

Maintenance Management System for Industrial Equipment

Specification of SRS Software Requirements

Summary.

Specification of requirements for maintenance
management system for industrial machinery

Authors

Arellano Jocelyn, Bravo Selegna, Zavala Jesús

Professor

Ray Brunnet Parra Galaviz

Technological University of Tijuana

Tijuana Baja California, México

Monday, September 22, 2024

Document Information Sheet

Date	Revision	Autor	Quality verification
September 22, 2024		Arellano Jocelyn Bravo Selegna Zavala Armando	

By the customer	By the company

Content

Document Information Sheet	1
1. Introduction.....	5
1.1 Purpose of the Document.....	5
1. 2. Scope of the System.....	5
1.3 Involved Personnel.....	5
1. 4. Definitions, acronyms and abbreviations	6
1. 5. Summary	6
1. 6. References	7
2. General Description	9
2.1 Product Perspective	9
2.2 Product Functionality	9
2.2.1 Incident Management.....	9
2.2.2 Automatic Technician Assignment.....	10
2.2.3 Preventive Maintenance Scheduling	10
2.2.4 Repair Management	10
2.2.5 Material Request and Approval	11
2.2.6 Machine Functionality Indicator	11
2.2.7 Reports and Data Analysis.....	11
2.3 User Characteristics.....	12
2.4 Constraints	12
2.5 Assumptions and Dependencies.....	13
2.6 Predictable Evolution of the System.....	14
3. Requirements	15

3.1 Functional Requirements.	15
3.2 No functional Requirements	21
4. Appendix.....	24
4.1 Entity Relationship Diagram.....	24
4.2 Use case diagram	25
4.3.1 Operator	26
4.3.2 Technicians	27
4.3 Sequence diagrams.....	28
4.4.1 Operator	28
4.4.2 Manager	28
4.4.3 Technician.....	29
4.5 Status diagrams	30
4.5.1 Operator	30
4.5.2 Manager	31
4.5.3 Technician.....	32
4.6 Activity diagrams.....	33
4.6.1 Operator	33
4.6.2 Manager	34
4.6.3 Technician.....	35
4.7 Class diagrams	36
4.7.1 Operator	36
4.7.2 Manager	37
4.7.3 Technician.....	37

1. Introduction

1.1 Purpose of the Document

This SRS aims to clearly define the functional and non-functional requirements of the "Maintenance Management System for Industrial Equipment (MEI)." This system aims to improve industrial maintenance management by minimizing downtime and optimizing resources through a centralized system for managing preventive and corrective maintenance, incidents, and technician assignments.

1.2. Scope of the System

The MEI system will be available as a web platform, allowing users to track industrial equipment maintenance comprehensively. The system will include incident management, technician assignment, preventive maintenance scheduling, repair management, material request and approval, and a machine functionality indicator. With its implementation, companies can expect to reduce maintenance costs, increase machine availability, and improve the efficiency of their operational processes.

1.3 Involved Personnel

Name:	Arellano Aramburo Jocelyn Astrid
Role:	Project Leader
Contact:	0322103847@ut-tijuana.edu.mx
Responsibilities:	Software developer and database manager.

Name:	Bravo Flores Selegna Odracir
Role:	Documentation Manager
Contact:	0322103684@ut-tijuana.edu.mx
Responsibilities:	Software developer and documentation manager.

Name:	Zavala Rodríguez Jesús Armando
Role:	System Analyst
Contact:	0323106191@ut-tijuana.edu.mx
Responsibilities:	Software developer and system manager

1. 4. Definitions, acronyms and abbreviations

Name	Description
SRS	Software Requirements Specification.
MEI	Maintenance for Industrial Equipment (System name).
HTML	HyperText Markup Language.
PHP	HyperText Preprocessor.
MySQL	My Structured Query Language,
NA	Not Applicable.
RF	Functional Requirements.
RNF	Non-Functional Requirements.

1. 5. Summary

This Software Requirements Specification (SRS) document describes the details of the "Maintenance Management System for Industrial Equipment (MEI)" project, covering key aspects such as the system's purpose, scope, functionalities, and technical diagrams.

The Introduction provides an overview of the document, defining the system's purpose, scope, and involved personnel. It also presents the main definitions, acronyms, and abbreviations used throughout the project. This section offers the necessary context to understand the system's objectives and scope.

The General Description of the system includes a detailed view of the product perspective and functionality, describing how the system is expected to operate within the work environment. It also addresses user characteristics, restrictions that may affect the system's development or operation, and assumptions and dependencies that have been considered for the system's planning and future evolution.

The Functional and Non-Functional Requirements specify the functionalities the system must meet, from the main tasks it must perform to performance, security, and other essential attributes to ensure software quality.

The Appendix provides important technical diagrams to support the understanding and development of the system. These include the Entity-Relationship Diagram, the database's relational model, the system's distributed architecture design, and the Gantt chart that illustrates the project's timeline.

The Use Cases section includes diagrams representing the system's dynamic behavior, detailing how external actors interact with the system through state transition diagrams, sequence diagrams, and activity diagrams.

1. 6. References

- Felipe Neon, G. T. (2019) . El mantenimiento preventivo de la maquinaria industrial: Qué es, tipos, ventajas y beneficios.
- Oscar Vásquez, O. M & Hernández Samuel, M. F. (2013). La importancia del mantenimiento en instalaciones industriales. Seguas
- RSR Definición: Revisión de la estrategia de los requisitos - Requirements Strategy Review. (n.d.). https://www.abbreviationfinder.org/es/acronyms/rsr_requirements-strategy-review.html
- Admin. (2019). Especificación de requisitos de software (SRS): sugerencias y plantilla. Visure Solutions. <https://visuresolutions.com/es/software-requirement-specification-srs-tips-template/>
- Asana, T. (2022, May 6). Cómo redactar un documento de requisitos de software (incluye una plantilla) [2022] • Asana. Asana. <https://asana.com/es/resources/software-requirement-document-template>
- Plantilla Especificación de Requisitos | Marco de Desarrollo de la Junta de Andalucía. (n.d.). <https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/456>
- Solutions, V., & Jain, A. (2023). Qué son los requisitos funcionales: ejemplos, definición, guía completa. Visure Solutions. <https://visuresolutions.com/es/blog/functional-requirements/>

- Solutions, V., & Jain, A. (2023b). Qué son los requisitos no funcionales: ejemplos, definición, guía completa. Visure Solutions.
[https://visuresolutions.com/es/blog/nonfunctionalrequirements/#:~:text=Los%20requisitos%20no%20funcionales%20\(NFR,la%20confiabilidad%20y%20muchos%20m%C3%A1s.](https://visuresolutions.com/es/blog/nonfunctionalrequirements/#:~:text=Los%20requisitos%20no%20funcionales%20(NFR,la%20confiabilidad%20y%20muchos%20m%C3%A1s.)

2. General Description

2.1 Product Perspective

The Maintenance Management System for Industrial Equipment (MEI) is envisioned as a comprehensive solution for managing machinery maintenance in industrial plants. This product will be designed as a centralized platform, accessible through a web interface developed using HTML, PHP, and MySQL technologies. The system will serve to automate and optimize processes related to preventive and corrective maintenance, as well as incident management and resource administration.

MEI will enable greater efficiency in equipment monitoring, reducing downtime and costs associated with unexpected repairs. It will provide a consolidated database that centralizes all relevant information to improve decision-making regarding maintenance and the lifecycle of machinery.

2.2 Product Functionality

The Maintenance Management System for Industrial Equipment (MEI) is designed to offer a set of critical functionalities that optimize the maintenance process of industrial equipment. These functionalities range from incident management to preventive maintenance scheduling, all supported by a robust database and an efficient automation system. The main functionalities are detailed below.

2.2.1 Incident Management

Description: The system will allow operators to report anomalies or failures detected in equipment. Each incident will be recorded with details such as the date, time, affected equipment, and description of the problem.

User Actions:

- The operator generates an incident when they detect a problem with a machine.
- The system automatically assigns the incident to the appropriate technician based on specialization, workload, and the technician's work shift.

Objective: Ensure efficient and quick incident management, reducing equipment downtime.

2.2.2 Automatic Technician Assignment

Description: When a new incident is logged, the system automatically assigns an available technician based on the technician's specialization, current workload, and work shift (morning or afternoon).

User Actions:

- The technician receives a notification with the details of the assigned incident.
- The system updates the list of incidents assigned to each technician, avoiding overloading.

Objective: Automate the equitable distribution of work among technicians, preventing workload overload and maximizing efficiency.

2.2.3 Preventive Maintenance Scheduling

Description: The system will allow preventive maintenance scheduling based on equipment usage history and condition. The system will recommend dates and maintenance tasks for each machine based on its operation time and current conditions.

User Actions:

- The technician or manager can view the preventive maintenance calendar.
- The system will automatically generate reminders for upcoming maintenance dates.

Objective: Minimize the risk of unexpected failures by scheduling preventive maintenance based on actual data.

2.2.4 Repair Management

Description: Technicians can record repairs made to equipment, detailing the issues encountered, time spent, and materials used. After each repair, the system will update the equipment status and recalculate its profitability.

User Actions:

- The technician logs a repair and updates the equipment's status.
- The manager can view a detailed report of the repairs performed.

Objective: Maintain comprehensive control of all repairs to evaluate the efficiency and profitability of continuing to repair the equipment.

2.2.5 Material Request and Approval

Description: When a technician identifies the need for additional materials to perform a repair or maintenance, they can submit a request through the system. The system will notify the manager to review and approve or reject the request.

User Actions:

- The technician submits a material request, specifying the required items and quantities.
- The manager reviews the request, and the warehouse processes the delivery if approved.

Objective: Facilitate the management of materials needed for repairs and maintenance, ensuring resource availability and controlling associated costs.

2.2.6 Machine Functionality Indicator

Description: The system will implement a functionality indicator for each machine, replacing the traditional availability percentage. This indicator will reflect the machine's efficiency and reliability, considering operation time, repairs performed, and downtime.

User Actions:

- Administrators can view the real-time functionality indicator for each piece of equipment.
- The system automatically calculates the functionality of each machine.

Objective: Provide a more accurate view of each machine's performance, helping managers make informed decisions about its future, such as repair or replacement.

2.2.7 Reports and Data Analysis

Description: The system will allow the generation of detailed reports on incidents, maintenance, repairs, and associated costs. These reports will help managers identify failure patterns, assess technician effectiveness, and make strategic decisions to improve operational efficiency.

User Actions:

- The manager can generate reports filtered by dates, machine types, technicians involved, and costs.

