

Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”
(Университет ИТМО)

Факультет Программной инженерии и компьютерной техники

ОТЧЕТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ:
“Веб-проект WChunter”
структура и правила оформления
в соответствии с требованиями ГОСТ 7.32–2021

Проект выполняли:
Салеев Николай Р3220

Преподаватель:
Доц. факультета ПИиКТ Ефимчик Е.А

Санкт-Петербург
2021

Содержание

Содержание	2
Введение	3
1. Аналитический раздел	4
1.1 Идея	4
2 Высокоуровневое описание :	4
2.1 Описание аналогов:	4
2.2 Сравнительная таблица аналогов:	4
2.3 Вывод:	5
3. Раздел прецедентов использования	6
3.1 Список акторов	6
3.2 Диаграмма прецедентов использования	7
3.3 Спецификация прецедентов использования	7
4 Проектирование БД	14
4.1 Список сущностей	14
4.2 Отношение между сущностями	15
4.3 ER–диаграмма	15
4.4 Даталогическая модель	16
5. Проектирование интерфейсов	16
5.1 Список интерфейсов	16
5.2 Диаграмма для гостей и пользователей	23
5.3 Диаграмма для администратора и модератора	24
Диаграмма интерфейсов для модератора представлена на рисунке 5.4	25
6. Реализация проекта	26
6.1 Структура	26
6.2 База данных	35
7. Вывод	36

Введение

Целью работы является создание веб-приложения, отображающее на карте сохраненные пользователями точки. Для достижения поставленной цели необходимо решить следующие задачи:

- сформулировать идею приложения;
- проанализировать задачу и рассмотреть аналоги;
- спроектировать прецеденты использования;
- спроектировать структуру базы данных;
- спроектировать интерфейс приложения;
- выбрать стек технологий, на которых будет работать приложение;
- сверстать интерфейс и разработать клиентскую часть;
- написать серверную часть для обработки запросов от клиента;
- запустить приложение и проверить его работоспособность.

1. Аналитический раздел

1.1 Идея

Идея состоит в том, чтобы создать веб-приложение для быстрого нахождения доступного общественного туалета. Сортировка их по рейтингу и возможностью их комментировать.

2 Высокоуровневое описание :

Целевая аудитория - пользователи которые много времени проводят в городской среде вдали от своего дома. Нужда может застать врасплох каждого, особенно если ты курьер, человек в командировке или турист.

В системе существует 4 роли. Гость будет иметь доступ к просмотру карты и точек на ней. Так же он сможет просматривать комментарии, оставленные авторизованными пользователями. Авторизованный пользователь сможет добавлять точки, оставлять к ним комментарии. Так же он будет иметь доступ к личному кабинету, из которого будет иметь доступ ко всем добавленным им точкам. Модератор будет обрабатывать жалобы на добавленные точки. Администратор будет назначать модераторов. Естественно, все роли будут иметь доступ к карте с точками.

2.1 Описание аналогов:

- 2Gis – приложение с картами , которое используется в основном для построения маршрутов до точек. Этими точками могут быть туалеты , но в основном только платные.
- Яндекс карты — приложение с картами , имеет схожий функционал с 2Гис.
- "ГУП "Водоканал Санкт-Петербурга" — на сайте есть карта со всеми общественными туалетами СПб. Но нельзя добавлять новые , нет рейтинга точки и невозможно построить маршрут от текущего местоположения.

2.2 Сравнительная таблица аналогов:

В таблице 1.0 представлено сравнение аналогов с нашей задумкой.

Шкала: 0 - min, 10 - max

Таблица 1.0 - Сравнительная таблица аналогов

Продукт	Корректное внесение пользовательских меток	Проверка информации по меткам	Удобство использования	Инструменты для меток
2Gis	3 (зачастую информация неактуальна/некорректна)	3 (зачастую, информация неактуальна / некорректна	8	5 (есть возможность добавить только "другой объект")
Яндекс Карты	7(организовано энтузиастами с помощью яндекс толоки)	3 (зачастую, информация неактуальна / некорректна	9	2 (исключительно для дорожных происшествий)
"ГУП "Водоканал Санкт- Петербургa "	0 (отстутсвует инструмент)	0 (отстутсвует инструмент)	4 (нет возможности поиска пользовательской геопозиции, проблемы с навигацией)	0 (отстутсвует инструмент)

2.3 Вывод:

Ни один из рассмотренных аналогов не может предоставить функционал, позволяющий пользователям добавлять на карту отметки с туалетами. Наиболее близкий аналог - Яндекс Карты - даёт поставить отметку, но не даёт обозначить что это туалет. Во всех аналогах упомянуты лишь общественные туалеты, в то время как в нашу задачу входит внести туалеты с торговых центров, ресторанов быстрого питания и так далее

3. Раздел прецедентов использования

Данный раздел содержит список акторов, все прецеденты использования, действующие лица и их взаимодействие.

3.1 Список акторов

Неавторизированные пользователи могут:

- просматривать карту
- регистрироваться
- авторизовываться

Авторизированные пользователи могут:

- менять пароль
- добавлять точки
- комментировать точки
- жаловаться на точки
- добавлять в избранное
- все что могут гости

Модераторы могут:

- удалять точки
- просматривать жалобы
- все что могут пользователи и гости

Администраторы могут:

- назначать модераторов
- все что могут пользователи и гости

3.2 Диаграмма прецедентов использования

На рисунке 1.0 представлена диаграмма прецедентов использования для перечисленных выше акторов

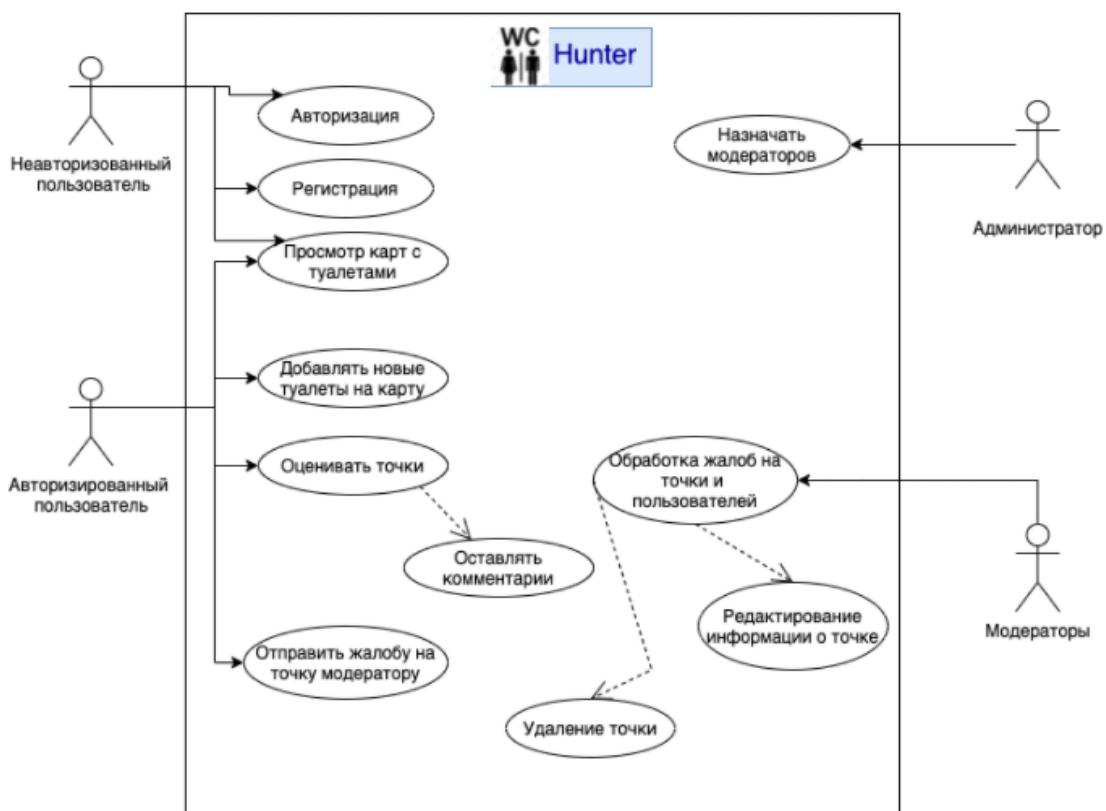


Рисунок 1.0 - Диаграмма прецедентов использования

3.3 Спецификация прецедентов использования

В таблицах 2.1-2.8 представлены прецеденты использования, которые доступны для перечисленных выше акторов

Таблица 2.1 - Прецедент использования регистрация

Краткое описание	этот прецедент использования позволяет гостю создать новую учетную запись в системе.
Действующие лица	Гость
Предусловия	гость захотел зарегистрировать новую учетную запись

Основной поток	гость заполняет регистрационную форму, после чего отправляет заявку на регистрацию. прецедент использования завершается.
Альтернативные потоки	гость неверно заполнил форму или пользователь с такими данными уже существует в системе. появляется ошибка и гость либо повторяет попытку регистрации с другими данными, либо отменяет ее, завершая прецедент использования
Постусловия	при успешном завершении прецедента в системе появится новый авторизованный пользователь, в противном случае система остается неизменной.

