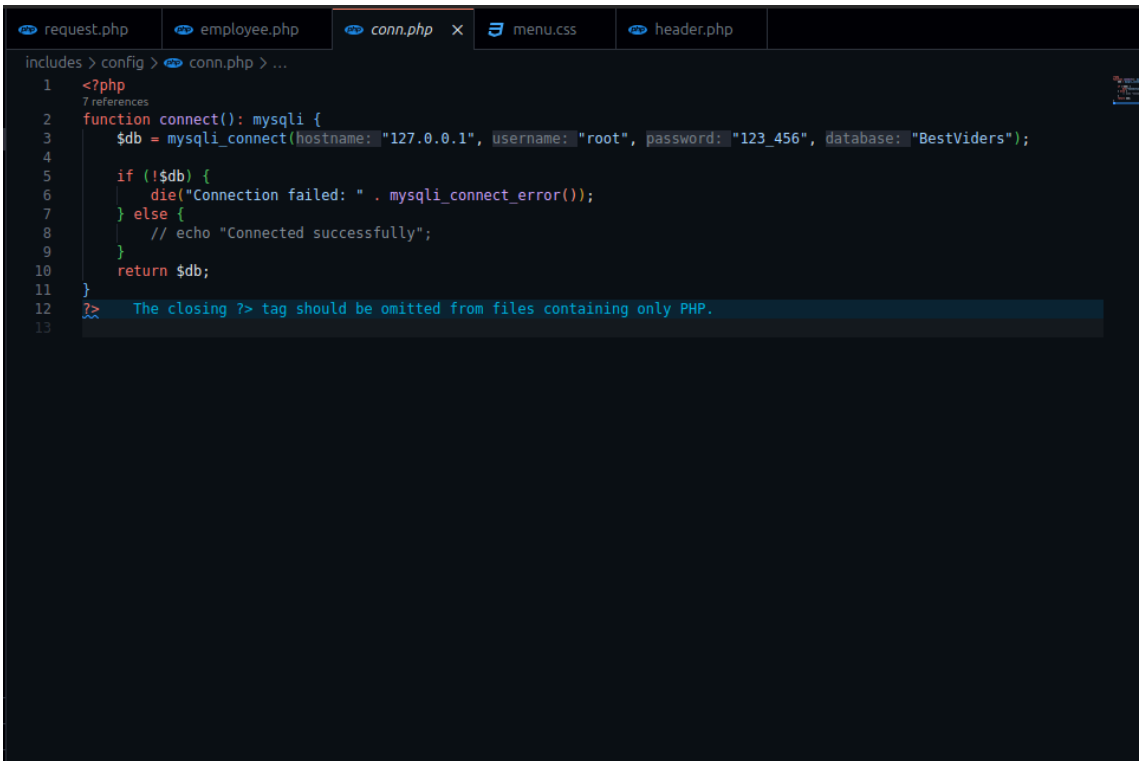

Purchasing
https://www.draw.io
BESTVIDERS
Menu
Reports
Request
History
Notifies
Providers
Generate Report
Create New Purchase Order
Pending Orders
Order History
Reception Notifications
Suppliers list


RH
https://www.draw.io
BESTVIDERS
Menu
Reports
Request
History
Notifies
Providers
Check Reports
Generate Purchasing request
Request History
Orders Received
Suppliers Managment


Recepcion
https://www.draw.io
BESTVIDERS
Menu
Reports
History
Suppliers
Manage Returns
Reception History
Schedule Deliviries
Send Report

4.5 Construction

FILE CONNECTION.



This file contains a PHP script that defines a function called connect() in order to establish a connection to the DB using the mysqli extension.

After attempting to connect, the function evaluates whether the connection was successful. In case it fails, the die() function is executed, which terminates the script and displays an error message, which includes the specific error provided by mysqli_connect_error() to facilitate diagnosis. If the connection is successful, the script is designed to simply return the $db connection object without displaying additional messages.