Objective: Provide analysis tools to evaluate maintenance efficiency and optimize plant operations.

2.3 User Characteristics

This maintenance management system is designed for three main types of users: operators, technicians, and managers, each with clearly defined responsibilities within the maintenance cycle.

- **Operator:** This user is responsible for operating the machines in the plant. When the operator detects any anomaly in the equipment's operation, they can report an incident through the system, which generates an alert for the maintenance team to take the necessary actions. It is important to note that the operator does not have the authority to assign incidents to technicians; they are limited to reporting problems with the equipment only.
- **Technician:** Responsible for repairs and corrective and preventive maintenance on the machines. The technician also manages the incidents reported by the operators and provides appropriate solutions. Additionally, suppose materials are needed to complete a repair or maintenance task. In that case, the technician has access to the system to request those materials if they are not available in the warehouse.
- **Manager:** The manager has a more administrative role within the system. Their primary responsibility is to generate and review reports related to the status of the equipment, reported incidents, and performed repairs. Although they can view all the data, they do not have permission to modify it. Moreover, the manager is responsible for approving the material requests generated by the technicians.

2.4 Constraints

The MEI system presents certain limitations that should be considered during its implementation:

- No mobile application: The system will not include a version for mobile devices in its initial phase, limiting its use to desktop environments.
- Internet connection dependency: Access and use of the system will require a stable Internet connection, as all functionalities depend on online servers to process and store data.
- Restricted access for authorized users: Only users with the appropriate permissions and authorized access can use the system. Each employee will have a limited access level according to their role in the organization.
- Data validation: If the user enters incorrect information, such as letters in numeric fields or data outside the expected parameters, the system will display error messages to guide the user in correcting the information.
- Technician assignment: The system will not allow more than one technician to be assigned to the same incident; a single technician will resolve each incident to avoid confusion or duplication of efforts.

2.5 Assumptions and Dependencies

The development and functioning of the MEI system are based on the following assumptions and dependencies:

- Availability of technical staff: It is assumed that the maintenance technical staff will be available to receive notifications of incidents and work assignments generated by the system at all times, according to the established shifts.
- Constant connectivity: The system relies on a stable connection to the local network or the internet to synchronize data, perform remote operations, and ensure user accessibility in different locations.
- User training: It is assumed that all personnel interacting with the system have been adequately trained in its use, minimizing operational errors and facilitating software administration.
- Continuous technical support: The system's proper functioning will depend on the technical support available to resolve technological incidents, make adjustments, and ensure preventive software maintenance.

2.6 Predictable Evolution of the System

Several improvements are anticipated for the MEI system as it evolves:

- **User interface enhancements:** The system will feature continuous improvements in its graphical interface to provide a more intuitive experience. The improvements will include the implementation of interactive dashboards and customized reports for more precise and efficient visualization of equipment status and incidents.
- **Mobile device compatibility:** In the long term, there are plans to develop a mobile application, allowing users to access the system from anywhere, facilitating remote management and real-time monitoring.

3. Requirements

3.1 Functional Requirements.

Code	RF-001
Requirement Name	Equipment registration
Type	Functional
Requirement Source	Maintenance manager
Requirement's Priority	High
Requirement Description	The system should allow new equipment to be registered with detailed information (serial number, manufacturer, model, etc.).

Code	RF-002
Requirement Name	Equipment management
Type	Functional
Requirement Source	Maintenance manager
Requirement's Priority	High
Requirement Description	The system should allow you to edit, delete and view information on registered computers.

Code	RF-003
Requirement Name	Status report of the equipment
Type	Functional
Requirement Source	Maintenance manager
Requirement's Priority	Medium
Requirement Description	The system should generate reports on the condition of equipment, indicating whether it requires preventive or corrective maintenance.

Code	RF-004
Requirement Name	Incident log
Type	Functional
Requirement Source	Manager, technical
Requirement's Priority	High
Requirement Description	The system should allow the recording of incidents associated with the equipment, specifying details such as type of problem, technical assigned, date, etc.

Code	RF-005
Requirement Name	Automatic allocation of incidents
Type	Functional
Requirement Source	Maintenance manager
Requirement's Priority	High
Requirement Description	The system should automatically assign incidents to technicians according to their shift and workload.

Code	RF-006
Requirement Name	History of incidents.
Type	Functional
Requirement Source	Manager, technical
Requirement's Priority	Medium
Requirement Description	The system should record the history of incidents for each computer, with resolution details.

Code	RF-007
Requirement Name	Maintenance scheduling
Type	Functional
Requirement Source	Maintenance manager
Requirement's Priority	High
Requirement Description	The system must allow the scheduling of preventive and corrective maintenance, with automatic alerts to technicians.

Code	RF-008
Requirement Name	Maintenance history
Type	Functional
Requirement Source	Manager, technical
Requirement's Priority	Medium
Requirement Description	The system should record the maintenance history of each computer.

Code	RF-009
Requirement Name	Material management
Type	Functional
Requirement Source	Warehouse manager, Technician
Requirement's Priority	High
Requirement Description	The system should allow the request and management of materials necessary for repairs and maintenance, updating the inventory automatically.

Code	RF-010
Requirement Name	Employee management
Type	Functional
Requirement Source	Human resources manager, Area manager
Requirement's Priority	High
Requirement Description	The system should allow to record, edit and manage information of employees (managers, technicians, operators), including their role and shift.

Code	RF-011
Requirement Name	Report generation
Type	Functional
Requirement Source	Maintenance manager
Requirement's Priority	Medium
Requirement Description	The system should generate detailed reports on equipment functionality, incidents, maintenance and use of materials.

Code	RF-012
Requirement Name	Push notifications
Type	Functional
Requirement Source	Manager, technical
Requirement's Priority	High
Requirement Description	The system should send automatic notifications when a team needs maintenance, new incidents, or when a request for materials is raised.

Code	RF-013
Requirement Name	Access control
Type	Functional
Requirement Source	Security department
Requirement's Priority	High
Requirement Description	The system should allow users to access functionality only according to their role (technical, manager, operator) through role-based authentication.

3.2 No functional Requirements

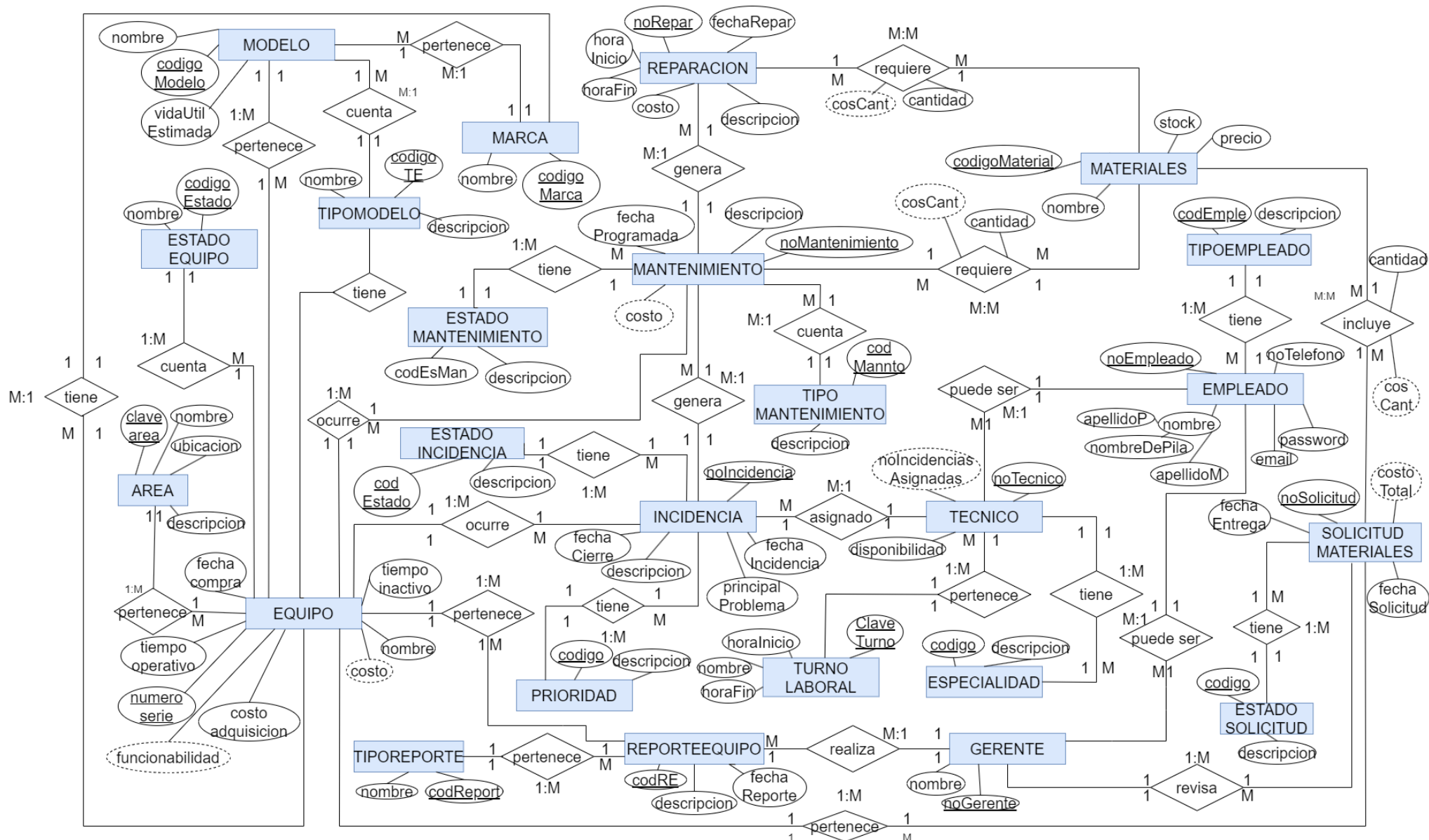
Code	RNF-001
Requirement Name	Response time
Type	No functional
Requirement Source	Final users
Requirement's Priority	High
Requirement Description	The system should process queries and records in a time not exceeding 2 seconds under normal load.

Code	RNF-002
Requirement Name	Scalability
Type	No functional
Requirement Source	Maintenance manager
Requirement's Priority	Medium
Requirement Description	The system must be able to withstand expansion to include more equipment and personnel without significant performance loss.