Таблица 2.2 - Прецедент использования авторизация

Краткое описание	этот прецедент использования позволяет гостю зайти в свою учетную запись в системе.
Действующие лица	Гость
Предусловия	гость уже имеет учетную запись в системе и хочет авторизоваться.
Основной поток	гость заполняет форму авторизации, после отправляет заявку на авторизацию. если пользователь с такими данными существует, то гость становится авторизованным пользователем. прецедент использования завершается.
Альтернативные потоки	гость ввел несуществующие или некорректные данные в форму. появится ошибка и будет предложено повторить попытку или отменить авторизацию. прецедент использования завершается.

Постусловия	если прецедент использования завершился успешно, то гость обретает новые права в системе.
-------------	-------------------------------------------------------------------------------------------

Таблица 2.3 - Прецедент использования просмотр меток на карте

Краткое описание	этот прецедент использования позволяет гостю и авторизированному пользователю просматривать метки на карте с кратким описанием (как добраться, пароль на двери итд)
Действующие лица	гость, авторизированный пользователь, модератор , администратор
Предусловия	гость или авторизированный пользователь хотят ознакомиться метками на карте
Основной поток	гость или авторизированный пользователь могут пользоваться просмотром меток на карте. прецедент использования завершится, когда гость или авторизированный пользователь закончит просмотр.
Альтернативные потоки	-
Постусловия	состояние системы не меняется от этого прецедента использования.

Таблица 2.4 - Прецедент использования добавление новых меток на карту

Краткое описание	этот прецедент использования позволяет авторизированному пользователю добавлять новые метки на карту.
Действующие лица	авторизированный пользователь
Предусловия	авторизированный пользователь захотел поделиться новой меткой с остальными пользователями (авторизованными и неавторизованными) сервиса.
Основной поток	авторизированный пользователь ставит новую метку на карте, при желании добавляя краткое описание (как добраться, пароль на двери итд). после определенного

	количества отметок «соответствует действительности» поставленных зарегистрированными пользователями статус метки сменяется с «не проверено» на «проверено пользователями»
Альтернативные потоки	авторизованный пользователь ставит новую метку на карте, при желании добавляя краткое описание (как добраться, пароль на двери итд). после определенного количества отметок «не соответствует действительности» поставленных зарегистрированными пользователями метке присуждается статус «ложная информация», метка автоматически удаляется, зарегистрированному пользователю, поставившему данную метку, урезается доступный функционал на 1 месяц (доступен только просмотр без добавления новых меток и оценивания других меток)
Постусловия	в системе появляется новая метка, имеющая статусы «не проверено», «проверено пользователями» или «ложная информация» (в последнем случае метка автоматически удаляется)

Таблица 2.5 - Прецедент использования оставление комментария к метке

Краткое описание	этот прецедент использования позволяет авторизованному пользователю оставить комментарий о метке.
Действующие лица	авторизованный пользователь
Предусловия	авторизованный пользователь захотел оценить метку
Основной поток	авторизованный пользователь, помимо выбора отзыва о метке, может дополнить его комментарием, например «информация действительна, на момент написания комментария код подходит к двери в KFC
Альтернативные потоки	-

Постусловия	к метке добавляется комментарий авторизованного пользователя.
-------------	---------------------------------------------------------------

Таблица 2.6 - Прецедент использования отправление жалобы на точку модератору.

Краткое описание	этот прецедент использования позволяет модератору удалить метку до набора необходимого количества отметок «не соответствует действительности», оставленных авторизованными пользователями.
Действующие лица	авторизованный пользователь
Предусловия	авторизованный пользователь отправил жалобу с подробным описанием, почему точку следует удалить (например, не соответствует действительности или в здании нет ничего, что подходило бы под описание метки)
Основной поток	модератор принимает решение об удалении метки. в случае удаления зарегистрированному пользователю, поставившему данную метку, урезается доступный функционал на 1 месяц (доступен только просмотр без добавления новых меток и оценивания других меток)
Альтернативные потоки	модератор принимает решение игнорировать жалобу. в таком случае метка остаётся на карте, но требует тех же действий, что и новая метка - после определенного количества отметок «соответствует действительности» поставленных зарегистрированными пользователями статус метки сменяется с «не проверено» на «проверено пользователем», или же после определенного количества отметок «не соответствует действительности» поставленных зарегистрированными пользователями метке присуждается статус «ложная информация», метка автоматически удаляется,

	зарегистрированному пользователю, поставившему данную метку, урезается доступный функционал на 1 месяц (доступен только просмотр без добавления новых меток и оценивания других меток)
Постусловия	в системе изменяется запись о метке (сituативно: и пользователе).

Таблица 2.7 - Прецедент использования назначение ролей в системе

Краткое описание	этот прецедент использования позволяет администратору присваивать/отнимать роли авторизированным пользователям.
Действующие лица	администратор
Предусловия	в состав сервиса требуется новый модератор. (или) модератор не соответствует критериям сервиса, администрация выносит решение о удалении данного пользователя с роли модератора.
Основной поток	администратор назначает авторизованного пользователя модератором.
Альтернативные потоки	администратор снимает с роли модератора.
Постусловия	в системе обновляется роль пользователя.

Таблица 2.8 - Прецедент использования удаление точки

Краткое описание	этот прецедент использования позволяет модератору удалить точку в случае несоответствия действительности информации или нарушений правил сервиса после рассмотрения жалобы,ставленной авторизованным пользователем.
------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Действующие лица	модератор
Предусловия	авторизованный пользователь отправил жалобу с подробным описанием, почему точку следует удалить (например, не соответствует действительности или в здании нет ничего, что подходило бы под описание метки)
Основной поток	после рассмотрения жалобы модератор принимает решение об удалении точки.
Альтернативные потоки	-
Постусловия	в системе изменяется запись о метке.

4 Проектирование БД

4.1 Список сущностей

Таблица 3 – Список сущностей в БД

Сущность	Свойство	Комментарий
User	Id	Primary Key.
	Login	Логин пользователя , по которому он будет авторизоваться в системе. Обязателен.
	Password	Пароль пользователя , по которому он будет авторизоваться в системе. Обязателен.
	Role	Роль пользователя в системе.
Complaints	Id	Primary Key.
	ToiletId	Primary key туалета , на который жалуются.
	Username	Никнейм пользователя, который оставил жалобу.
	Text	Содержание жалобы
Comment	ID	Primary Key.
	ComText	Текст комментария.
	Username	Никнейм пользователя , оставившего комментарий.
	Mark	Оценка
Toilet	ID	Primary Key.
	Name	Название точки
	Latitude	Ширина
	Longitude	Долгота
	Mark	Оценка
	Time	Время работы
	Type	Тип туалета
	Describe	Описание туалета

