DEVELOPER’S MANUAL

INDEX

1. INTRODUCTION

1.1 Purpose

The purpose of this manual is to provide a comprehensive guide for setting up, developing, and maintaining the JobMaster project. JobMaster is a platform designed to streamline employee management processes for small and medium-sized enterprises (SMEs). This manual aims to ensure that developers working on JobMaster have a clear, consistent, and efficient reference to guide them through the various stages of the project's lifecycle.

By providing detailed instructions on environment setup, code repository management, and development processes, this manual helps maintain high standards and best practices throughout the development process. It also includes guidelines for future enhancements to keep the project scalable, secure, and up-to-date with evolving technologies and user needs.

By adhering to the instructions and guidelines in this manual, developers can contribute to the successful development and maintenance of JobMaster, ultimately providing a robust, user-friendly, and efficient solution for employee management in SMEs.

1.2 Audience

This manual is intended for developers who will be working on the JobMaster project. It is designed to cater to both new and experienced developers, providing them with the necessary knowledge and tools to efficiently contribute to the project.

The primary audience includes:

Backend Developers: Developers focusing on the server-side logic, database interactions, and API development using Laravel.
Frontend Developers: Developers responsible for implementing the user interface and client-side logic using Angular and Tailwind CSS.
Full Stack Developers: Developers who work on both the frontend and backend aspects of the project.
DevOps Engineers: Professionals who manage the deployment, integration, and operational aspects of the project, ensuring smooth and reliable delivery of the application.
Project Managers and Team Leads: Individuals overseeing the development process, ensuring that the project adheres to timelines, quality standards, and best practices.

This manual aims to provide each of these roles with clear and concise guidelines on setting up their development environment, managing code repositories, following coding standards,

integrating APIs, and troubleshooting common issues. By doing so, it ensures that all team members are aligned and can collaborate effectively, leading to a more streamlined and efficient development process.

1.3 Scope

This manual covers all aspects of the JobMaster project from initial setup to future enhancements. It is designed to be a complete reference for developers, encompassing the following areas:

Environment Setup: Detailed instructions on installing and configuring the necessary tools and software to create a local development environment. This includes setting up Git, Composer, PHP, MySQL, Laravel, Angular CLI, and Tailwind CSS.

Development Processes: Best practices for frontend and backend development, including coding standards, API integration, and using Tailwind CSS for styling. This section also covers database setup, running the project locally, and testing the full-stack application.

Future Enhancements: Recommendations for future improvements to the project, such as scalability enhancements, performance optimizations, security measures, new features.

By covering these areas comprehensively, this manual ensures that all developers have the information they need to effectively contribute to the JobMaster project, maintain high-quality standards, and plan for future growth and enhancements.

2. Project Overview

2.1 Project Description

JobMaster is an employee management platform specifically designed to meet the needs of small and medium-sized enterprises (SMEs). It addresses the prevalent reliance on manual processes for managing various aspects of employee operations such as attendance tracking, scheduling, and document handling. These manual processes often lead to errors, inefficiencies, and increased operational costs.

JobMaster aims to streamline these administrative tasks by providing a comprehensive, efficient, and user-friendly online platform. The platform offers features such as creating and managing employee profiles, establishing work schedules, generating detailed attendance and performance reports, and integrating with existing payroll systems. By automating these processes, JobMaster helps SMEs save time, reduce errors, and improve overall workforce management.

Key features include:

- Administrator Capabilities:
    - Creating and managing employee profiles.
    - Establishing work schedules, including shifts, days off, and vacations.
    - Integrating with payroll systems to streamline payment processes.

- Employee Capabilities:
    - accessing personal profiles to view contact details and communication preferences.
    - Recording working hours including clock-in and clock-out.
    - Requesting days off through the platform
    - Access to personal documents.

3. Development Environment Setup

3.1 Tools and Software Required

Setting up the development environment for JobMaster involves installing and configuring various tools and software necessary for both backend and frontend development. Below is a list of required tools and software, along with a brief description and installation instructions:

1. **XAMPP:**

- **Download**: XAMPP
- **Purpose**: Provides a local server environment with Apache, MySQL (or MariaDB), and PHP.

Installation on Windows

1. Go to the XAMPP download page.
2. Download the XAMPP installer for Windows.
3. Run the installer and follow the on-screen instructions to complete the installation.
4. Open the XAMPP Control Panel and start the Apache and MySQL services.