LOGIN

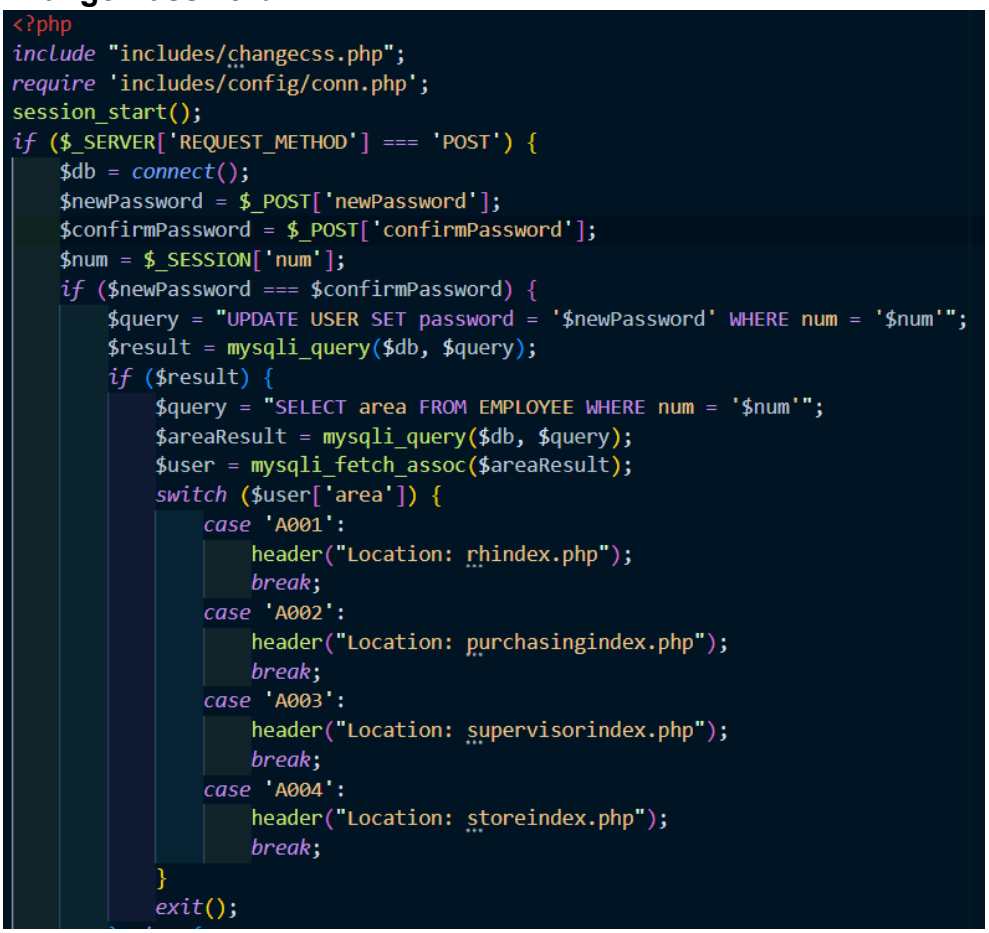
```php
<?php
include "includes/loginh.php";
require 'includes/config/conn.php';
session_start();
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $db = connect();
    $query = "
        SELECT E.num, E.firstName, E.lastName, E.area, U.password
        FROM EMPLOYEE AS E
        INNER JOIN USER AS U ON E.num = U.num
        WHERE E.email = '$email' AND U.password = '$password'
    ";
    $result = mysqli_query($db, $query);
```