4.2 Отношение между сущностями

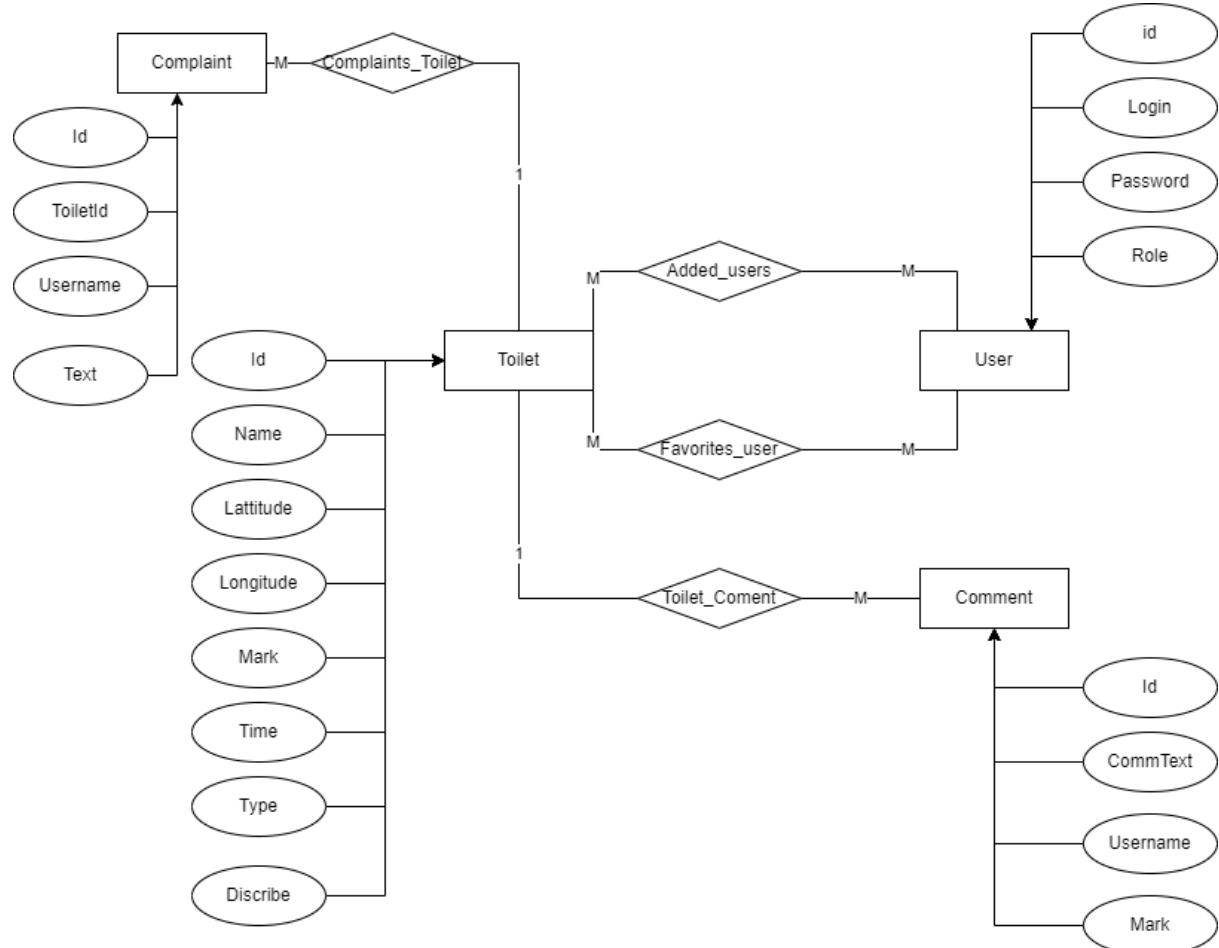
Таблица 3.1 – Список отношений между сущностями в бд.

Сущность	Тип связи	Сущность
Toilet	Многие ко многим Добавленные в избраное	User
Toilet	Многие к 1 Добавленные на карту точки текущим пользователем	User
Complaints	Многие к 1 Отправленные жалобы на точку	Toilet
Comments	Многие к 1 Отправленные комментарии на точку	Toilet

4.3 ER-диаграмма

ER-диаграмма отношений между сущностями представлена на рисунке 2.

Рисунок 2 - ER-диаграмма



4.4 Даталогическая модель

Даталогическая модель построенная на основе ER-диаграммы представлена на рисунке 3

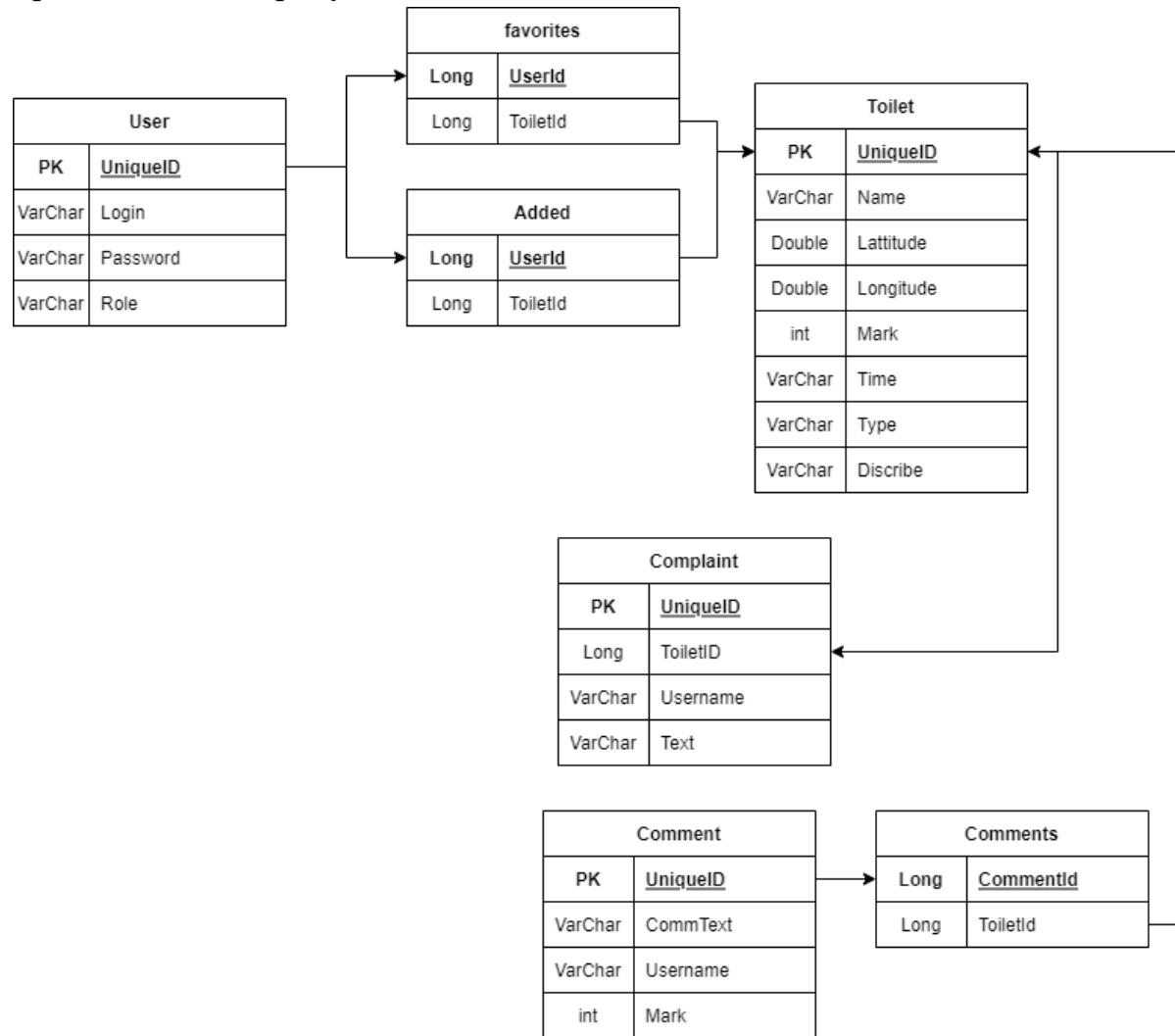


Рисунок 3 - Даталогическая модель

5. Проектирование интерфейсов

5.1 Список интерфейсов

Доступные гостю:

- Первая страница
- Интерфейс авторизации
- Интерфейс регистрации
- Карта

Доступные пользователю:

- Интерфейс добавления точки

- Интерфейс личного кабинета
- Интерфейс комментирования
- те, что доступны гостю