Installation on Ubuntu:

1. Open a terminal.
2. Download the XAMPP installer:

```
wget https://www.apachefriends.org/xampp-files/7.4.24/xampp-linux-x64-7.4.24-0-installer.run
```

3. Make the installer executable:

```
chmod +x xampp-linux-x64-7.4.24-0-installer.run
```

4. Run the installer:

```
sudo ./xampp-linux-x64-7.4.24-0-installer.run
```

5. Follow the on-screen instructions to complete the installation.
6. Start XAMPP

```
sudo /opt/lampp/lampp start
```

2. **Node.js and npm:**

Download: Node.js
Purpose: Node.js is required to run Angular CLI and other JavaScript tools. npm (Node Package Manager) comes with Node.js and is used to manage JavaScript packages.

Installation on Windows:

1. Go to the Node.js download page.
2. Download the Windows installer for the LTS (Long Term Support) version.
3. Run the installer and follow the on-screen instructions to complete the installation.
4. Verify the installation by opening a command prompt and running:

```
node -v
npm -v
```

Installation on Ubuntu:

1. Open a terminal.
2. Update the package index:

```
sudo apt update
```

3. Install Node.js and npm:

```
sudo apt install nodejs npm
```

4. Verify the installation by running:

```
node -v
npm -v
```

3. Composer:

Download: Composer
Purpose: A dependency manager for PHP, used to install and manage Laravel packages.

Installation on Windows:

1. Go to the Composer download page.
2. Download the Composer-Setup.exe installer.
3. Run the installer and follow the on-screen instructions to complete the installation.
4. Verify the installation by opening a command prompt and running:

```
composer -V
```

Installation on Ubuntu:

1. Open a terminal.
2. Download the Composer installer:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
```

3. Verify the installer SHA-384 hash:

```
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'YOUR_EXPECTED_HASH') { echo 'Installer verified'; } else { echo 'Installer
corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
```

Replace YOUR_EXPECTED_HASH with the latest hash from the Composer Public Keys / Signatures page.

4. Install Composer globally:

```
sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

5. Verify the installation by running:

```
composer -V
```

4.Laravel Installer:

Installation Command: composer global require laravel/installer
Purpose: Facilitates the creation and management of Laravel projects.

Installation on Windows:

1. Open a command prompt.
2. Run the installation command:

```
composer global require laravel/installer
```

3. Add Composer's global bin directory to your system's PATH environment variable to run the laravel command globally. You can find the global bin directory by running composer global config bin-dir --absolute in the command prompt. Then, add this path to your PATH environment variable through the system settings.
4. Verify the installation by running:

```
laravel --version
```

Installation on Ubuntu:

1. Open a terminal.
2. Run the installation command:

```
composer global require laravel/installer
```

3. Add Composer's global bin directory to your system's PATH environment variable by adding the following line to your ~/.bashrc or ~/.zshrc file:

```
export PATH="$HOME/.composer/vendor/bin:$PATH"
```

After adding the line, reload your shell configuration:

```
source ~/.bashrc
 or
source ~/.zshrc
```

4. Verify the installation by running:

```
laravel --version
```

5.Angular CLI:

Installation command:

```
npm install -g @angular/cli
```

Purpose: A command-line interface for Angular, helps in creating, managing, and building Angular projects.

Installation on windows & Ubuntu

1. Open a command prompt.
2. Run the installation command:

```
npm install -g @angular/cli
```

3. Verify the installation by running:

```
ng --version
```

6.Install Git:

Download and Install Git:

1. Windows: Download and install from Git for Windows.
2. Ubuntu: Install via terminal with the command sudo apt-get install git.

Set Up Your GitHub Account:

1. Create a GitHub account if you don't have one already.
2. Create a repository for your project on GitHub
3. The JobMaster repositories are public, so users can clone them directly without needing to set up a personal repository.

Clone the Repository to Your Local Machine:

1. **Use the command:**

```
git clone repository_url
```

Optional: Use GitHub Desktop for Easy Pull and Push:

1. Download and install GitHub Desktop.
2. Use GitHub Desktop to clone repositories, and easily manage pull and push operations through a graphical interface.

