

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

Отчет

Лабораторная работа № 2  
“REST, RESTful, SOAP, GraphQL”

Выполнил:

Коротин А.М.

К33392

Проверил:

Добряков Д. И.

Санкт-Петербург

2024 г.

## Задача

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript.

Вариант 4: Сервис для управления умным домом. Требуемый функционал: регистрация, авторизация, создание профиля, добавление нового устройства, настройка устройства, создание сценариев, запуск сценариев по триггерам/либо по времени (запуск сценариев можно симитировать, главное, чтобы о запуске сохранилась информация в логах).

## Ход работы

Для реализации приложения была выбрана многослойная архитектура с разделением на доменный слой, слой инфраструктуры (работа с БД) и слой представления (REST API).

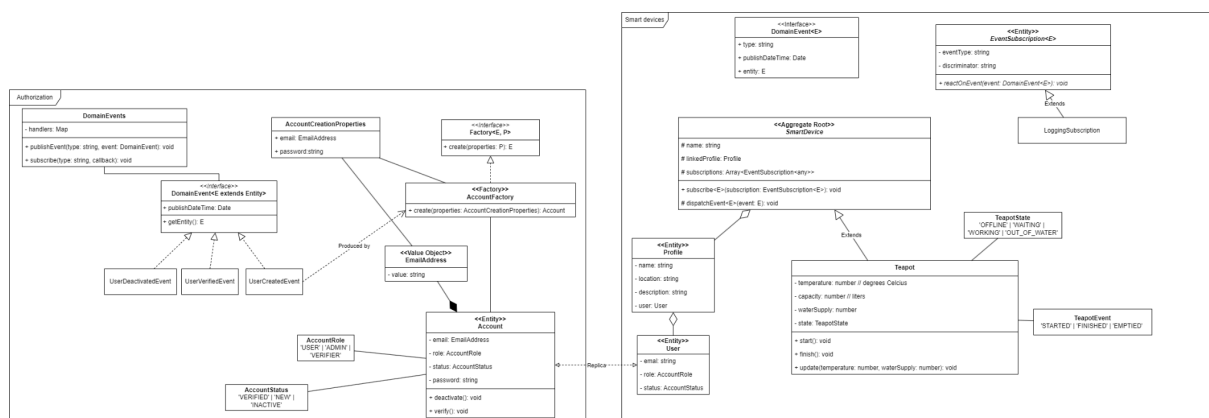


Рисунок 1 — UML диаграмма классов предметной области

Запуск сценариев по триггерам реализован через систему доменных событий и подписок.

Для работы с БД был выделен параметризованный абстрактный класс BaseRepository (рисунок 2).

```

1+ usages  Alexey Korotin
export default abstract class BaseRepository<ID extends Identifier, E extends Entity<ID>, M extends Model> {

    1+ usages  Alexey Korotin
    protected constructor(protected repository: Repository<M>, protected mapper: Mapper<E, M>) {
    }

    1 override  1+ usages  Alexey Korotin
    public async findById(id: ID): Promise<E> {
        const model: M | null = await this.repository.findById(id, {include: {all: true}})
        .then(m : M | null => {
            if (m) {
                return m.reload();
            }
            return m;
        });
        if (model) {
            return Promise.resolve(this.mapper.toEntity(model));
        }
        return Promise.reject({reason: {message: "Not found"}});
    }
}

```

Рисунок 2 — Фрагмент абстрактного класса BaseRepository

Функционал системы был разделен по логическим частям при помощи функционала роутов. Далее все роуты соединялись воедино в общем express-приложении (рисунок 3).

```

import express from 'express';
import profileRouter from './profile';
import teapotRouter from './teapot';

const routes : Express = express();
routes.use('/profiles', profileRouter);
routes.use('/devices/teapots', teapotRouter);

1+ usages  Alexey Korotin
export default routes;

```

Рисунок 3 — Express-приложение

## Вывод

В ходе выполнения лабораторной работы были изучены принципы построения Rest API средствами Express + Typescript