The PHP code makes a query to EMPLOYEE AND USER, after selecting the necessary columns to identify the user and filtering the records according to the email and password provided, if the data matches, continue with the code.

```php
    $result = mysqli_query($db, $query);
    if (mysqli_num_rows($result) === 1) {
        $user = mysqli_fetch_assoc($result);
        $_SESSION['user_name'] = $user['firstName'] . ' ' . $user['lastName'];
        $_SESSION['num'] = $user['num'];
        if ($password === "1234567890") {
            header("Location: changepassword.php");
            exit();
        }
        switch ($user['area']) {
            case 'A001':
                header("Location: rhindex.php");
                break;
            case 'A002':
                header("Location: purchasingindex.php");
                break;
            case 'A003':
                header("Location: supervisorindex.php");
                break;
            case 'A004':
                header("Location: storeindex.php");
                break;
        }
        exit();
    } else {
        $error = "Incorrect email or password.";
    }
    mysqli_close($db);
}
```

Continuing with the code, it checks that the credentials are validated and recovers the data by storing it in $user. Then if your password is equal to "1234567890" it will send you to a page to change your password, "1234567890" is the default password when your user is created so the first time you enter it will send you to change your password. If your password is not the default one, the code will send you to your next page, which would be the index depending on which area you work in.

Change Password



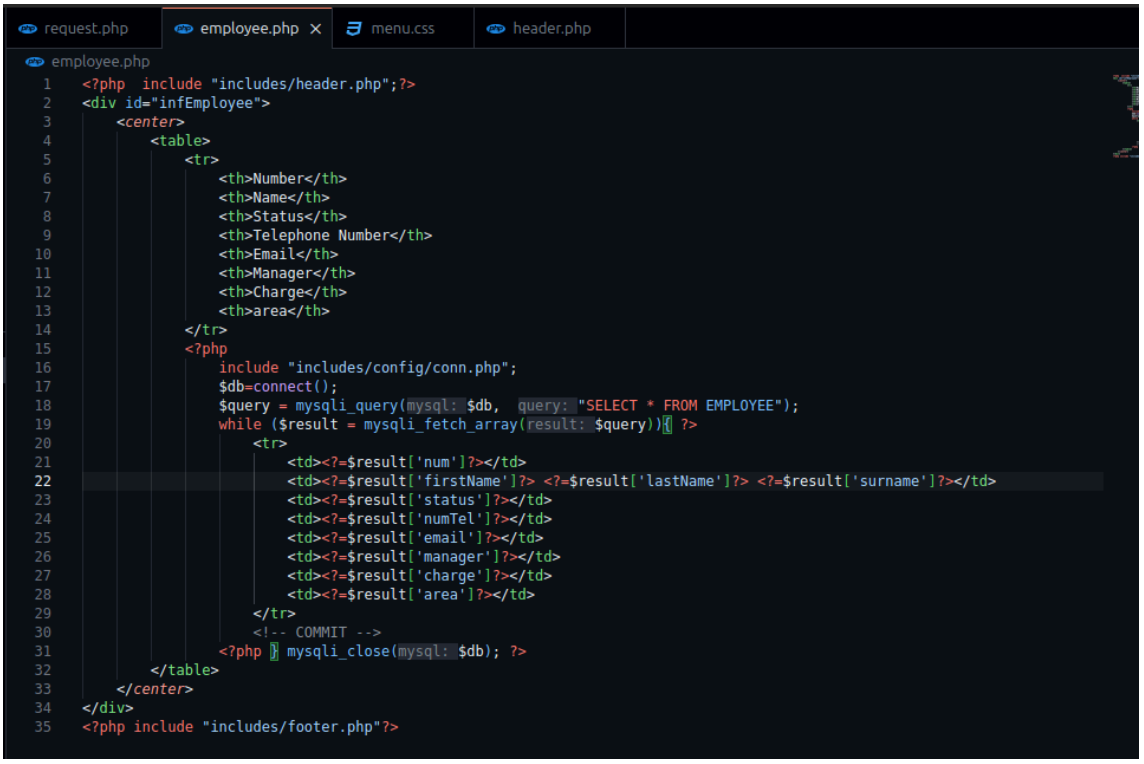
```php
<?php
include "includes/changecss.php";
require 'includes/config/conn.php';
session_start();
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $db = connect();
    $newPassword = $_POST['newPassword'];
    $confirmPassword = $_POST['confirmPassword'];
    $num = $_SESSION['num'];
    if ($newPassword === $confirmPassword) {
        $query = "UPDATE USER SET password = '$newPassword' WHERE num = '$num'";
        $result = mysqli_query($db, $query);
        if ($result) {
            $query = "SELECT area FROM EMPLOYEE WHERE num = '$num'";
            $areaResult = mysqli_query($db, $query);
            $user = mysqli_fetch_assoc($areaResult);
            switch ($user['area']) {
                case 'A001':
                    header("Location: rhindex.php");
                    break;
                case 'A002':
                    header("Location: purchasingindex.php");
                    break;
                case 'A003':
                    header("Location: supervisorindex.php");
                    break;
                case 'A004':
                    header("Location: storeindex.php");
                    break;
            }
            exit();
```