7.Setting Up the Projects:

- **Laravel:**
    1. Navigate to your Laravel project directory.
    2. Install dependencies using composer install.
    3. Ensure you have Node.js and npm installed on your machine. Install the necessary dependencies by running:

       ```
       npm install
       ```

       This will install all the dependencies listed in your package.json file.
    4. Run Command: Execute the following command to start the asset compilation process in development mode:

       ```
       npm run dev
       ```

    5. Watch Mode: When running npm run dev, Webpack enters "watch" mode, monitoring your asset files for changes and recompiling them as needed. This is particularly useful during active development, as it ensures that your latest changes are always reflected without the need to manually recompile.
    6. Run the Laravel development server using **php artisan serve**.

- **Angular:**
    1. Navigate to your Angular project directory.
    2. Install dependencies using npm install.
    3. Start the development server using **ng serve**.

8.Connect Laravel to Angular:

1. Set up API endpoints in Laravel to communicate with the Angular frontend.
2. Use Angular services to make HTTP requests to the Laravel API.

9.Integrating Tailwind CSS

Tailwind CSS has already been integrated into the JobMaster project. Below are the steps that were followed to integrate Tailwind CSS into the Angular project:

1. **Install Tailwind CSS via npm:**

```
npm install tailwindcss
```

2. **Initialize Tailwind CSS Configuration:**

```
npx tailwindcss init
```

3. **Configure Tailwind to Remove Unused Styles in Production:**
   - Edit the tailwind.config.js file to include the paths to all of your template files.

```
module.exports = {
  purge: ['./src/**/*.{html,ts}'],
  darkMode: false, // or 'media' or 'class'
  theme: {
    extend: {},
  },
  variants: {
    extend: {},
  },
  plugins: [],
}
```

4. **Include Tailwind in Your CSS:**

```
@import 'tailwindcss/base';
@import 'tailwindcss/components';
@import 'tailwindcss/utilities';
```

5. **Ensure Angular Uses the Updated Styles:**
   - Make sure that the Angular project is configured to use the styles file where Tailwind is imported. This is typically done in angular.json under the styles array.

CREDENTIALS

ADMINISTRATOR:

NIF:
admin

PASSWORD:
admin

JOB MASTER

NIF

admin

Contraseña

admin Ocultar

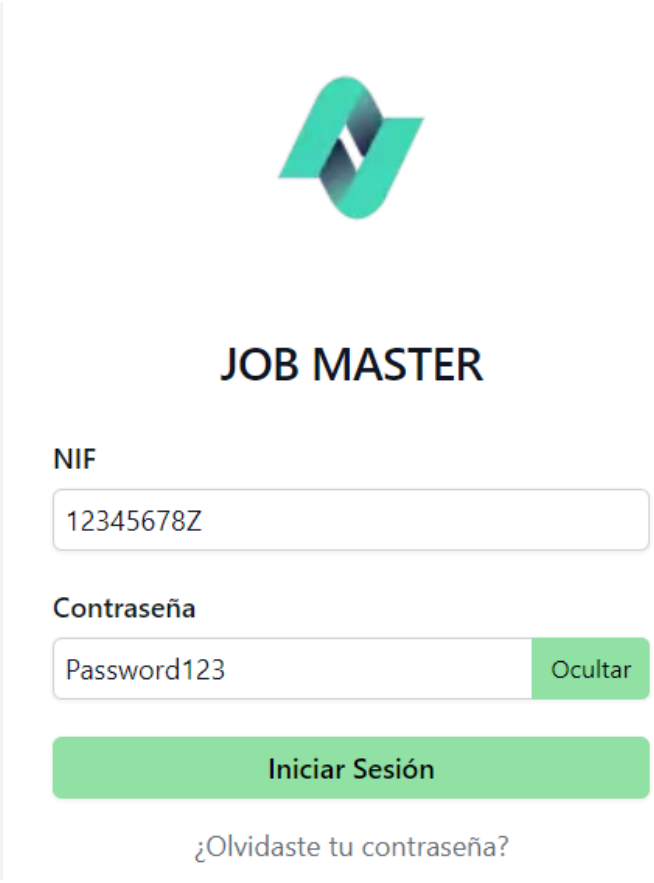
Iniciar Sesión

¿Olvidaste tu contraseña?