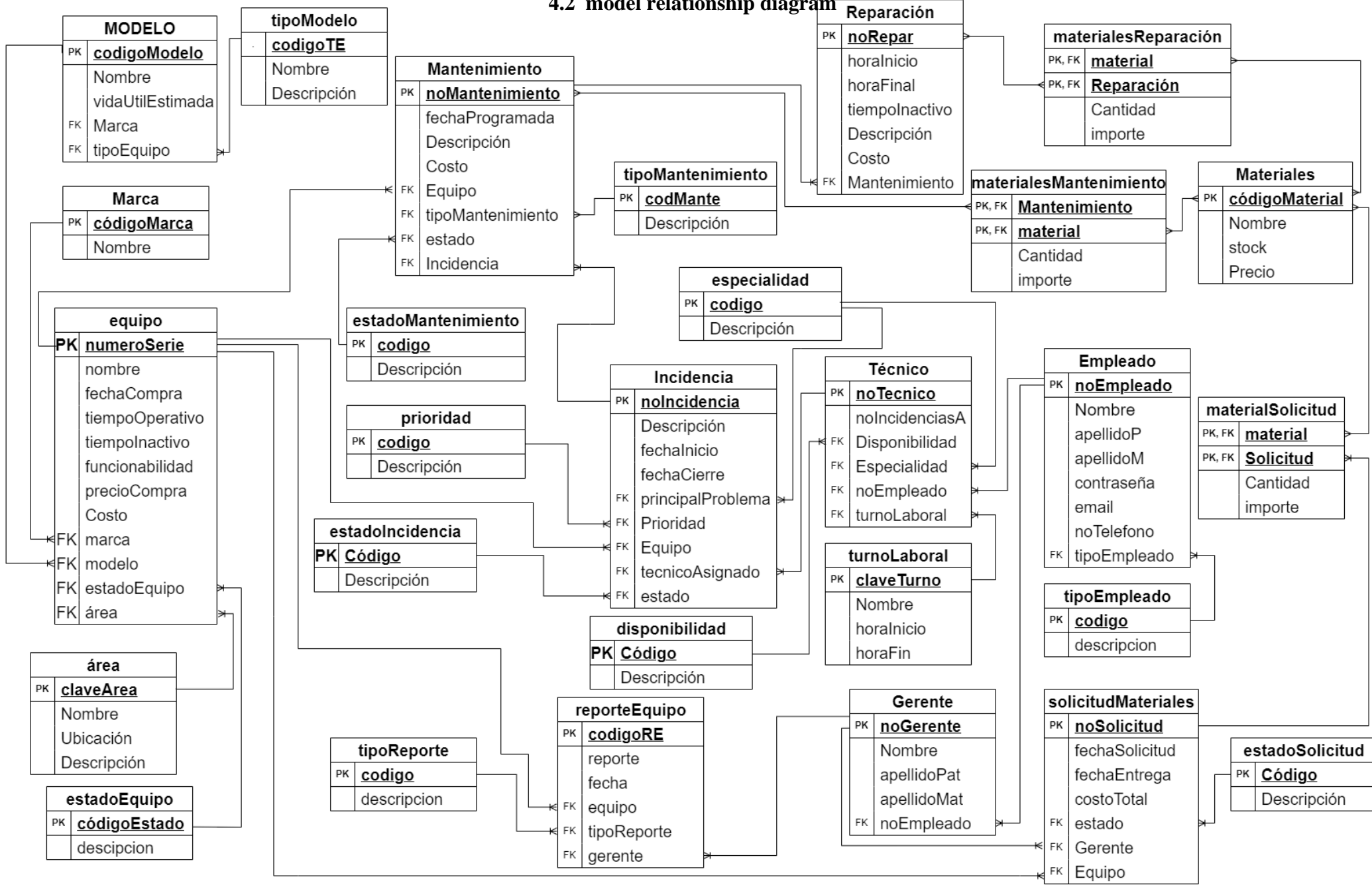
Code	RNF-003
Requirement Name	Data encryption
Type	No functional
Requirement Source	Security department
Requirement's Priority	High
Requirement Description	The system must have encryption of sensitive data such as credentials and computer logs.

Code	RNF-004
Requirement Name	Availability
Type	No functional
Requirement Source	Maintenance Manager, IT
Requirement's Priority	High
Requirement Description	The system should be available 24/7 with a downtime of no more than 1% per month.

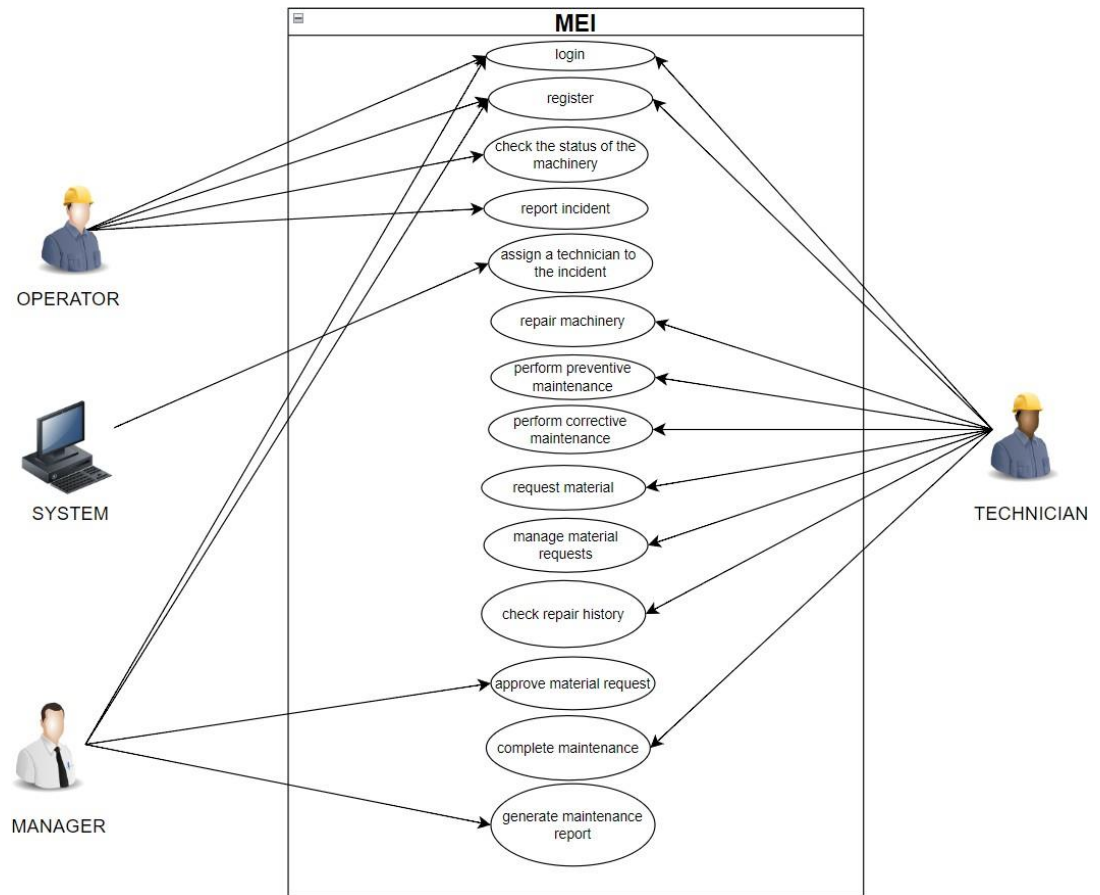
Code	RNF-005
Requirement Name	Support for modern browsers
Type	No functional
Requirement Source	Final users
Requirement's Priority	Medium
Requirement Description	The system should be accessible from modern browsers such as Chrome, Firefox and Edge, without the need to install additional software.