The PHP code makes a query to EMPLOYEE AND USER, after selecting the necessary columns to identify the user and filtering the records according to the email and password provided, if the data matches, continue with the code.This code is a continuation of the previous one and is executed when the password entered is the default one. First, it retrieves the user's unique identifier (num) to ensure that the correct password is updated. Then, check if the new password matches the confirmation entered. If they do not match, it will display an error message; Otherwise, it will proceed to update the password. Once the password update is successful, the code queries the EMPLOYEE table to obtain the user's area and redirect them to their corresponding workplace. On future logins, the user will not have to go through this update process.

TABLE PROVIDERS

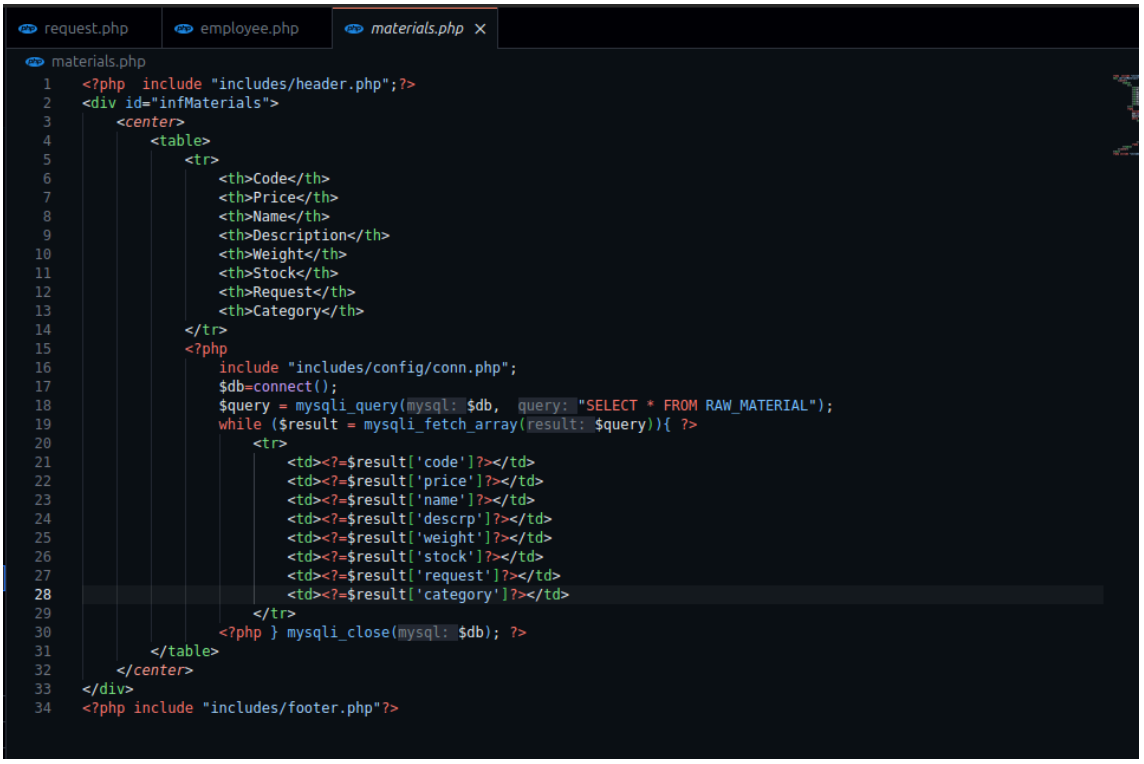
This php and html code makes a table by means of a SELECT query to the PROVIDER table which extracts all the records from it. Subsequently these are placed into an html table by means of a while loop that iterates the array in which the result of the query was stored.
The result is a table in which all supplier information can be displayed.