Доступные Модератору:

- Интерфейс обработки жалоб на точки
- те, что доступны гостю и пользователю

Доступные Администратору:

- Интерфейс назначения модераторов.
- те, что доступны гостю и пользователю

Эскизы интерфейсов представлены на рисунках 4.1-4.8

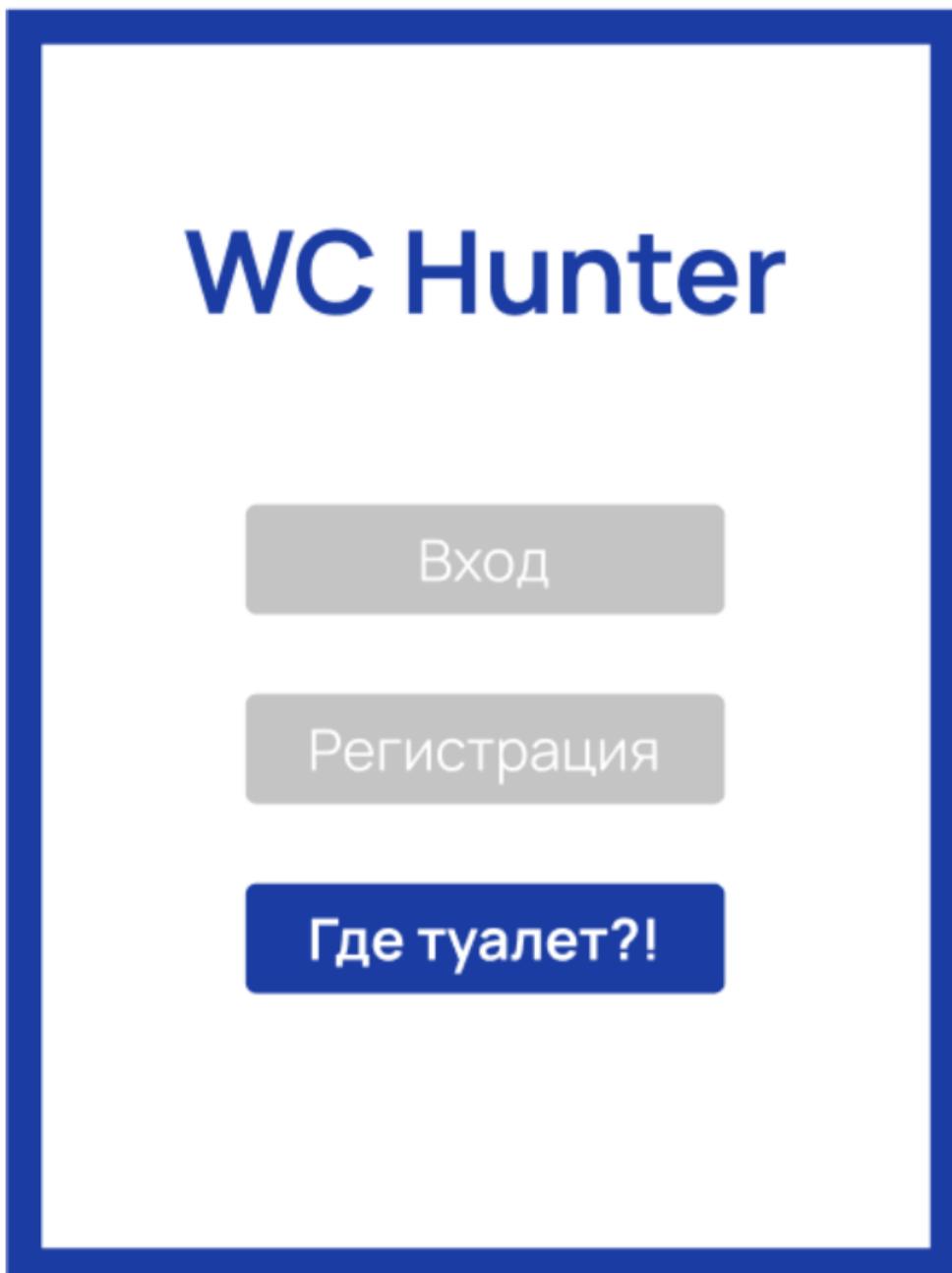


Рисунок 4.1 - Первая страница

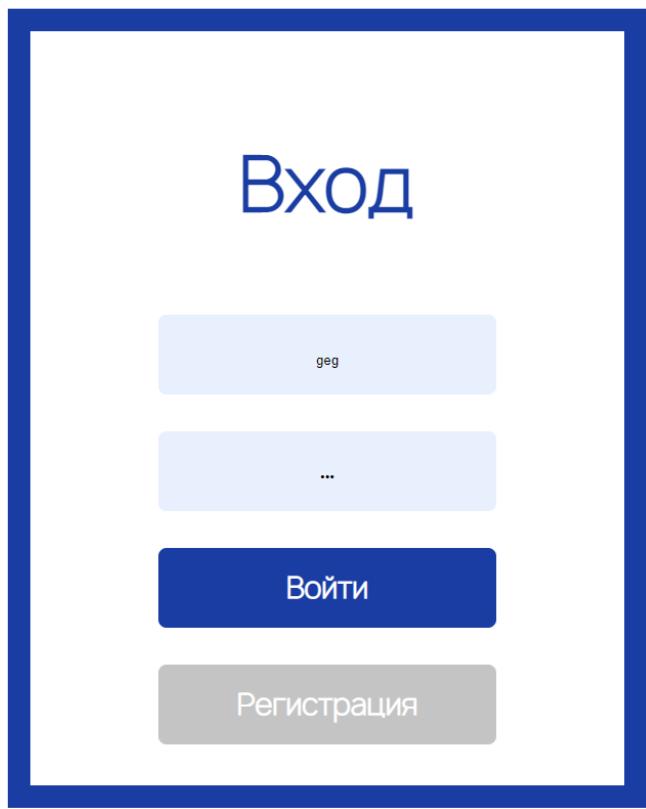


Рисунок 4.2.1 - форма авторизации

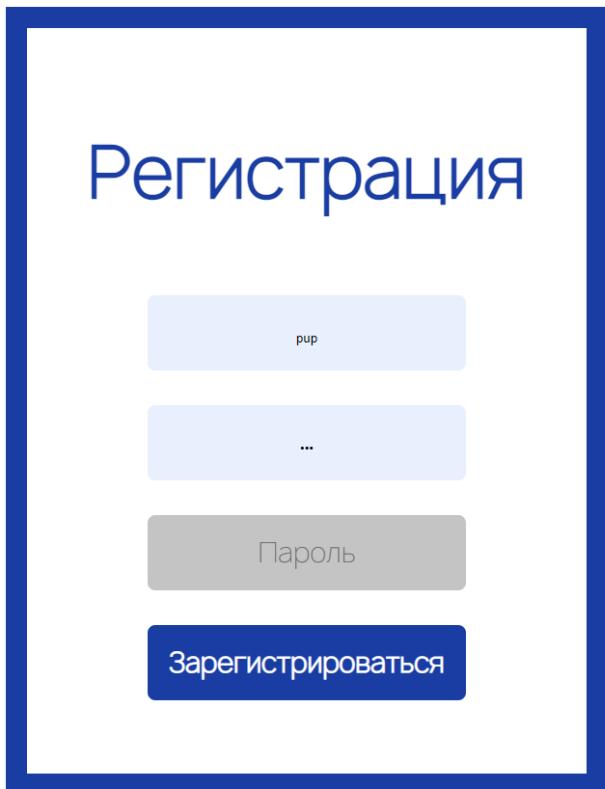