USER/EMPLOYEE:

NIF:
12345678Z

Contraseña:
Password123



4. Future Development and Scalability

4.1 Feature: Clock In and Clock Out System for attendance (Fichaje) in employees

Overview:
The clock in and clock out system allows employee users to record their daily attendance by clocking in at the start and clocking out at the end of their workday. This feature helps in tracking work hours accurately and ensures proper attendance management.

Current Functionality:

- Clocking In:
  - The employee logs into the system.
  - The employee navigates to the time clock area.
  - The employee selects the option to clock in.
  - The system records the start time when the clock in button is pressed.

- Clocking Out:
  - When the employee finishes their shift, they press the clock out button.
  - The system records the total time worked and the date.

Enhancements:

1. Geolocation Restriction:
   - Implement geolocation to ensure that the employee can only clock in if they are within a specified distance from the workplace. If the employee is too far, the system will prevent clocking in and display a message.
2. Late Arrival Note:
   - Allow employees to write a note if they arrive late. The system will record the late clock-in time and send the note to the admin for review.
3. Missed Clock-In Notification:
   - If an employee forgets to clock in or out, they can send a notification to the admin explaining the situation. The admin can then manually adjust the attendance record.
4. Worked Hours Table:

- The system will automatically calculate and display the total hours worked once the employee has clocked out. This will be presented in a table format, summarizing daily, weekly, and monthly totals.

5. Extra Hours Tracking:
   - The system will identify and record any extra hours worked beyond the regular shift. This information will be highlighted separately.
6. Admin Access and Modification:
   - Admin will have access to all clock-in and clock-out details. They can view the total hours worked, extra hours, and any notes provided by employees. Admins will also have the capability to modify attendance records if necessary.

.

Implementation Steps:

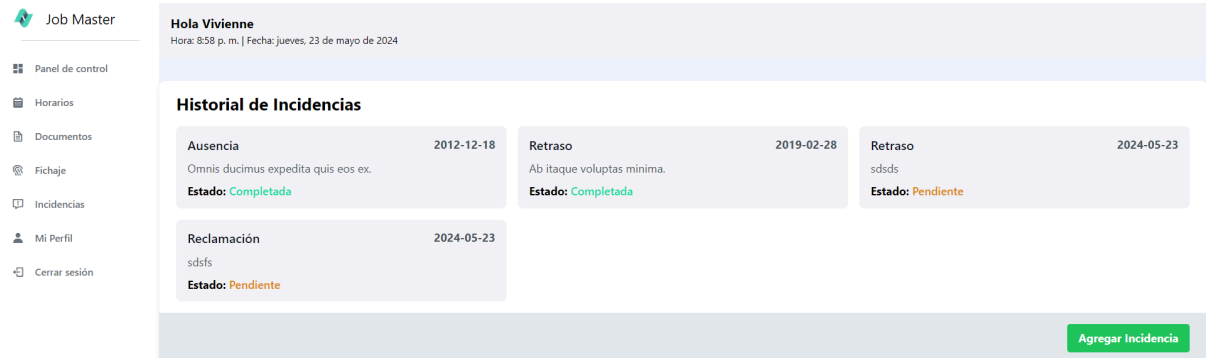
1. **Geolocation Integration:**
   - Use geolocation APIs to track the employee's location at the time of clocking in.
   - Set up location boundaries to allow clocking in only within the designated area.
2. **Late Arrival and Missed Clock-In Notes:**
   - Add a feature for employees to submit notes explaining late arrivals or missed clock-ins.
   - Ensure these notes are sent to the admin dashboard for review and action.

3. **Automated Hour Calculation:**
   - Implement logic to calculate total worked hours and extra hours.
   - Update the user interface to display these calculations in a table format.

By incorporating these enhancements, the clock in and clock out system will become more robust, ensuring accurate attendance tracking and providing additional functionalities to support both employees and administrators in managing work hours efficiently.

4.2 Feature: Enhanced Incident Reporting System

Employee Side:



Admin side:

Overview:
The Incident Reporting feature allows employees to add new incidents by providing details such as incident type, description, and any relevant information. Employees must be logged into the system to access the incident reporting feature.

Future Enhancements:

1. **Document Upload Capability:**
    - Employees will be able to upload documents related to the incident. This could include any supporting documents, such as photos, reports, or requests for changes in personal documents.

