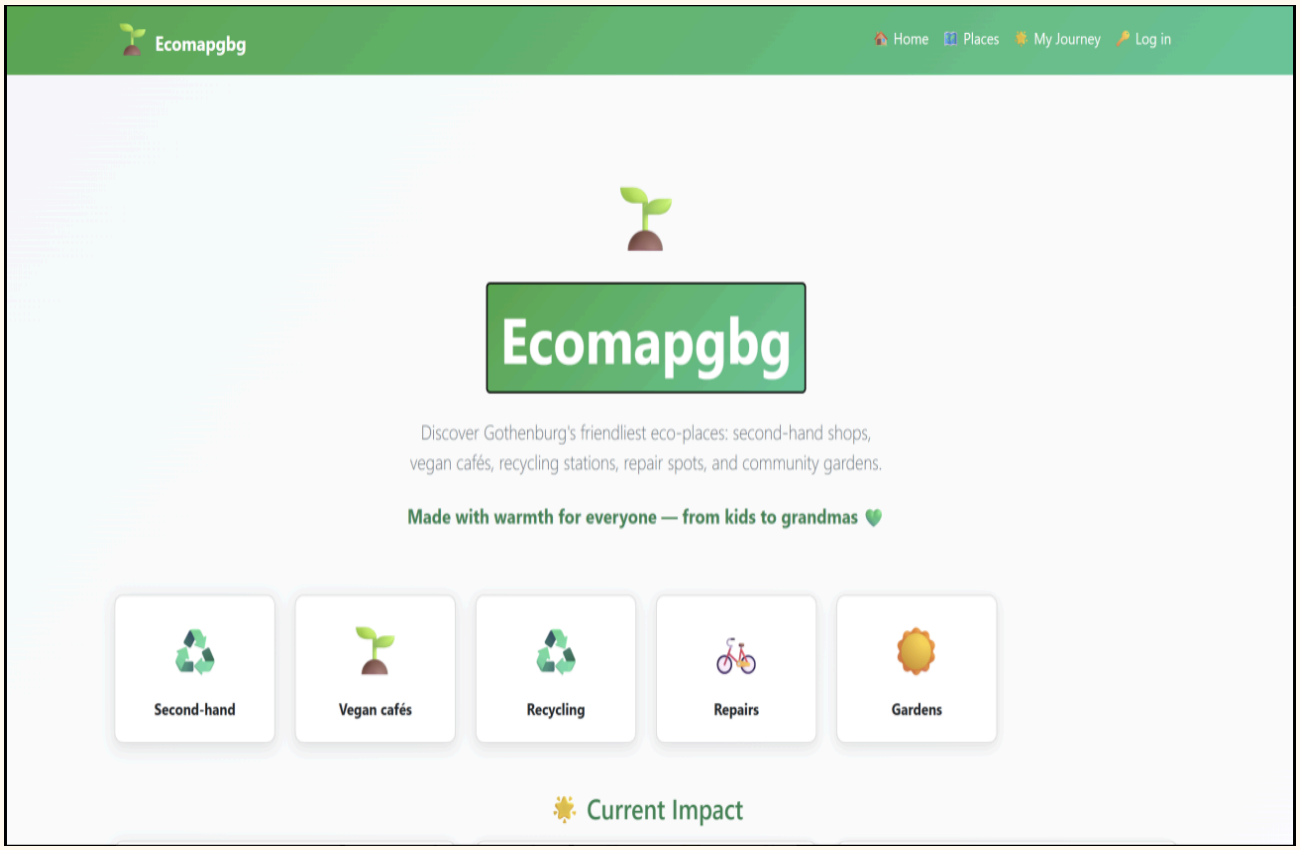
Software Design Description

Av Nor & Yotaka

https://github.com/Bombalaka/ecomapgbg



MI10 AI Appathon

Software Design Document for EcoMapGbg

📑 Table of Contents

1. Introduction pg.3
2. Architecture Overview pg.3
3. Main Application Features pg.4
4. Data Management and Persistence pg.4
5. Backend and API pg.5
6. Frontend pg.5
7. Testing Strategy pg.6
8. CI/CD and Deployment pg.6
9. Security Considerations pg.6
10. Future Plans and Feature Ideas pg.7
11. Documentation and Repositories pg.7
12. Contributors pg.7
13. License pg.7

1. Introduktion

EcoMapGbg is a web-based platform developed to support sustainable living in the city of Gothenburg. The core purpose of the application is to connect people with local reuse hubs, second-hand shops, repair cafes, recycling centers, and other eco-friendly locations. By encouraging circular consumption, EcoMapGbg aims to reduce waste and help build a greener, more resilient city.

The application provides an intuitive map interface where users can find, explore, and even contribute eco-spots. It combines modern web technologies with cloud-ready architecture and an accessible, inspiring user experience.

2. Architecture Overview

The system uses a **single-project layered architecture**, which means that although the entire application resides within one Blazor Server project, the internal logic is clearly separated into well-defined responsibilities. This results in a maintainable, testable, and scalable codebase.