Рисунок 4.2.2 - форма регистрации

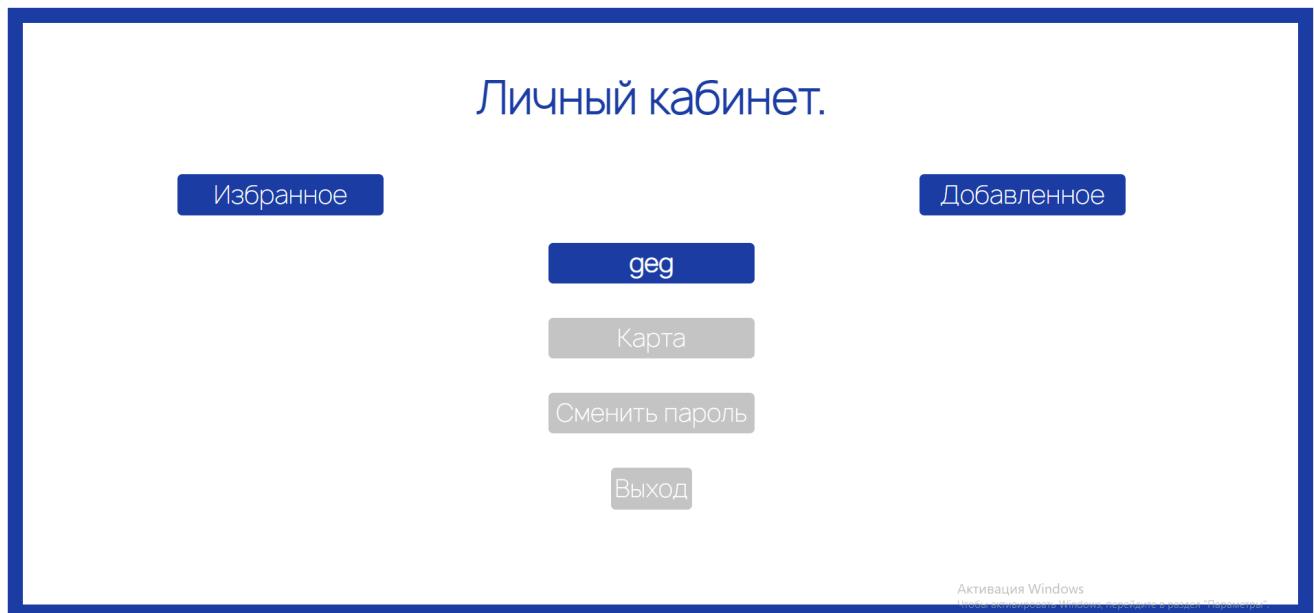


Рисунок 4.3 - интерфейс Личный кабинет

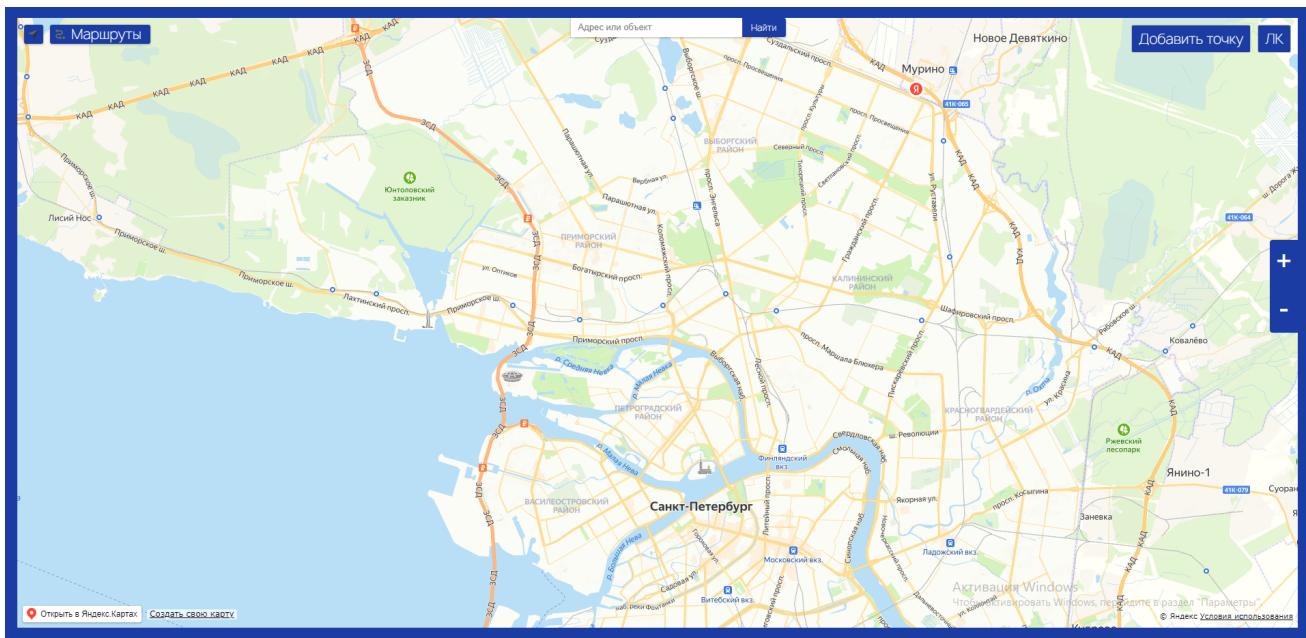


Рисунок 4.4 - интерфейс Карта

Рисунок 4.5 - Интерфейс Добавления точки

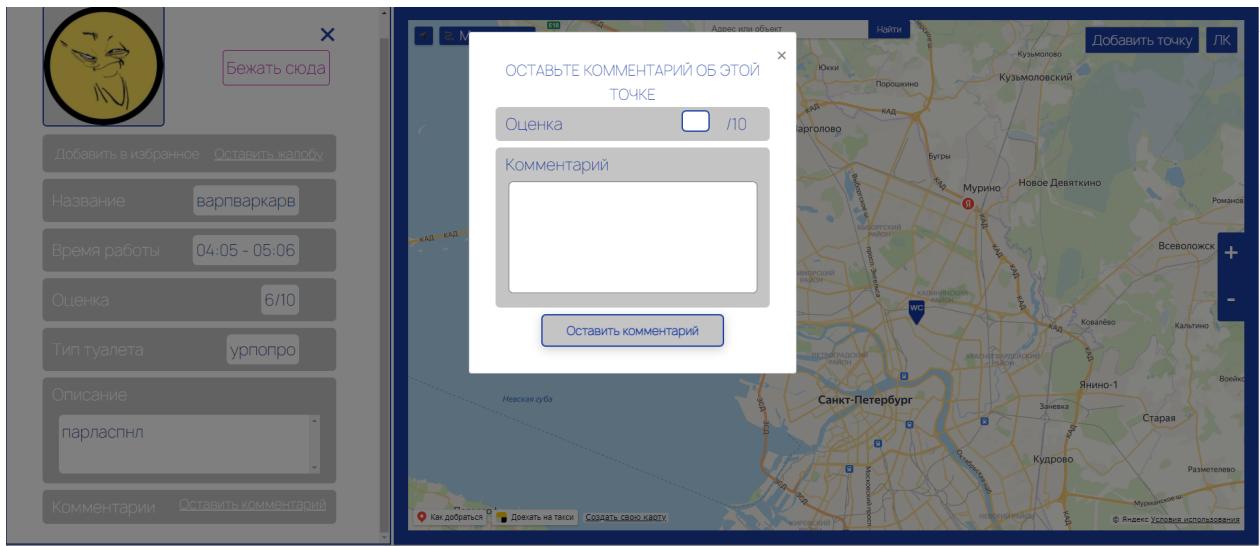


Рисунок 4.6 – Интерфейс комментирования

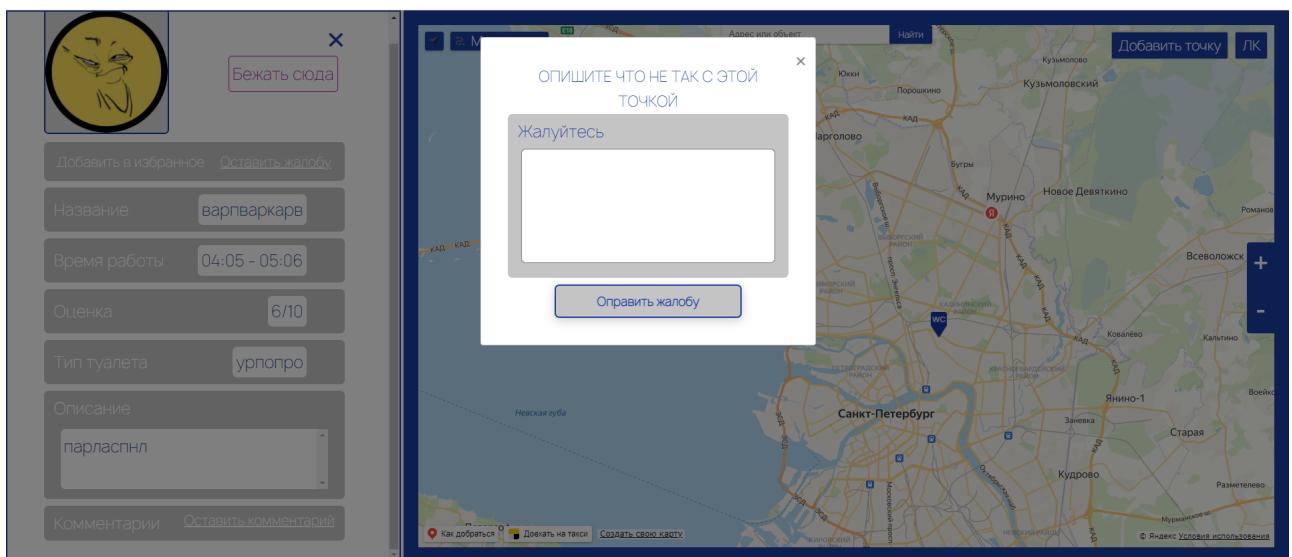


Рисунок 4.7 - Интерфейс добавления жалоб