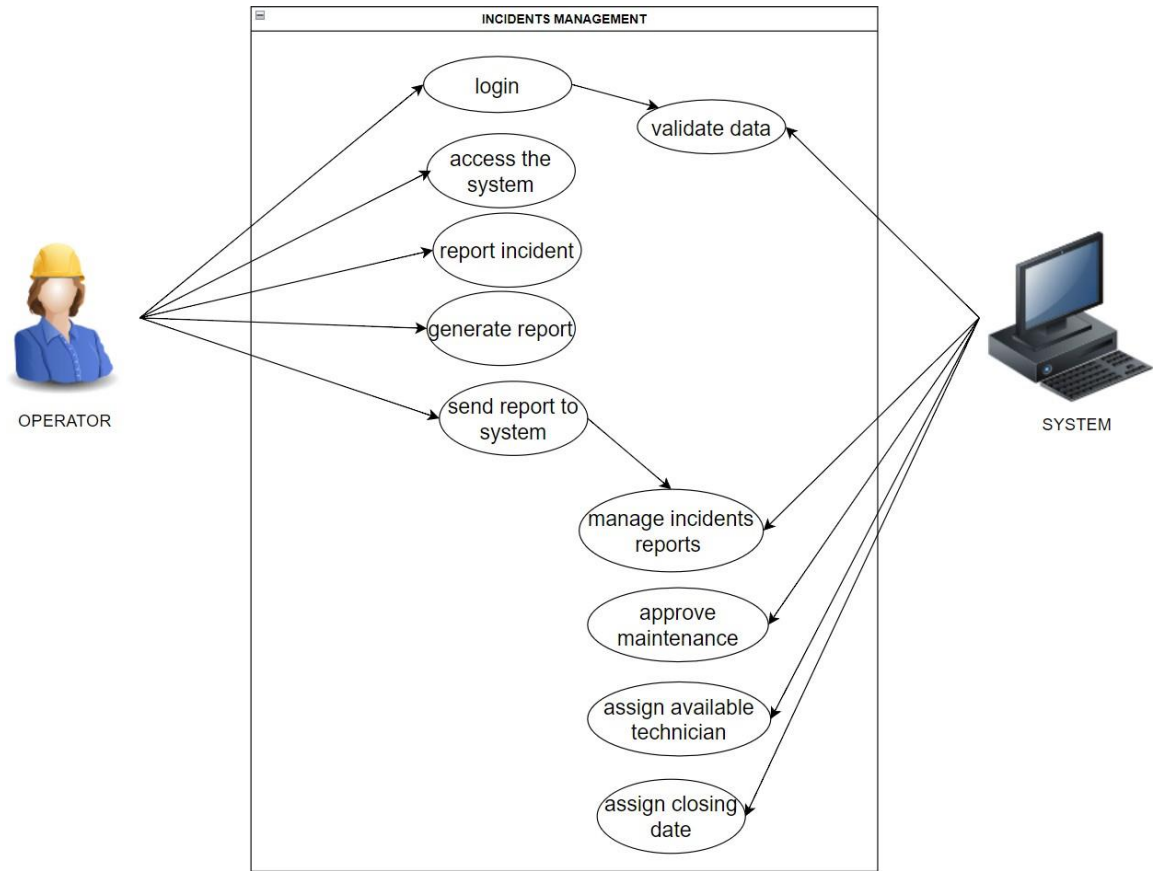
4.2 model relationship diagram



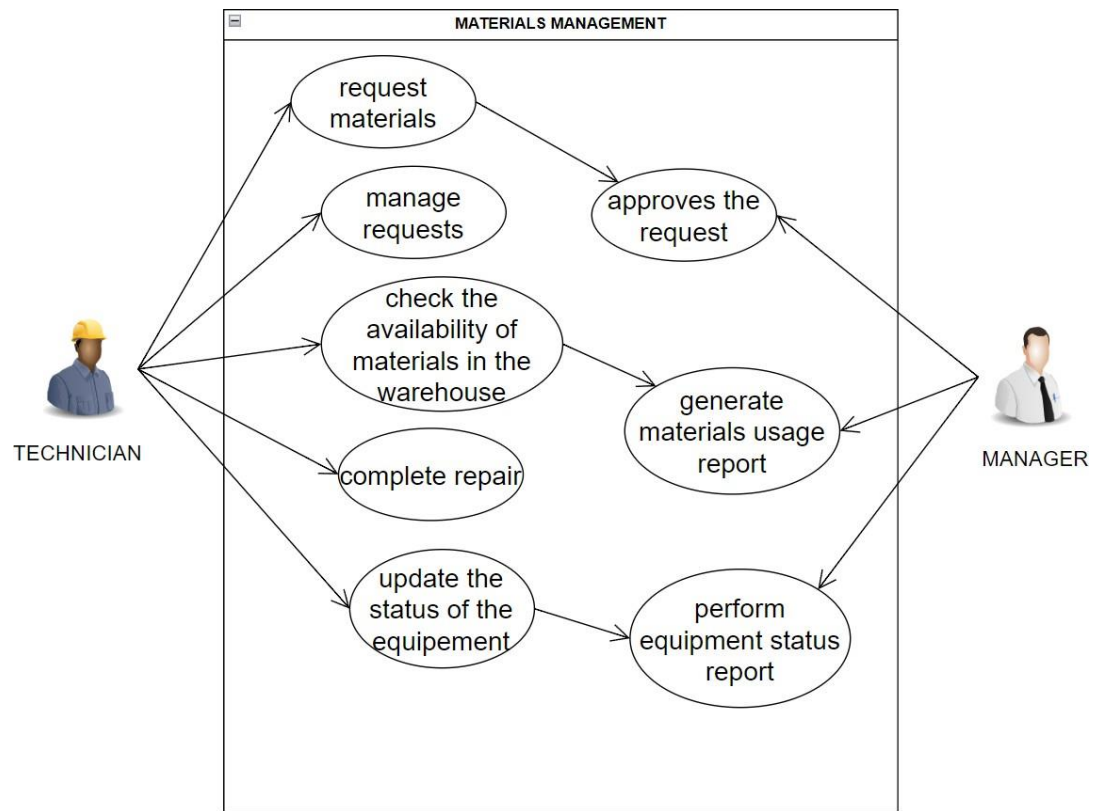
4.3 Use case diagram



4.3.1 Operator

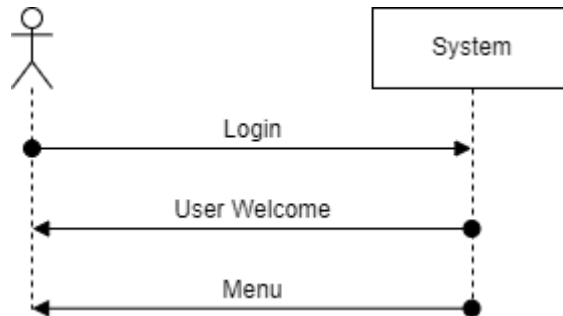


4.3.2 Technicians

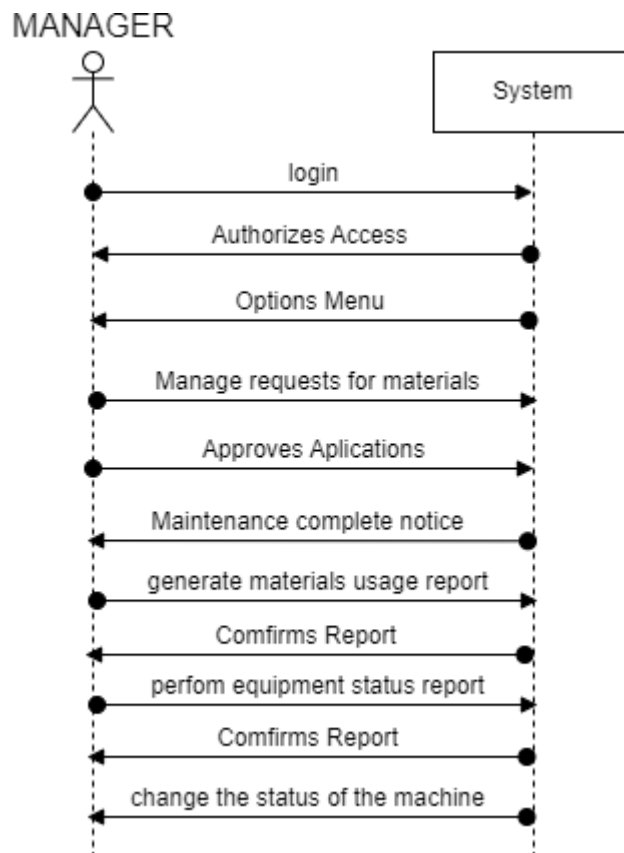


4.4 Sequence diagrams

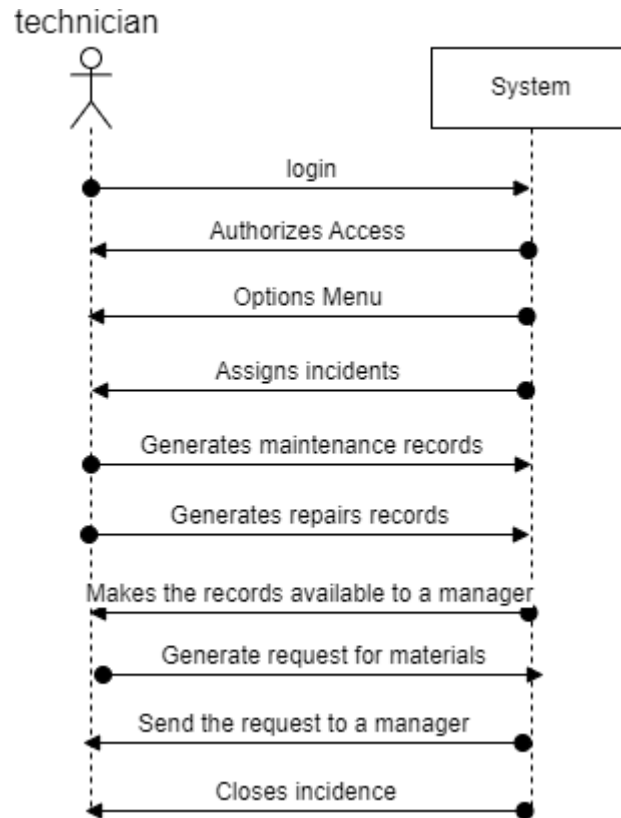
4.4.1 Operator



4.4.2 Manager

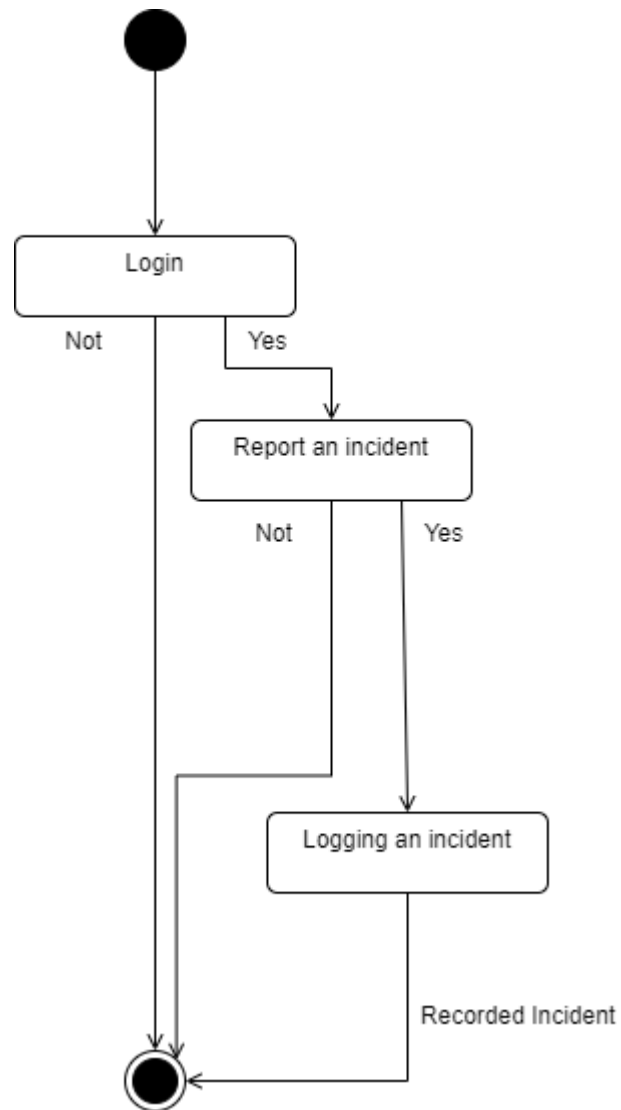


4.4.3 Technician

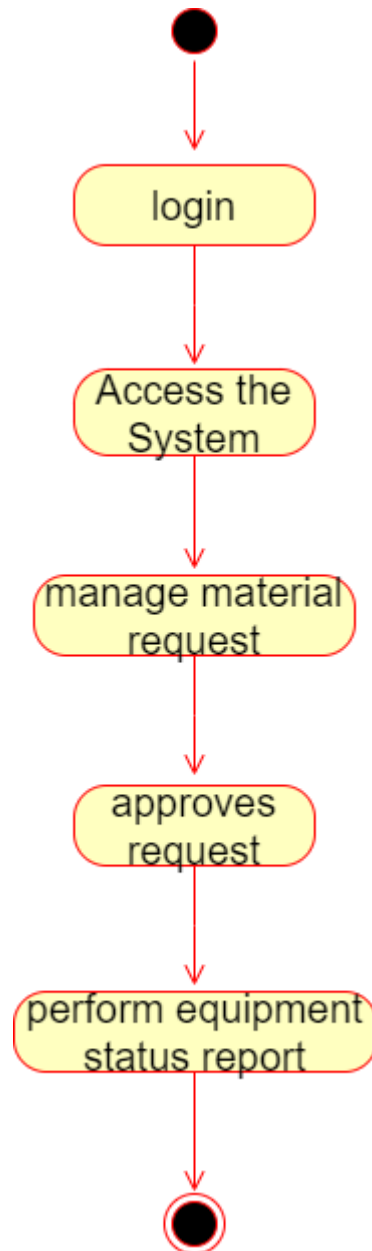


4.5 Status diagrams

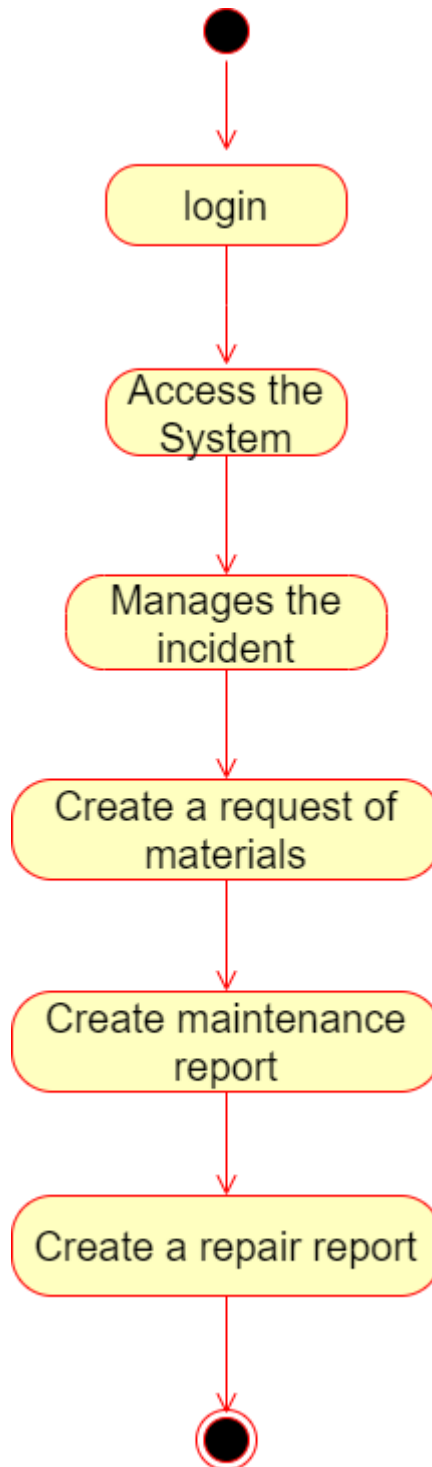
4.5.1 Operator



4.5.2 Manager

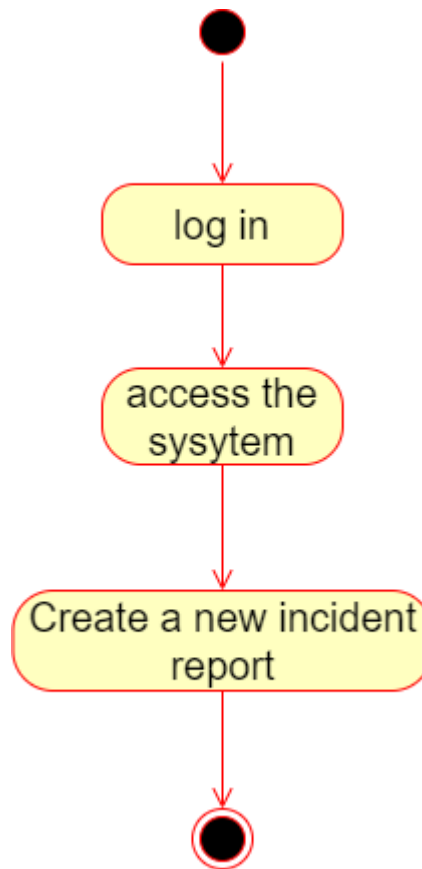


4.5.3 Technician

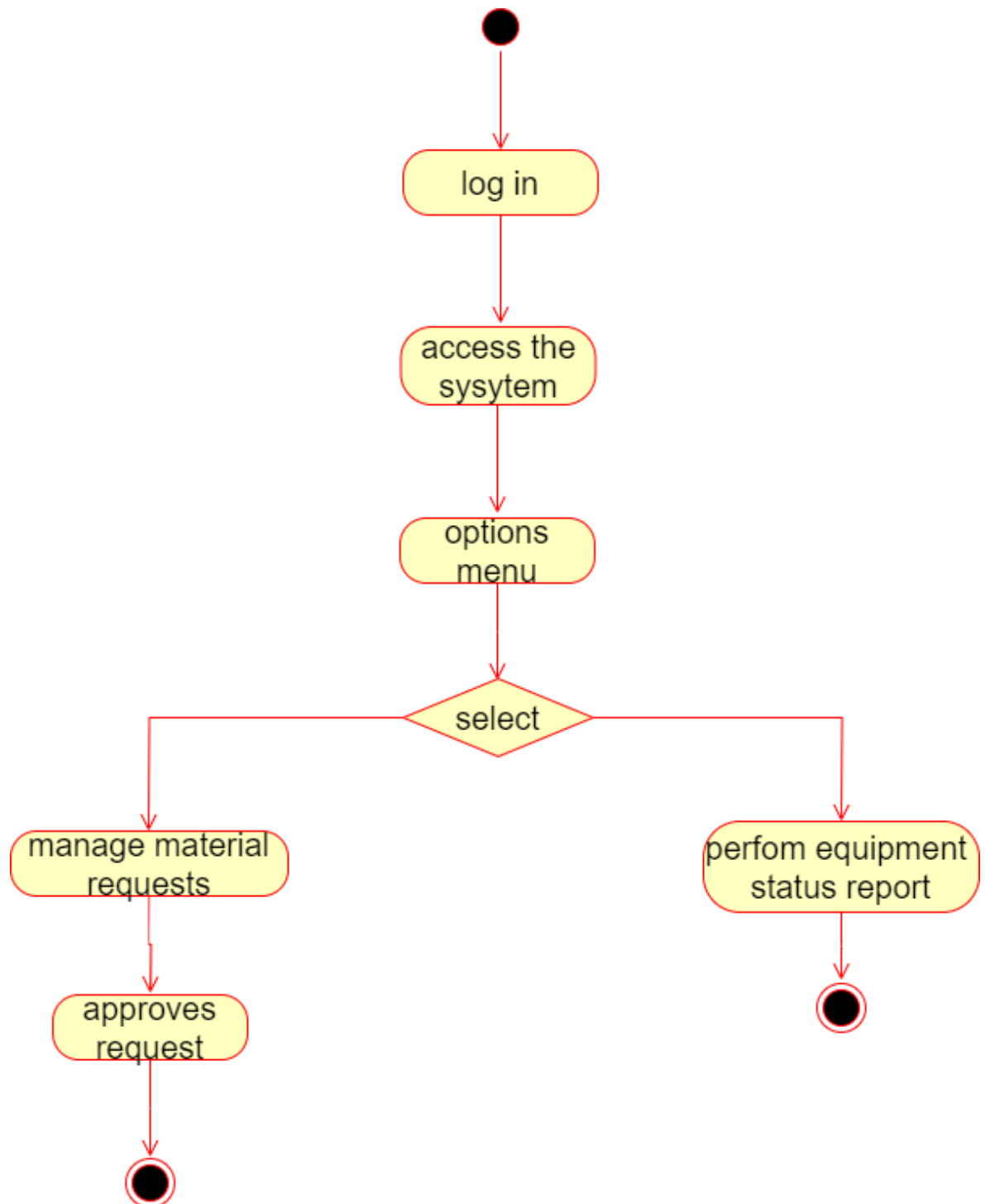


4.6 Activity diagrams

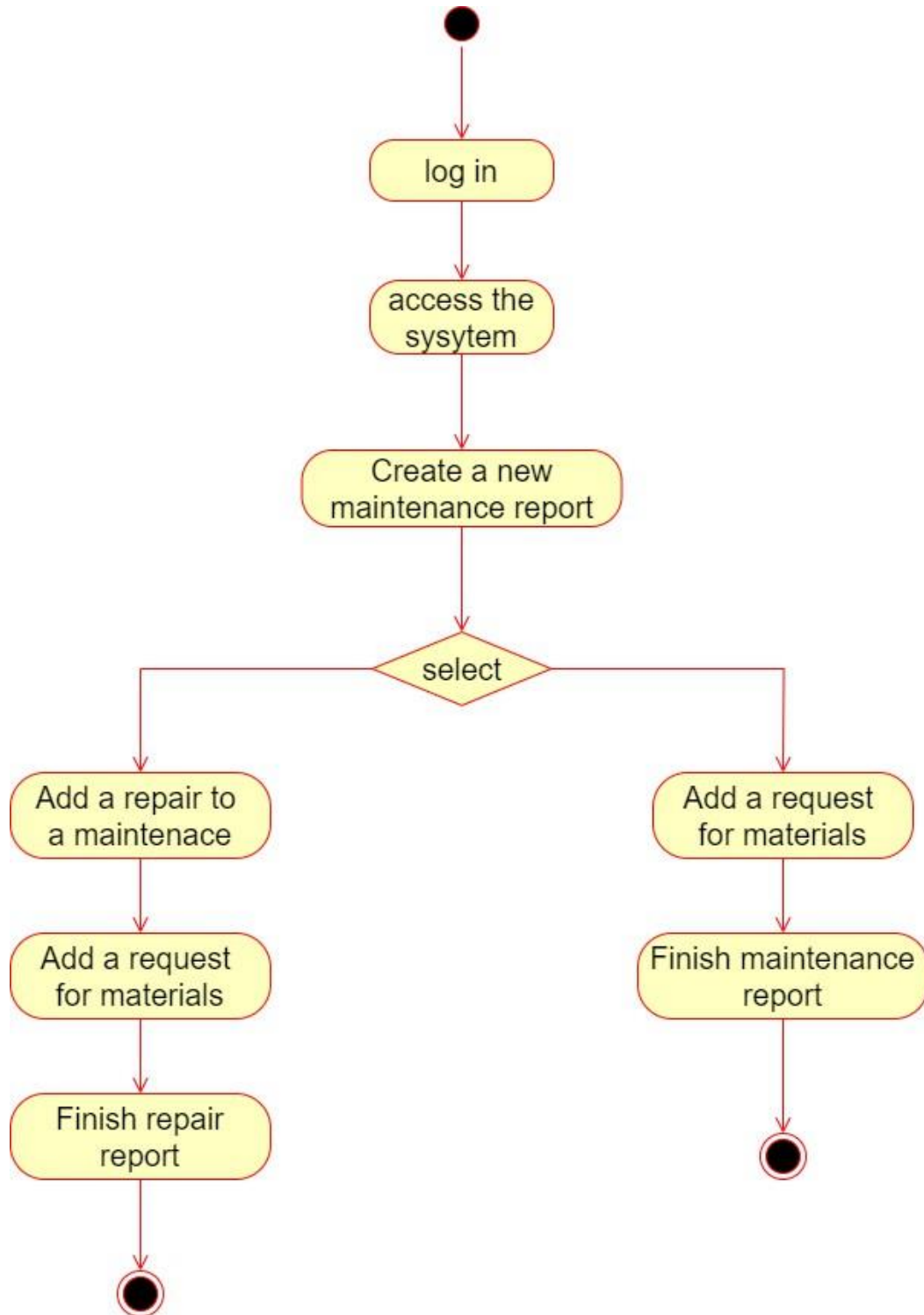
4.6.1 Operator



4.6.2 Manager

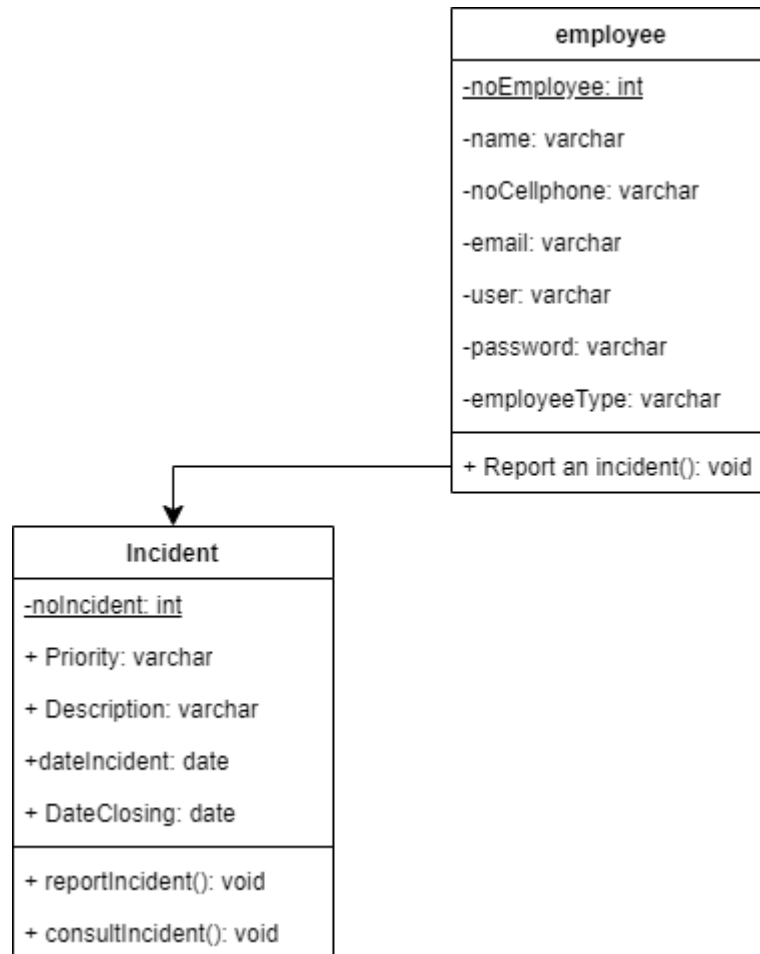


4.6.3 Technician

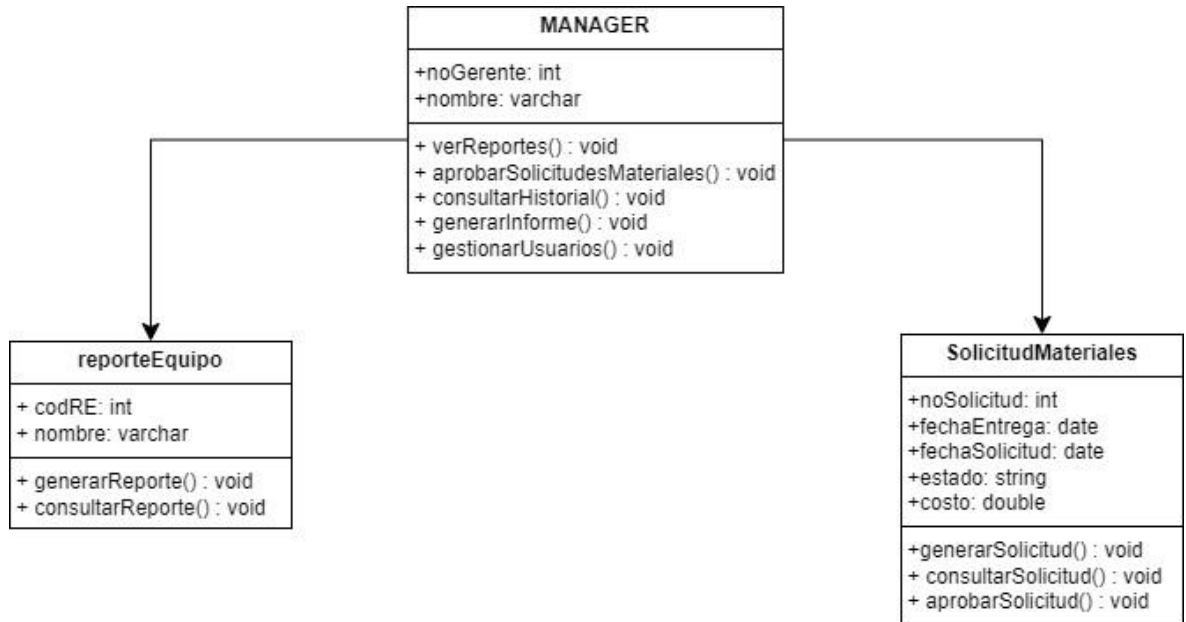


4.7 Class diagrams

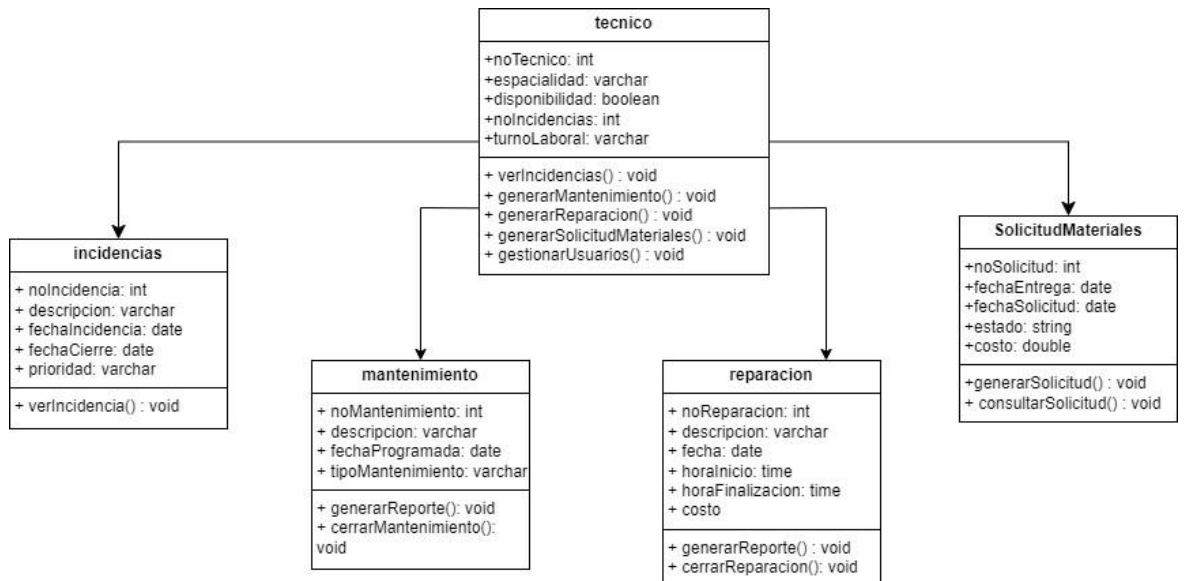
4.7.1 Operator



4.7.2 Manager



4.7.3 Technician



4.8 Project code explanation

4.8.1 Login

```

CREATE PROCEDURE datosInicioSesion(