The core layers include:

- **Presentation Layer**: Built with Blazor components and Razor pages. Responsible for rendering the UI and handling user interactions.
- **Application Layer**: Contains business logic and service interfaces. Acts as the bridge between the UI and the data layer.
- **Data Layer**: Includes repository interfaces and concrete implementations that manage data access through MongoDB.
- **Domain Layer**: Defines the fundamental data models and entities, such as `Location`, `LocationType`, and related DTOs.

This architecture is inspired by Clean Architecture and DDD (Domain-Driven Design) principles.

3. Main Application Features

The following key features define the capabilities of the EcoMapGbg platform:

- **Interactive Map**: Users can browse an embedded map to find reuse and recycling related locations in Gothenburg.
- **Filtering Options**: Visitors can filter locations by type (e.g., second-hand, recycling), neighborhood, or opening hours.
- **Crowdsourced Submission**: A submission form allows users to suggest new locations, contributing to community knowledge.
- **Detailed Location Pages**: Each spot has its own page with details such as address, description, contact info, and open hours.
- **Dark Mode UI**: Designed with accessibility and modern aesthetics in mind, the application uses a dark-themed palette with contrasting accent colors.
  Docker-Enabled: The application supports containerized deployment for portability.

4. Data Management and Persistence

EcoMapGbg uses **MongoDB** as its primary database, a natural choice for document-based storage and flexibility. Each location is saved as a document with fields such as `Name`, `Coordinates`, `Category`, `Neighborhood`, and `OpeningHours`. This format allows for flexible querying and efficient filtering.

The `ILocationRepository` interface defines the available database operations such as retrieving all locations, fetching by ID, filtering, and adding new entries. The concrete implementation `LocationRepository` uses the MongoDB C# driver and is injected via dependency injection.

To prevent configuration leaks, the connection string is stored securely in `appsettings.json` or injected via environment variables in production.

---

5. Backend and API

An internal API is provided via `LocationsController.cs`, following REST principles. This API allows external systems (e.g., a mobile app or kiosk display) to consume the location data in a structured JSON format.

Endpoints include:

- `GET /api/locations` — returns all locations
- `GET /api/locations/{id}` — returns details of a specific location
- `POST /api/locations` — accepts new location submissions
- `GET /api/locations/filter` — filters by type, neighborhood, or time

All API calls return DTOs rather than full entities to preserve encapsulation and reduce payload size.

6. Frontend (Blazor Server UI)

The frontend leverages Blazor Server for real-time interaction without page reloads. Bootstrap is used for styling, with responsive layouts for mobile friendliness.

Each user-facing page is built as a component:

- `Home.razor`: Introduces the mission with an uplifting message and CTA. Consider including a background image of Gothenburg here (e.g., a calm street with bikes or a reused furniture market).
- `Map.razor`: Contains the interactive Leaflet map and filter controls.
- `AddLocation.razor`: Form for community-contributed locations.
- `LocationDetails.razor`: Individual location view with badges, categories, and contact info.

---

7. Testing Strategy

Unit testing is conducted using **xUnit**. The main test coverage includes:

- **Service Layer**: Filtering logic, business rules for submissions.
- **Repository Layer**: Verifying MongoDB operations (with mocks).
- **Controller Layer**: Ensuring correct responses and status codes.

Tests are automated through GitHub Actions and executed during pull requests to maintain reliability.

8. CI/CD and Deployment

Deployment is designed to be cloud-ready and containerized:

- **Docker Compose**: Starts both the app and MongoDB for local or production environments.
- **GitHub Actions**: CI pipeline for build, test, and future publishing steps.

The app can be deployed to:

- **Azure Web App + Cosmos DB**
- **AWS Elastic Beanstalk with MongoDB Atlas**

All environment settings (e.g., DB connection, API keys) are configurable via environment variables.

9. Security Considerations

Security practices include:

- Sanitized and validated inputs on all forms.
- Anti-forgery tokens (`[ValidateAntiForgeryToken]`) used in form submissions.
- MongoDB connection strings stored in secrets/environment.
- CORS policies restricted to trusted origins.

10. Future Plans and Feature Ideas

EcoMapGbg is designed with extensibility in mind. Planned or potential features include:

- Integration with **Västtrafik API** to suggest eco-friendly transport routes.
- **Event calendar** for local sustainability workshops.
- **Gamification**: award badges to users who submit or visit locations.
- **User accounts** and profile dashboards.
- **Location moderation** workflows for verifying new submissions.
- **AI-powered suggestions** for nearest reuse options based on habits.

---

11. Documentation and Repositories

- **GitHub Repository**: https://github.com/Bombalaka/ecomapgbg
- **Swagger/OpenAPI** available at `/swagger` in development mode.
- **README.md** includes full setup guide, licensing, and team info.

12. Contributors

- **Yotaka88** (@Yotaka88) – Lead Developer
- **Nor Ajami** (@NorAjami) – Lead Developer

Special thanks to the Gothenburg community for their inspiration.

13. License

Licensed under the MIT License. The application is open for reuse, modification, and contribution.

✨ "EcoMapGbg connects local hearts and habits – one reuse point at a time." ✨