UNIVERSITY OF BEIRA INTERIOR

DEPARTMENT OF INFORMATICS

Web Programming

Electricity Consumption and Generation Management Software

Author
Taha Yunus Demir – a55403

Supervisor

Prof. Dr. Nuno José Matos Pereira

Covilhã, June 2025

Contents

Final Project Report

Introduction

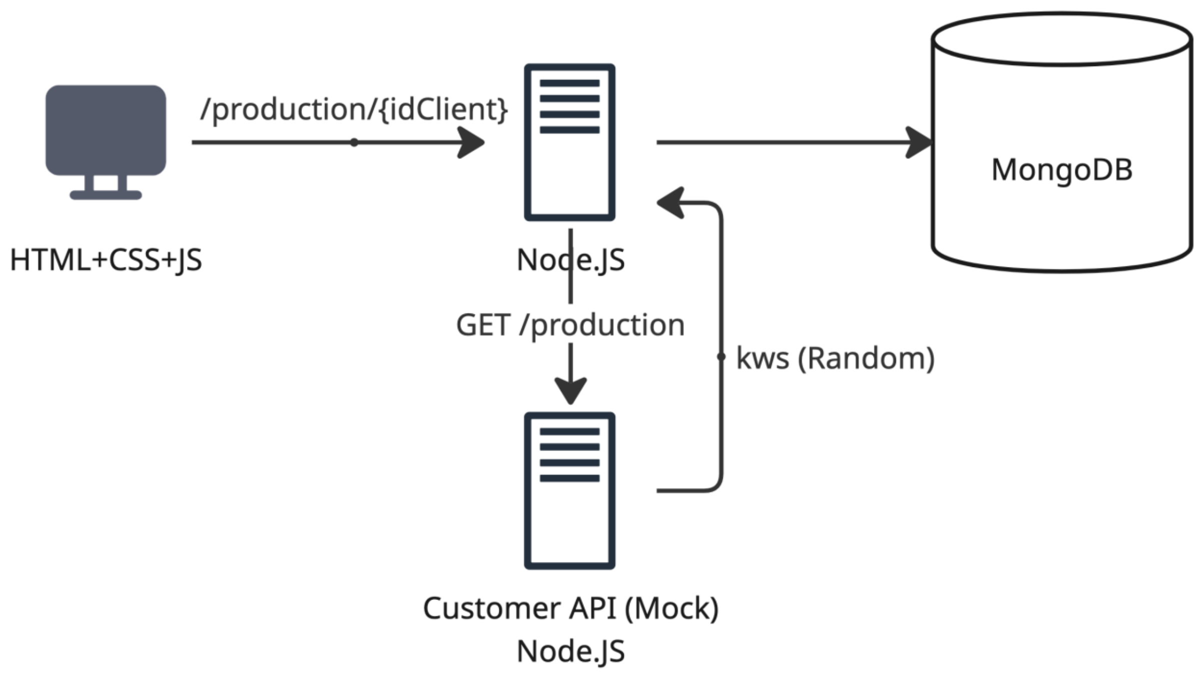
This document provides a comprehensive overview of the "Electricity Consumption and Generation Management Software" project. The primary goal of this application is to provide a robust platform for managing renewable energy generation from solar panels. It serves different stakeholders through a role-based access control system, allowing customers to register their installations, technicians to validate them, and operations managers to monitor energy production and manage energy credits.

The system is designed to be a complete solution, covering the entire lifecycle from installation registration to energy credit accounting. It is built with modern web technologies, ensuring scalability, maintainability, and a seamless user experience.

Implementation

The application is architected as a full-stack solution with a separate mock service to simulate real-world data, ensuring a decoupled and testable system.

System Architecture



The architecture consists of three main components: a frontend application, a backend server, and a mock API for energy production data.

- **Frontend:** A single-page application built with **React** and **Material-UI** for a modern and responsive user interface. It communicates with the backend via a RESTful API.