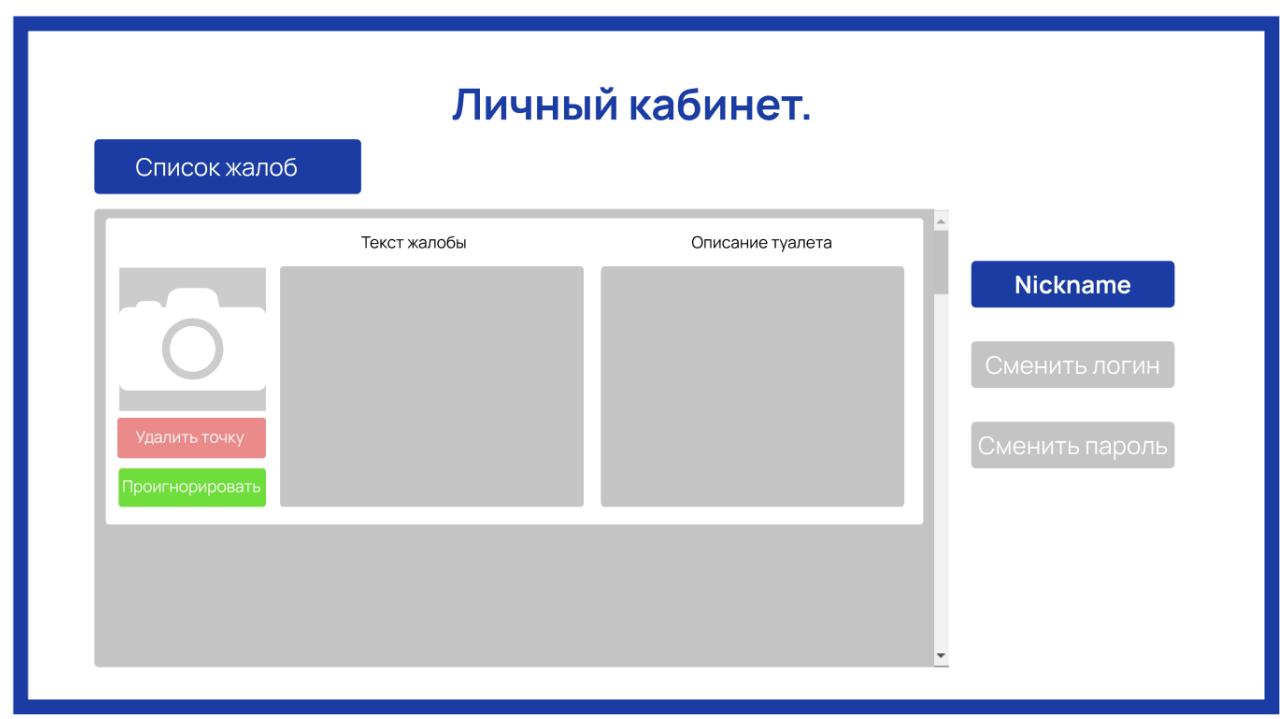


Рисунок 4.8 – Интерфейс обработки жалоб

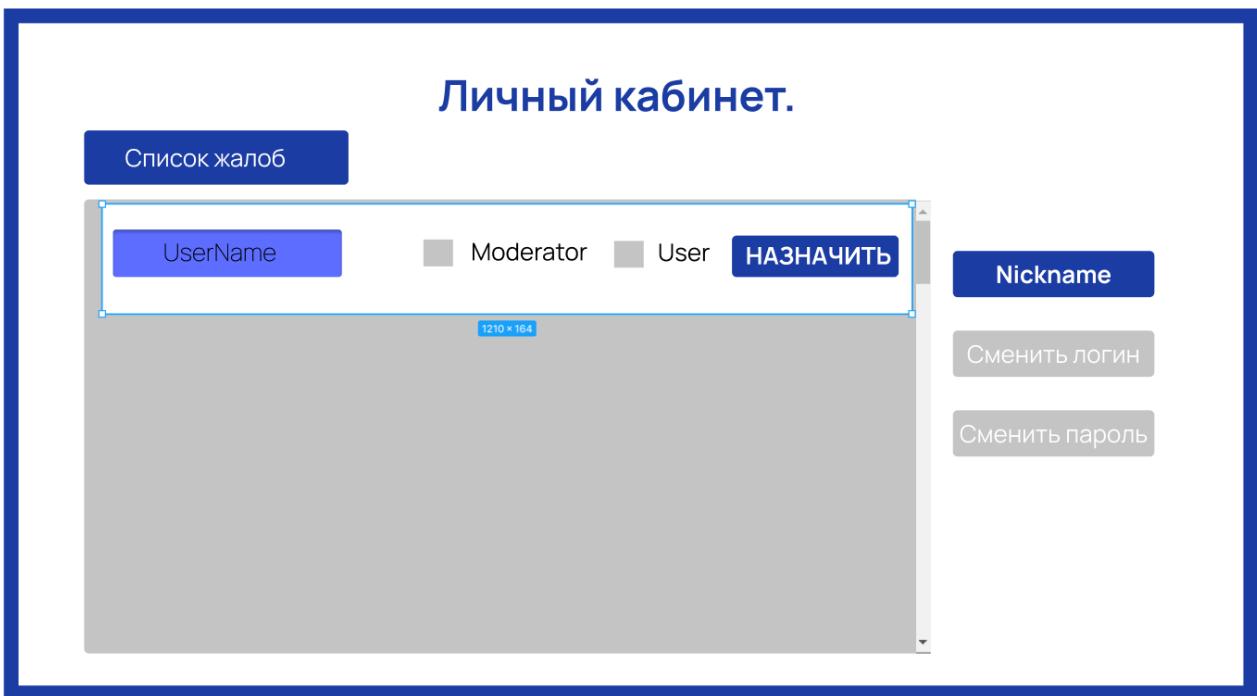


Рисунок 4.8 – Интерфейс Администратора для назначения роли

5.2 Диаграмма для гостей и пользователей

Диаграмма интерфейсов для гостей представлена на рисунке 5.1

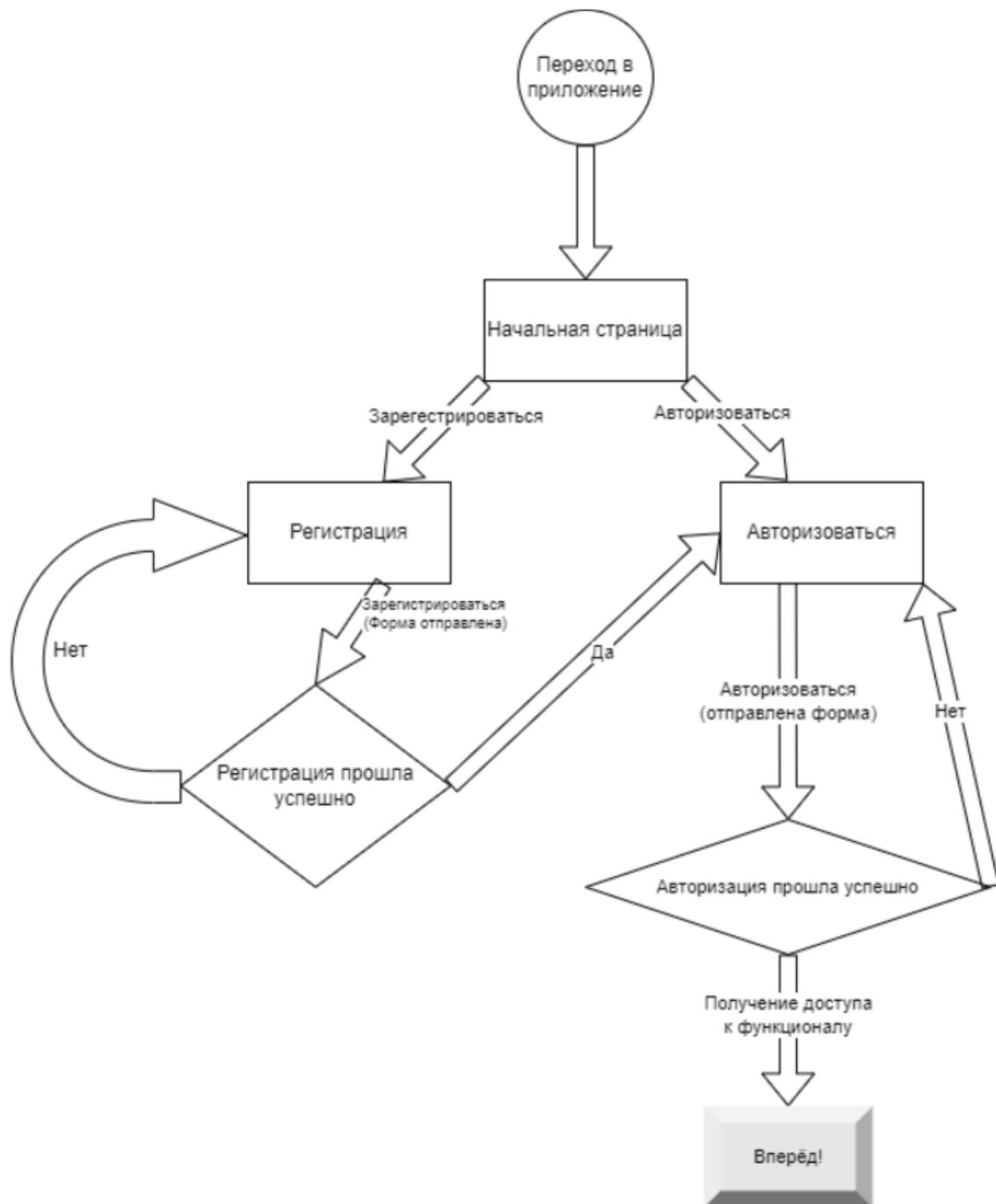


Рисунок 5.1 - Диаграмма для гостей

Диаграмма интерфейсов для авторизованных пользователей представлена на рисунке 5.2