    IN pas VARCHAR(255),
    IN numE INT,
    OUT nom VARCHAR(70),
    IN tipo VARCHAR(15)
)
BEGIN
    DECLARE msg VARCHAR(100);
    DECLARE tipoE VARCHAR(15);
    SELECT tipoEmpleado INTO tipoE
    FROM empleado
    WHERE noEmpleado = numE;
    IF EXISTS (
        SELECT 1
        FROM empleado
        WHERE noEmpleado = numE
        AND password = SHA2(pas, 256)
        AND tipoE = tipo
    ) THEN
        SET nom = (SELECT CONCAT(nombre, ' ', apellidoP, ' ', apellidoM)
        FROM empleado
        WHERE noEmpleado = numE);
        SET msg = CONCAT('Welcome ', nom);
    ELSE
        SET msg = "Employee number, password, or employee type incorrect. Please check your login information.";
    END IF;

    SELECT msg AS mensaje, nom AS nombreCompleto, tipoE AS tipoEmpleado;
END$$

```

The stored procedure `datosInicioSesion` is designed to manage user login sessions for employees. It takes four parameters: an input password (`pas`), an employee number (`numE`), an output variable for the employee's full name (`nom`), and an input employee type (`tipo`). The procedure begins by declaring two local variables: `msg` for storing a user message and `tipoE` for holding the retrieved employee type from the database. It retrieves the employee type associated with the given `numE` from the `empleado` table and stores it in `tipoE`. The procedure then checks if a record exists in the `empleado` table that matches the provided `numE`, has the correct password (hashed using SHA-256), and matches the provided `tipo`. If the credentials are valid, it constructs the employee's full name by concatenating their first name, paternal surname, and maternal surname, assigning it to `nom`, and creating a welcome message. If the credentials are invalid, it sets `msg` to indicate an error in the login attempt. Finally, the procedure returns a result set containing the message (`msg`), the employee's full name (`nom`), and the employee type (`tipoE`). Overall, its purpose is to authenticate an employee's login and provide personalized feedback based on the validity of their

credentials.