TABLE EMPLOYEES.



This code performs a query to the EMPLOYEE table, with which all the information of the table is obtained, after that it is stored in an array. Then by means of a while it iterates on the array and the fields stored in this one are accommodated inside its corresponding cell in the table, at the end of the iteration the cycle is closed. As a result we have a table which shows all the information of the employees of the purchasing area of the company.

TABLE MATERIALS



This code performs a query to the RAW_MATERIAL table from which it extracts all the information and in a similar way to the other two tables. It stores the result in an array which is then traversed with a while to display the information in a table to show the data of each material.

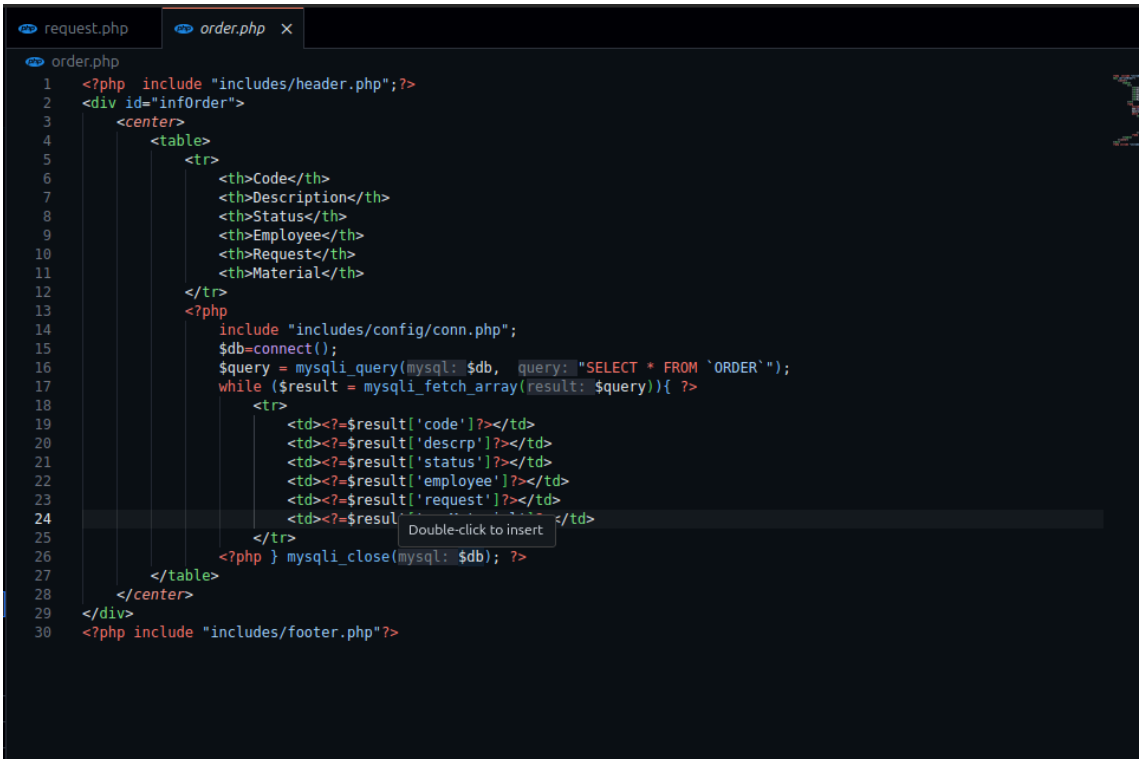
TABLE REQUEST.