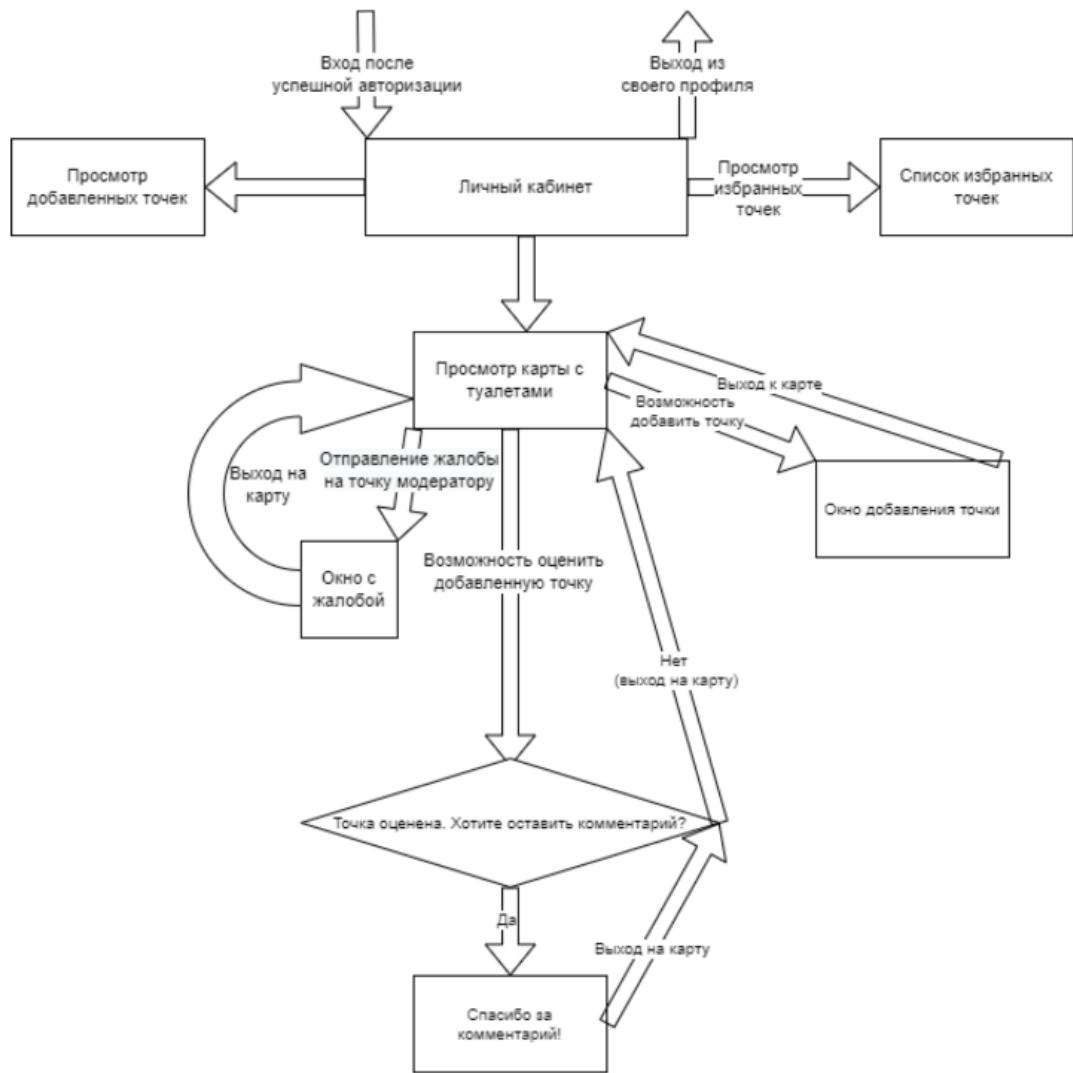


Рисунок 5.2

5.3 Диаграмма для администратора и модератора

Диаграмма интерфейсов для администратора представлена на рисунке 5.3

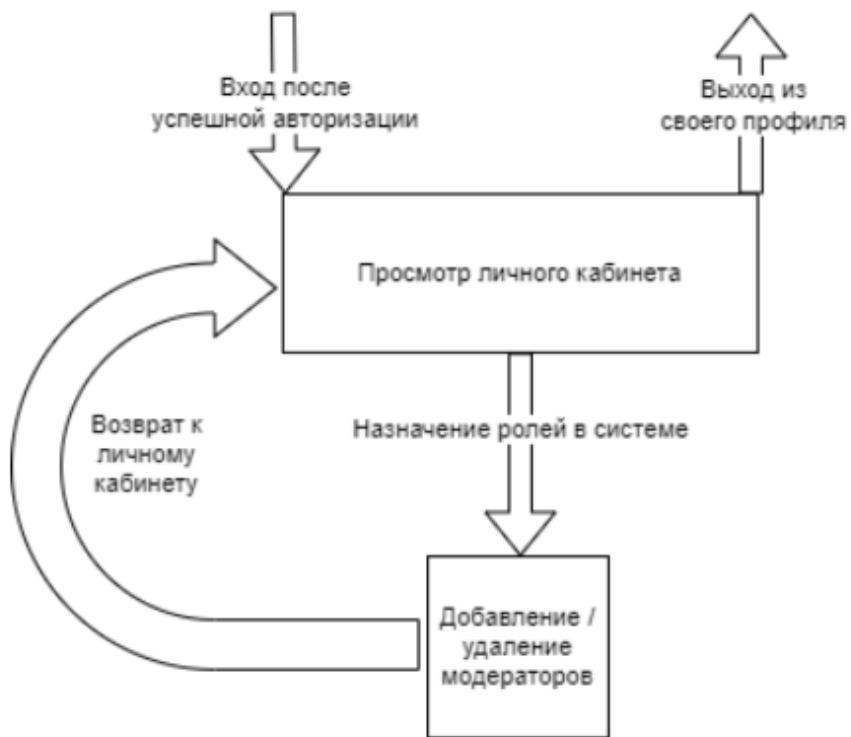


Рисунок 5.3 - Диаграмма для администратора

Диаграмма интерфейсов для модератора представлена на рисунке 5.4

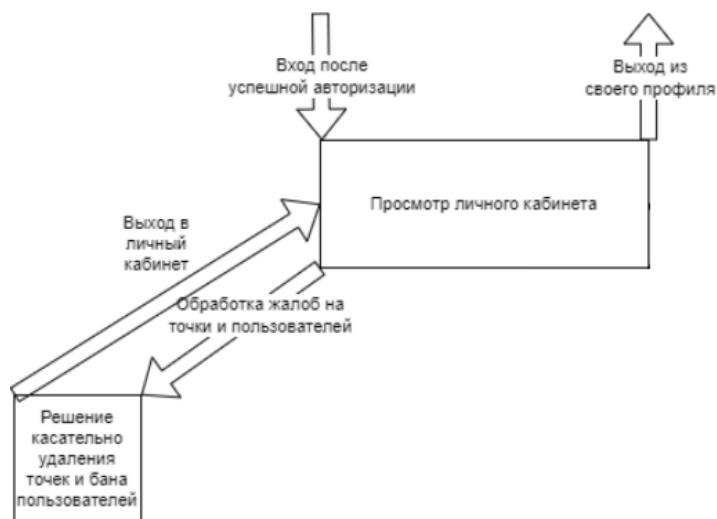


Рисунок 5.4 - Диаграмма для модератора

6. Реализация проекта

Для разработки нашего веб приложения мы использовали фреймворк Spring для бэк-энд части. Для фронтэнда использован JavaScript

6.1 Структура

На рисунке 6.1 представлена структура нашего приложения:

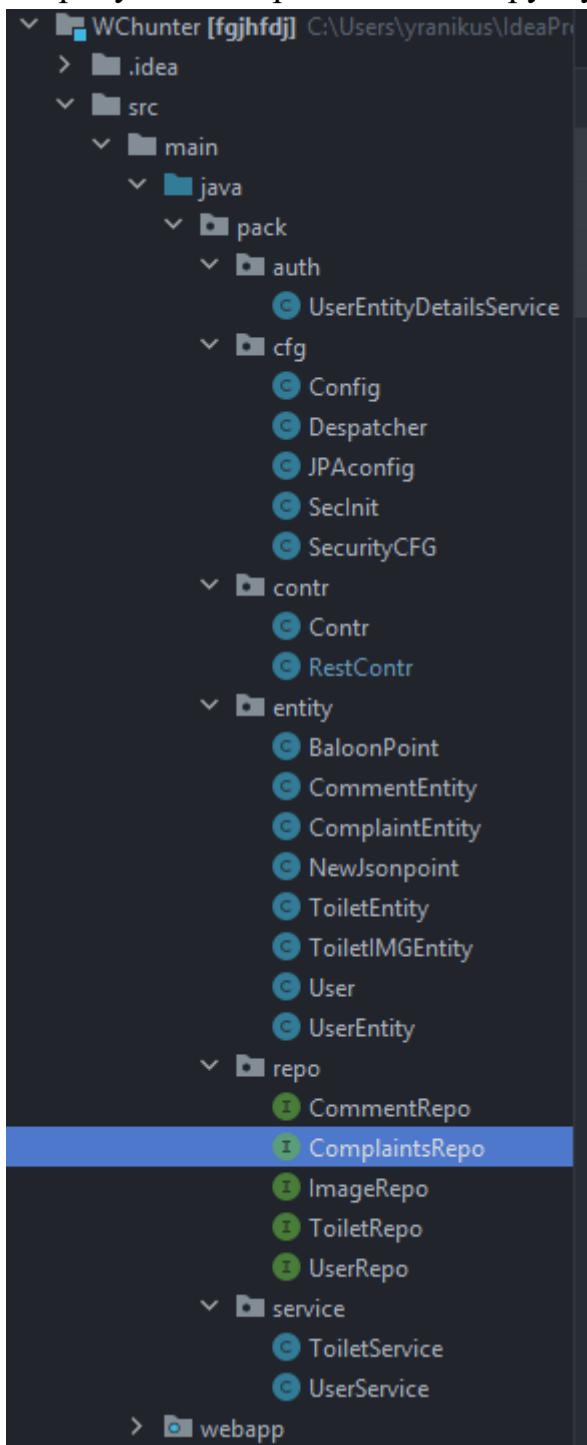


Рисунок 6.1 - структура приложения

- В директории cfg содержатся конфигурационные файлы приложения. На рисунке 6.2 фрагмент файла Config , который тимлиф и работу с тэмплейтами. На рисунке 6.3 кусок конфигурации доступа к базе данных.

```
    @ComponentScan("pack")
    @EnableWebMvc
    @Import({SecurityCFG.class,
             JPAconfig.class})
    @Configuration
    public class Config implements WebMvcConfigurer {

        @Bean
        public ObjectMapper mapper() { return new ObjectMapper(); }

        @Bean
        public SpringResourceTemplateResolver templateResolver() {
            SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();
            templateResolver.setApplicationContext(applicationContext);
            templateResolver.setCharacterEncoding("UTF-8");
            templateResolver.setPrefix("/WEB-INF/");
            templateResolver.setSuffix(".html");
            templateResolver.setTemplateMode(TemplateMode.HTML);
            return templateResolver;
        }

        @Bean