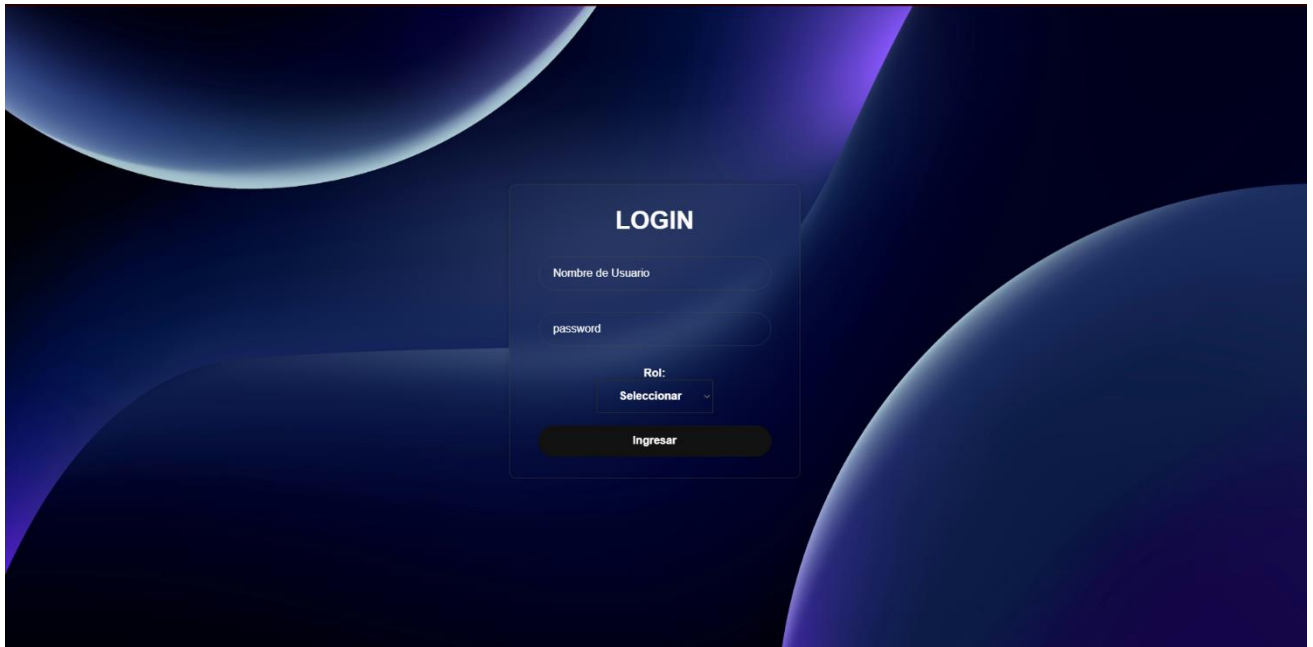
```
<body>
  <div class="wrapper">
    <?php var_dump($_POST);?>

    <form action="app/login.php" method="POST">
      <h1>LOGIN</h1>
      <div class="input-box">
        <input type="number" name="txtnick" placeholder="No. Employee" required>
      </div>
      <div class="input-box">
        <input type="password" name="txtpass" placeholder="Password" required>
      </div>
      <h4>Rol: </h4>
      <center>
        <select name="rol" required>
          <option value="0" style="display:none;">Choose</option>
          <option value="OPE">Operator</option>
          <option value="TEC">Technician</option>
          <option value="GER">Manager</option>
        </select>
      </center>
      <br>
      <button type="submit" class="btn">Login</button>
    </form>
```

The provided HTML code represents a login form that is designed for user authentication within a web application. It uses the POST method to submit data to `app/login.php`. The form begins with a header (`<h1>LOGIN</h1>`) to indicate its purpose. Inside the form, there are two input fields: the first is a number input (`<input type="number" name="txtnick" placeholder="No. Employee" required>`) for the employee number, which is mandatory, and the second is a password input (`<input type="password" name="txtpass" placeholder="Password" required>`) for the user to enter their password, also required for submission. Below the input fields, a `<h4>` tag is used to label the role selection. The role is chosen from a dropdown menu (`<select name="rol" required>`), which includes three options: Operator, Technician, and Manager. The first option is hidden until a selection is made. Finally, there is a submit button (`<button type="submit" class="btn">Login</button>`) that allows the user to submit the form. Overall, this form is structured to facilitate secure user login by collecting the necessary credentials and employee role before sending the information to the server for validation.

Web design for the login, the background image is still tentative.

We can see the combo-box that will be used for workers to choose what type of employees they are:



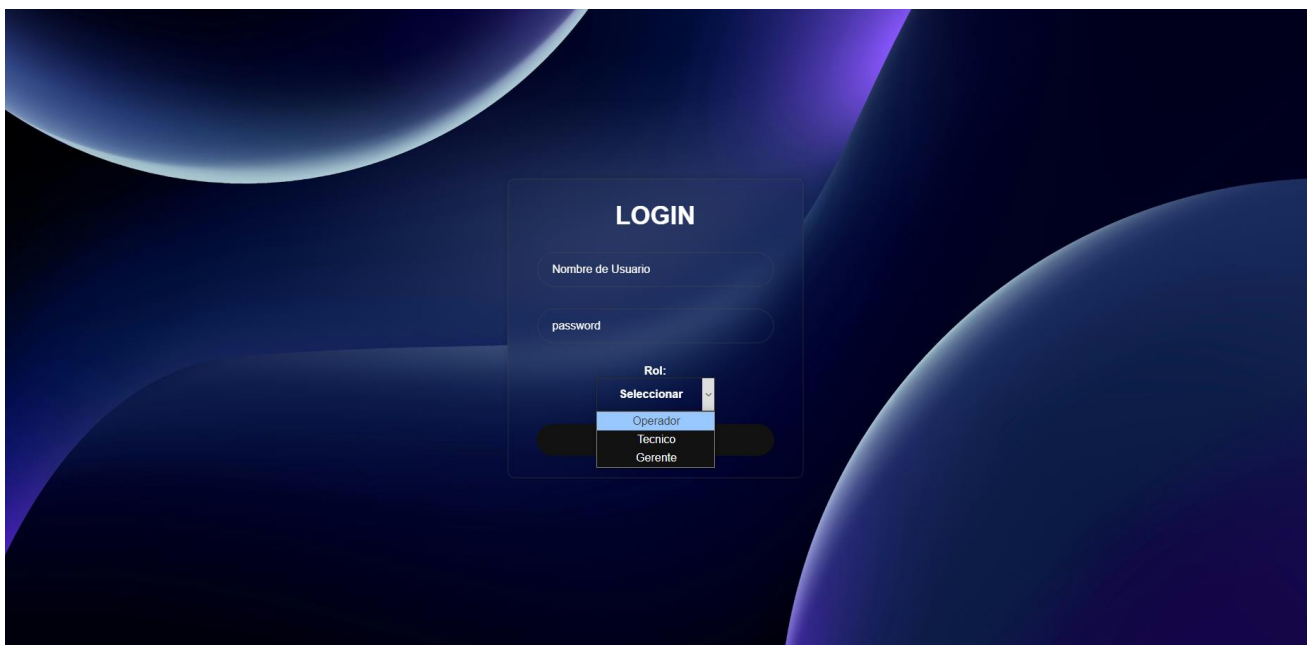
LOGIN

Nombre de Usuario

password

Rol:
Seleccionar

Ingresar



LOGIN

Nombre de Usuario

password

Rol:
Seleccionar

- Operador
- Técnico
- Gerente

4.8.2 login authenticity

```
<?php
include('../data/employee.php');

session_start();

$myuser = new Employee();
$myuser->setNickname($_POST['txtnick']);
$myuser->setPassword($_POST['txtpass']);
$rol = $_POST['rol'];

$userData = $myuser->getUserData($rol);

if ($userData != 'wrong') {
    $registro = mysqli_fetch_assoc($userData);
    $_SESSION['numEmpleado'] = $_POST['txtnick'];
    $_SESSION['nombreCompleto'] = $registro['nombreCompleto'];
    $_SESSION['tipoEmpleado'] = $registro['tipoEmpleado'];

    echo "Tipo de Empleado: " . $_SESSION['tipoEmpleado'];

    switch ($_SESSION['tipoEmpleado']) {
        case 'OPE':
            header("Location: http://localhost/maquinaria/operator.php");
            break;
        case 'TEC':
            header("Location: ../technician.php");
            break;
        case 'GER':
            header("Location: ../manager.php");
            break;
        default:
            header("Location: ../login.php?error=1");
            break;
    }
    exit();
} else {
    header("Location: ../login.php?error=1");
    exit();
}
?>
```

The script begins by including the `employee.php` file, which presumably contains the definition of the `Employee` class. This class provides methods for handling employee-related data, including setting the user's nickname and password.

```
include('../data/employee.php');
```

Next, the script starts a new session or resumes the existing session with `session_start()`. This allows the script to store user-related data in session variables, which can be accessed across different pages.

```
session_start();
```

An instance of the `Employee` class is created, and the user's nickname (employee number) and password are set using the `setNickname` and `setPassword` methods, respectively. The selected role (`rol`) is also retrieved from the submitted form data.

```
$myuser = new Employee();  
$myuser->setNickname($_POST['txtnick']);  
$myuser->setPassword($_POST['txtpass']);  
$rol = $_POST['rol'];
```

The script then attempts to fetch user data by calling the `getUserData` method with the provided role. This method checks the credentials against the database.

```
$userData = $myuser->getUserData($rol);
```

If the user data retrieval is successful (i.e., not equal to 'wrong'), the script fetches the user record as an associative array using `mysqli_fetch_assoc()`. It then stores the employee number, full name, and employee type in session variables.

```
if ($userData != 'wrong') {
```

```
$registro = mysqli_fetch_assoc($userData);  
$_SESSION['numEmpleado'] = $_POST['txtnick'];  
$_SESSION['nombreCompleto'] = $registro['nombreCompleto'];  
$_SESSION['tipoEmpleado'] = $registro['tipoEmpleado'];
```

A message displaying the employee type is echoed for confirmation.

```
echo "Tipo de Empleado: " . $_SESSION['tipoEmpleado'];
```

The script then uses a `switch` statement to redirect the user to different pages based on their employee type:

- Operators (OPE) are redirected to `operator.php`.
- Technicians (TEC) are redirected to `technician.php`.
- Managers (GER) are redirected to `manager.php`.
- If the employee type does not match any of these, the user is redirected back to the login page with an error indicator.

```
switch ($_SESSION['tipoEmpleado']) {  
    case 'OPE':  
        header("Location: http://localhost/maquinaria/operator.php");  
        break;  
    case 'TEC':  
        header("Location: ../technician.php");  
        break;  
    case 'GER':  
        header("Location: ../manager.php");  
        break;  
    default:  
        header("Location: ../login.php?error=1");  
        break;  
}
```

```
exit();  
...
```

If the user data retrieval is unsuccessful (i.e., it returns 'wrong'), the user is redirected to the login page with an error indicator.

```
} else {  
    header("Location: ../login.php?error=1");  
    exit();  
}
```

```
public function getUserData($rol) {  
    $connection = connection();  
    $numE = $this->nickname;  
    $password = $this->password;  
  
    $query = "CALL datosInicioSesion(?, ?, @nombreCompleto, @tipoEmpleado)";  
    $stmt = $connection->prepare($query);  
  
    if ($stmt === false) {  
        die("Error en la preparación de la consulta: " . $connection->error);  
    }  
  
    $stmt->bind_param("si", $password, $numE);  
    $stmt->execute();  
    $stmt->close();  
  
    $result = $connection->query("SELECT @nombreCompleto AS nombreCompleto, @tipoEmpleado AS tipoEmpleado");  
  
    if ($result && $result->num_rows > 0) {  
        return $result;  
    } else {  
        return 'wrong';  
    }  
}
```

The PHP function `getUserData` is a method designed to retrieve user data related to an employee based on their credentials.