- **Backend:** A **Node.js** server using the **Express** framework. It handles business logic, authentication, and data persistence.

- **Database: MongoDB** is used as the database, providing a flexible schema for storing user credentials, installation details, and energy production logs.

- **Mock Customer API:** A separate **Node.js** service that simulates an external API from a customer's installed system. It provides random kilowatt (kW) production data upon request.

Key Features

The implementation was divided into three sprints, each focusing on a core set of features.

1. Authentication and Authorization (Sprint 1)

A secure authentication system was implemented using **JSON Web Tokens (JWT)**.

- **Role-Based Access:** The application supports three user roles: `Client`, `Technician`, and `Operations Manager`.

- **Login:** Users are authenticated against credentials stored in MongoDB. Passwords are securely hashed using `bcryptjs`.

- **Authorization:** Upon successful login, a JWT containing the user's ID and role is issued. This token is used to protect routes on both the frontend and backend, ensuring users can only access features permitted for their role. A database seeding script (`seed.js`) was created to pre-populate the system with initial user data.

2. Solar Panel Installation & Certificate Management (Sprint 2)

This feature allows clients and technicians to manage the installation lifecycle.

- **Client-Side Registration:** Clients can fill out a form to register their solar panel installations, providing technical specifications and location data.

- **Technician-Side Validation:** Technicians have an interface to view all installation requests. They can search for specific installations, review the details, and validate them.

- **Certificate Upload:** After validation, a certified technician can upload an energy producer certificate (a `.pdf` file) for the installation. The backend uses `multer` to handle file uploads, and the certificate is stored and linked to the corresponding installation record.

3. Energy Monitoring & Credit Accounting (Sprint 3)

This feature is exclusive to the 'Operations Manager' role and focuses on monitoring and financial accounting.

- **Production Monitoring:** The manager can select a client and trigger a request to fetch their latest energy production data. The backend calls the **Mock Customer API**, receives a random kW value, and stores it in the database, timestamped and associated with the client.

- **Credit Accounting:** The manager can trigger a monthly credit calculation for a client. The system calculates the total energy produced over the last month, compares it against a placeholder for consumption, and converts any surplus energy into credits.

- **Notifications:** The user's credit balance is updated, and a simulated email notification is logged to the console to inform the customer of their newly generated credits.

Conclusion

The project successfully delivers a functional proof-of-concept for an electricity management platform. The core requirements for authentication, installation management, energy monitoring, and credit accounting have been implemented, providing a solid foundation for a real-world application.

Limitations

- **Mock Data:** The energy production data is randomly generated by a mock API. For a production system, this would need to be replaced with an integration to a real hardware monitoring service.

- **Simplified Credit Calculation:** The energy credit calculation uses a static placeholder for energy consumption. A real system would need to integrate with utility data to get actual consumption figures.

- **Simulated Notifications:** The email notification system is currently a simulation that logs to the console. A full implementation would require integrating a third-party email service (e.g., SendGrid, Mailgun).

- **No Admin Panel:** The application lacks a dedicated administrative interface for managing users, roles, and system settings. Initial data is managed via a seed script.

Potential Improvements and Future Work

- **Real-time Data Integration:** Replace the mock service with a real API from solar panel inverters or smart meters.

- **Advanced Dashboards:** Develop comprehensive dashboards with charts and analytics for energy production, consumption, and credit history.

- **Full-Fledged Notification System:** Implement a complete notification service for emails, SMS, or in-app alerts.

- **Mobile Application:** Create a companion mobile app for customers to monitor their energy production and credits on the go.

- **Billing Integration:** Integrate with a billing system to automatically apply energy credits to customer invoices.

- **Enhanced Security:** Implement further security measures such as two-factor authentication (2FA) and detailed audit logs.

- **Automated Jobs:** Use a cron job or scheduler (like node-cron) to automate the monthly credit calculation process instead of relying on a manual trigger.