2. **Functionality Enhancements:**
    - **Document Attachment:**
        - When creating an incident, the system will include an option to attach files.
        - The form will have an additional field for uploading documents: Attach Document [Adjuntar Documento] (Optional field, allows file upload).
    - Employees can upload relevant documents when reporting an incident.
    - Viewing Attached Documents:
        - The incident list will display icons or links to view attached documents for each incident.
    - Administrative Review:
        - Admins will have the ability to view and download attached documents for further review.

Implementation Steps:

- **Document Upload Feature:**
  - Integrate file upload functionality in the incident reporting form.
  - Modify the incident creation form to include a file input field for document attachment.
  - Ensure uploaded documents are stored securely and linked to the respective incident records in the database.
- **Enhancing Incident Reporting UI:**
  - Update the incident list UI to display links or icons for attached documents.
  - Implement a preview feature for admins to quickly view attached documents without downloading them.
- **Backend Changes:**
  - Extend the incident database schema to handle file metadata and storage paths.
  - Update API endpoints to support file uploads and retrievals

4.3 Feature: Admin Configuration Section

Overview:

Administrators manage employee profiles, schedules, incidents, and other administrative tasks, but there is no existing feature for customizing company branding within the application.

Enhancement:
Introduce a configuration section in the sidebar for administrators, enabling them to customize the company's branding by changing the company name and logo.

Configuration Section Features:

1. **Accessing Configuration:**
   - A new "Configuration" section will be added to the admin sidebar.
   - Only users with administrative privileges will have access to this section**.**
2. **Changing Company Name:**
   - **Field:** An input field to update the company name.
   - **Validation:** Ensure the company name is not empty and follows any predefined format rules.
   - **Save:** A button to save changes, updating the company name across the application.
3. **Changing Company Logo:**
   - **Field:** A file upload input to change the company logo.
   - **Validation:** Validate the file type (e.g., PNG, JPEG) and size to ensure it meets the requirements.
   - **Preview:** Display a preview of the uploaded logo before saving.
   - **Save:** A button to save changes, updating the logo displayed throughout the application.

By adding this configuration section, administrators will have the flexibility to customize the application's branding to align with their company’s identity, enhancing the professional appearance and relevance of the JobMaster platform.

4.4 Feature: Alert Messages for Content Changes

Enhancement:
Implement alert messages that appear whenever new content is added or existing content is edited. These alerts will provide immediate feedback to users about the success of their actions.

New Alert Message Features:

1. **Alert for Adding Content:**
    - When a user adds new content (e.g., employee profiles, incidents, schedules), an alert message will appear confirming the successful addition.
    - Example Message: "New [content type] added successfully."

2. **Alert for Editing Content:**
    - When a user edits existing content, an alert message will appear confirming the successful update.
    - Example Message: "[Content type] updated successfully."

5. Conclusions

The JobMaster Developer Manual provides a comprehensive guide for setting up, developing, and maintaining the JobMaster project. This manual is designed to ensure that developers have the necessary tools, knowledge, and best practices to work efficiently and effectively on the project. Here are the key takeaways:

1. **Environment Setup:**
    - Detailed instructions for setting up the development environment, including the installation of essential tools like Git, Node.js, npm, Composer, and XAMPP.
    - Step-by-step guidance on cloning the JobMaster repositories and configuring both the Laravel and Angular projects.

2. **Project Configuration:**
    - Instructions for setting up API endpoints and integrating the backend (Laravel) with the frontend (Angular).
    - Guidelines for using Tailwind CSS for styling the Angular application.
3. **Future Enhancements:**
    - Proposals for new features and improvements to enhance the functionality and scalability of JobMaster.
    - Detailed plans for implementing a clock in and clock out system with geolocation and additional features for managing attendance.

- Enhancements to the incident reporting system to include document uploads and more robust management capabilities.
- Introduction of a configuration section for administrators to customize the company name and logo.
- Implementation of alert messages for content changes to provide immediate feedback to users.

4. **Best Practices:**
    - Emphasis on maintaining consistency, efficiency, and high standards throughout the development process.
    - Encouragement of best practices for coding, testing, and deployment to ensure the project remains maintainable and scalable.

By following the guidelines and instructions outlined in this manual, developers will be well-equipped to contribute to the JobMaster project, ensuring its continued success and evolution. The manual not only serves as a technical reference but also fosters a collaborative and productive development environment, paving the way for future growth and enhancements.