The function begins by establishing a connection to the database through a function call to `connection()`, which is expected to return a database connection object.

```
$connection = connection();
```

It retrieves the employee number (`numE`) and password from the current instance of the class, which has properties `nickname` (for employee number) and `password`.

```
$numE = $this->nickname;  
$password = $this->password;
```

The function prepares a stored procedure call to `datosInicioSesion`, which handles the login logic. It uses placeholders (`?`) for the parameters to prevent SQL injection.

The stored procedure expects two input parameters (the password and employee number) and sets two output parameters (`@nombreCompleto` and `@tipoEmpleado`) for the user's full name and employee type.

```
$query = "CALL datosInicioSesion(?, ?, @nombreCompleto, @tipoEmpleado)";  
$stmt = $connection->prepare($query);
```

The function checks if the statement preparation was successful. If not, it terminates execution and outputs an error message.

```
if ($stmt === false) {  
    die("Error en la preparación de la consulta: " . $connection->error);  
}
```

It binds the `password` and `numE` parameters to the prepared statement, where `password` is a string (`s`) and `numE` is an integer (`i`).

```
$stmt->bind_param("si", $password, $numE);
```

The prepared statement is executed, and the statement is closed after execution.

```
$stmt->execute();  
$stmt->close();
```

The function queries the output parameters using a `SELECT` statement that retrieves the values of `@nombreCompleto` and `@tipoEmpleado`.

```
$result = $connection->query("SELECT @nombreCompleto AS nombreCompleto,  
@tipoEmpleado AS tipoEmpleado");
```

If the result set contains rows, the function returns the result. Otherwise, it returns the string 'wrong', indicating that the authentication failed.

```
if ($result && $result->num_rows > 0) {  
    return $result;  
} else {  
    return 'wrong';  
}
```

4.8.3 log out

```
1  <?php  
2  session_start();  
3  session_destroy();  
4  header('Location: ../login.php');  
5  ?>
```

This PHP code logs out the current user and redirects them to the login page. It is used to implement the "Log Out" functionality found in every employee interface, removing all session information for the user and returning them to the login screen.

4.8.4 Single-Page Application

a web application that loads a single HTML page and updates its content dynamically, without requiring full page reloads

```
function showContent(sectionId) {
    var sections = document.querySelectorAll('.section');
    sections.forEach(function(section) {
        section.classList.remove('active');
    });

    var selectedSection = document.getElementById(sectionId);
    selectedSection.classList.add('active');
}
```

This **showContent** function displays the content of a specific section on a web page and hides the rest. In general, it finds all sections with the **.section** class and removes the active class from them (which makes them invisible or inactive). It then selects the section indicated by the **sectionId** and adds the active class to it so that it is displayed. It is useful in interfaces where the user can navigate between several sections without reloading the page, as it allows content to be dynamically shown or hidden based on the selection.

All this with the idea that the user does not have to go from page to page, but rather do their work right there.

4.8.4 Example with Operator Interface

MEI

Bienvenido OPERADOR

Generar Reporte de Incidencia

Equipos Registrados

Cerrar Sesión

Generar Reporte de Incidencia

Add a new incident

Fill all form fields

Description

Problem

Priority

Equipment

Create a New Incident

```
<!DOCTYPE html>
<html lang="es">
<head>
```



```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>NTERFAZ OPERADOR</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <div class="sidebar">
      <h2>MEI</h2>
      <p>Bienvenido OPERADOR</p>
      <button onclick="showContent('reporte')">Generar Reporte de Incidencia</button>
      <button onclick="showContent('equipos')">Equipos Registrados</button>
      <button onclick="showContent('cerrar')">Cerrar Sesión</button>
    </div>

    <div class="content">
      <div id="reporte" class="section active">
        <h3>Generar Reporte de Incidencia</h3>
        <section>
          <h2>Add a new incident</h2>
          <div>
            <form action="addIncident.php" method="post">
              <fieldset>
                <legend>Fill all form fields</legend>
                <div>
                  <label for="description">Description</label>
                  <input type="text" id="description" name="description">
                </div>
                <div>
                  <label for="problem">Problem</label>
                  <select id="problem" name="problem" required>
                    <option value="ELE">Electric</option>
                    <option value="MEC">Mechanical</option>
                    <option value="MTO">Maintenance</option>
                  </select>
                </div>
                <div>
                  <label for="priority">Priority</label>
                  <select id="priority" name="priority" required>
                    <option value="ALT">High</option>
                    <option value="MED">Medium</option>
                    <option value="BAJ">Low</option>
                  </select>
                </div>
                <div>
                  <label for="equipment">Equipment</label>

```

```

        <input type="text" id="equipment" name="equipment" required>
      </div>
      <div>
        <button type="submit">Create a New Incident</button>
      </div>
    </fieldset>
  </form>
</div>
</section>
</div>
<div id="equipos" class="section">
  <h3>Equipos Registrados</h3>
</div>

<div id="cerrar" class="section">
  <h3>Cerrar Sesión</h3>
</div>
</div>
</div>
<script src="script.js"></script>
</body>
</html>

```

This HTML code creates a user interface for the operator in the MEI system, designed to manage incidents and query registered equipment. The page has a sidebar with three options: “Generate Incident Report”, “Registered Equipment” and “Log Out”, which when clicked display the corresponding section in the main content area.

The “Generate Incident Report” section includes a form to register new incidents, where the operator can enter the description, select the type of problem (electrical, mechanical or maintenance), the priority (high, medium or low) and the affected equipment. This form sends the information to addIncident.php using the POST method. The interface is dynamic, using JavaScript (script.js) to switch between sections without reloading the page, maintaining a fluid user experience.

MEI

BIENVENIDO TECNICO

Reporte de incidencia

Reporte de Reparacion

Reporte de Mantenimiento

Genera una Solicitud de Materiales

Cerrar Sesión

Reporte de Mantenimiento

— Fill all form fields —

Description

Maintenance Type

Incident

[Open a new maintenance](#)

4.8.5 Create New Incident

```
<?php
require "connection.php";
$db = connection();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST['description']) && isset($_POST['problem']) && isset($_POST['priority']) &&
        isset($_POST['equipment'])) {
        $description = $_POST["description"];
        $problem = $_POST["problem"];
        $priority = $_POST["priority"];
        $equipment = $_POST["equipment"];

        $query = "CALL insertarIncidencias('$description', '$problem', '$priority', '$equipment')";

        $response = mysqli_query($db, $query);
        if ($response) {
            echo "<script>alert('Incident added successfully.');

```

```
}
?>
```

This PHP code processes information submitted from a form to create a new incident in the database. First, it includes a connection file that establishes the connection to the database. Then, it checks if the request is a POST method, indicating that the form is being submitted. Next, it validates that all the required fields such as description, issue, priority, and team have been submitted. If any are missing, it displays an error message. If all fields are present, it creates a query to call a stored procedure with the incident data and executes it. If the query executes successfully, it displays a success message and redirects the user back to the operator page. If there is an error in the execution, it displays an error message that includes the reason for the failure. There is also an error message if the form is not submitted successfully. In summary, this script allows a new incident to be logged in the database and provides feedback to the user on the success or failure of the operation.

4.8.6 add a maintenance

```
<?php
include "../includes/headerO.php";

require "connection.php";
$db = connection();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST['description']) && isset($_POST['type']) && isset($_POST['incident'])) {
        $description = $_POST["description"];
        $type = $_POST["type"];
        $incident = $_POST["incident"];

        $query = "CALL insertarMantenimiento('$description', '$type', '$incident')";

        $response = mysqli_query($db, $query);
        if ($response) {
            echo "<script>alert('Maintenance added successfully.');

```

This PHP code is responsible for processing a form to add a new maintenance to the database. First, it includes a header file (headerO.php) that probably contains the structure and common elements of the page. Then, it establishes a connection to the database through a connection file. Next, it checks if the request was sent using the POST method, which indicates that the form has been submitted. It validates that all the necessary fields have been submitted: description, type, and incident. If any of these fields are missing, it displays a corresponding error message. If all the fields are present, a query is built to call a stored procedure that inserts the maintenance information into the database and is executed. If the query is executed successfully, an alert message is displayed confirming that the maintenance was successfully added and the user is redirected to the technician page. If there is an error in the execution of the query, an error message is displayed with the reason for the failure. An error message is also included if the form is not submitted successfully. In short, this script allows to record a new maintenance in the database and provides feedback to the user on the success or failure of the operation.

4.8.7 add Report

```
1  <?php
2  require "connection.php";
3  $db = connection();
4
5  if ($_SERVER["REQUEST_METHOD"] == "POST") {
6      if (isset($_POST['report']) && isset($_POST['type']) && isset($_POST['manager'])) {
7          $report = $_POST["report"];
8          $type = $_POST["type"];
9          $manager = $_POST["manager"];
10
11          $query = "CALL insertarReporte('$report', '$type', '$manager')";
12
13          $response = mysqli_query($db, $query);
14          if ($response) {
15              echo "<script>alert('Report added successfully.');
```