**Пояснительная записка проекта DM**

**(Document Manager), Backend.**

Проект состоит из классической трёхслойной архитектуры (PL, BAL, DAL).

В качестве БД — Postgresql. В качестве ORM используется Entity Framework.

В основе архитектурного стиля лежит REST API.

Для документации API используется Swagger UI.

Зависимости (Dependency injection) определяются в IOC-контейнерах в классе Startup.

Реализована требуемая бизнеслогика (работа с пользователями, организациями, проектами, записями(задачами) и их шаблонами, комментариями к записям и работа с файлами).

Реализована кастомная аутентификация по ролям без использования Identity (класс AuthorizeAttribute) через JWT-токен (класс JwtMiddleware),

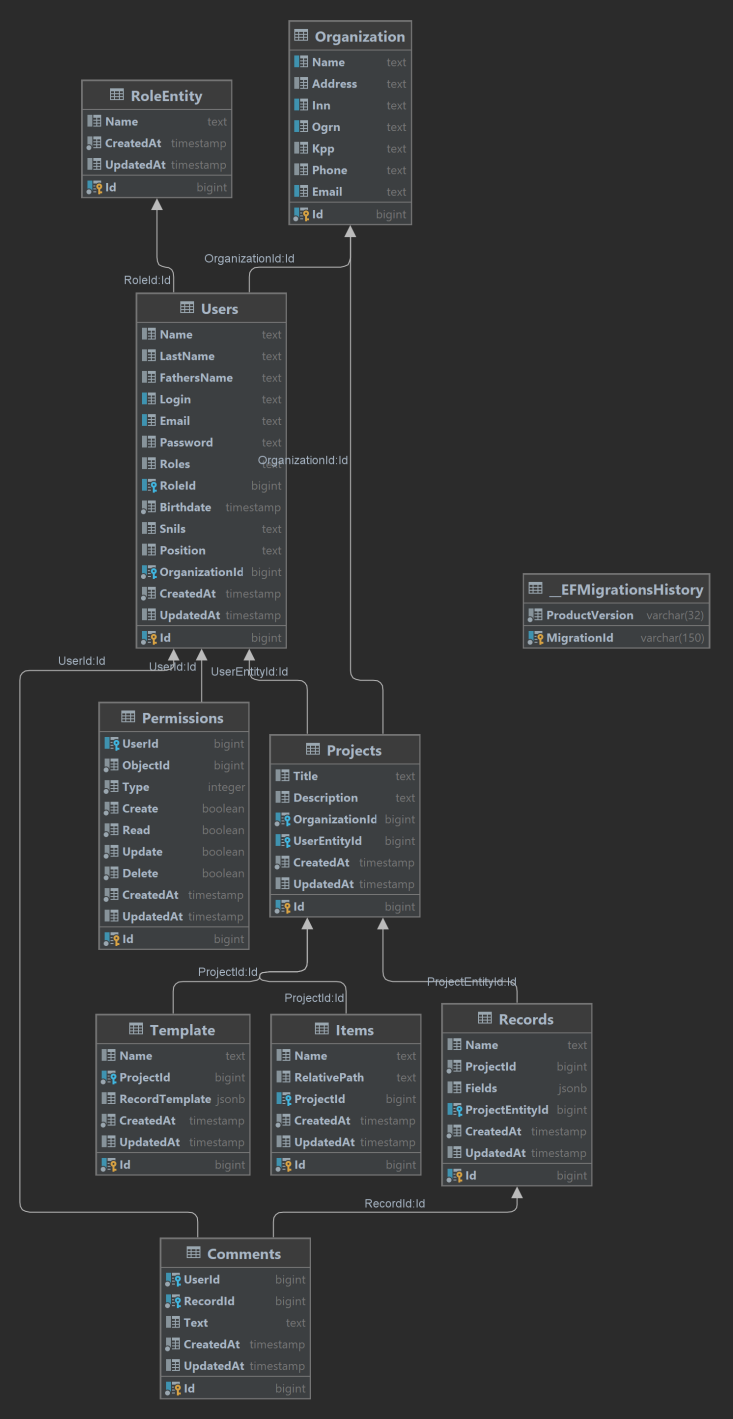
а также гибкая авторизация (flexible authorization) за счёт таблицы Permissions.

Cам токен генерируется в DM.Domain/UserService/UserHelper в методе GenerateJwtToken, алгоритм шифрования — hmacsha256.

Работа с паролем — в DM.Domain/PasswordHelper.

Работа с файлами лежит в классе ItemController, добавлена версионность и конвертация из ifc в wexbim.

Структура в Postgresql:

Таблица **Item** представляет собой информацию о загруженных на удалённый сервер файлах.

Сущность **Record** представляет собой записи к проектам, где поле Fields является динамическим и позволяет задавать любой формат «ключ-значение».

В таблице **Template** хранятся шаблоны для Record под определённый проект для определённой организации.

Как видно из схемы, головной сущностью является Organization. К ней привязываются пользователи и проекты. Без организации создать что-либо не получится.

Миграция из папки DM: dotnet ef database update --startup-project ../DM

**Wrapper (обёртка под проект Unity).**

ORM отсутствует, вместо неё ADO.NET для работы с SQLite.

Запросы реализованы с использованием RestSharp вместо HttpClient'а,

кэширование для GET запросов, запись запросов в SQLite для методов POST, DELETE.

Основные классы:

SqliteDatabaseContext — метод Seed отвечает за создание локальной БД на стороне клиента.

Controllers (Folder) — API.

RequestHelper ­— запись в SQLite

MemoryCache — кэширование

OfflineHelper — пинг ресурса (проверка наличия интернета).

**TODO:**

1. На данный момент необходимо протестировать работоспособность оффлайна обёртки (BackgroundTaskManager и кэширование запросов),

полностью рабочий пример можно увидеть в WrapperDM/UserController, остальное не протестировано.

Первичное тестирование для упрощения можно проводить в консольном проекте, финальное в проекте Unity.

1. Также нереализована запись фоновых задач для PUT запросов.

## **Программное обеспечение системы.**

В этом разделе перечислены основные фреймворки и библиотеки, применяемые при разработке серверной части.

* ASP.NET Core 5.0 — платформа для веб-разработки программного обеспечения на C#.
* ASP.NET Core — платформа для создания веб-приложений.
* EF Core — ORM фреймворк.
* PostgreSQL — реляционная СУБД.
* Xbim.Essentials/Xbim.Geometry — библиотеки для работы с данными в формате IFC.
* Automapper — библиотека для преобразования моделей.

Для тестов:

* Xunit, Moq.