request.php

```php
<?php  include "includes/header.php";?>
<div id="infRequest">
    <center>
        <table>
            <tr>
                <th>Number</th>
                <th>Status</th>
                <th>Subtotal</th>
                <th>Date</th>
                <th>Check by Employee</th>
                <th>Provider</th>
            </tr>
            <?php
                include "includes/config/conn.php";
                $db=connect();
                $query = mysqli_query(mysql: $db,  query: "SELECT * FROM REQUEST");
                while ($result = mysqli_fetch_array(result: $query)){ ?>
                    <tr>
                        <td><?=$result['num']?></td>
                        <td><?=$result['status']?></td>
                        <td><?=$result['subtotal']?></td>
                        <td><?=$result['requestDate']?></td>
                        <td><?=$result['employee']?></td>
                        <td><?=$result['provider']?></td>
                    </tr>
                <?php } mysqli_close(mysql: $db); ?>
        </table>
    </center>
</div>
<?php include "includes/footer.php"?>
```

This code performs a query to the REQUEST table from which it extracts all the information and in a similar way to the other two tables. It stores the result in an array which is then traversed with a while to display the information in a table to show the data of each REQUEST.

TABLE ORDER.



This code performs a query to the ORDER table from which it extracts all the information and in a similar way to the other two tables. It stores the result in an array which is then traversed with a while to display the information in a table to show the data of each ORDER.

ADD PROVIDERS.

```php
<?php
include "includes/config/conn.php";

$conn = conn(); // Usar la función para obtener la conexión

$result = $conn->query("SELECT MAX(num) AS last_id FROM provider");
$row = $result->fetch_assoc();
$last_id = $row['last_id'] + 1;
?>
<link rel="stylesheet" href="includes/css/forms.css">
<nav id="Return"><a href="index.php">Regresar</a></nav>
<h2>Add Provider</h2>
<sectio id="formCont">
    <div id=formCard>
        <form id ="form" method="POST" action="providerProcess.php">
            <LEGENd>Fill all fields</LEGENd>
            <label for="num">Provider Number</label>
            <input type="number" name="num" id="num" required readonly value="<?php echo $last_id; ?>">

            <label for="fiscalName">Fiscal name</label>
            <input type="text" name="fiscalName" id="fiscalName" required >

            <label for="numTel">Phone number</label>
            <input type="text" name="numTel" id="numTel" required>

            <label for="email">Email</label>
            <input type="email" name="email" id="email" required >
            <div id="btn">
                <button type="submit" class="button">ADD</button>
            </div>
        </form>
    </div>
</section>
```

This html and php are for make the form that the user will use to add a new provider to the database, the fragment of a php that is used for connect to the database then the next part of the $result make the query that will obtain the biggest number of the providers then it will assign the alias: “last_id”, then it fetches as an array and the last line takes the alias and increments the value by 1 so it can be printed on the input and the user can see the actual number of provider

```php
<?php
// Conexión a la base de datos
include "includes/config/conn.php";

$conn = conn();

$num = $_POST['num'];
$fiscalName = $_POST['fiscalName'];
$email = $_POST['email'];
$numTel = $_POST['numTel'];


$insert = "INSERT INTO provider (num, fiscalName, email, numTel) VALUES ('$num','$fiscalName','$email','$numTel')";

if ($conn->query($insert) === TRUE) {
    echo "Nuevo registro creado exitosamente";
} else {
    echo "Error: " . $insert . "<br>" . $conn->error;
}

// Cerrar la conexión
$conn->close();
?>
```

It starts with a database connection so the next lines can be executed properly. Then the data from the POST is retrieved and the values are stored on the defined variables. The information it’s added to the insert query and its added to the database and the final if its to check if the query is successful if it is it shows you a message of succes else an error message.
Finally close the database.