        public ThymeleafViewResolver viewResolver(final SpringTemplateEngine templateEngine){
            ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();
            viewResolver.setTemplateEngine(templateEngine);
            viewResolver.setCharacterEncoding("UTF-8");
            return viewResolver;
        }

        @Bean
        public SpringTemplateEngine templateEngine(final SpringResourceTemplateResolver templateResolver) {
            SpringTemplateEngine templateEngine = new SpringTemplateEngine();
            templateEngine.setTemplateResolver(templateResolver);
            templateEngine.setEnableSpringELCompiler(true);
            return templateEngine;
        }
    }
```

Рисунок 6.2 – файл Config.class

```

@Bean
public DataSource dataSource() {
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName("org.postgresql.Driver");
    dataSource.setPassword("f5a314180c3d422e01c8b11ab1cad7d08d1d634ee41b67cf9972844d76dc3681");
    dataSource.setUrl("jdbc:postgresql://ec2-18-235-165.compute-1.amazonaws.com:5432/d9m7abkvis7irf");
    dataSource.setUsername("bypbvxjvszeoyf");
    return dataSource;
}

@Bean
public JpaVendorAdapter jpaVendorAdapter() { return new HibernateJpaVendorAdapter(); }

@Bean
public Properties properties() {
    Properties properties = new Properties();
    properties.put("hibernate.dialect", "org.hibernate.dialect.PostgreSQL95Dialect");
    properties.put("hibernate.hbm2ddl.auto", "validate");
    return properties;
}

@Bean
public LocalContainerEntityManagerFactoryBean entityManagerFactory() {
    LocalContainerEntityManagerFactoryBean factoryBean = new LocalContainerEntityManagerFactoryBean();
    factoryBean.setDataSource(dataSource());
    factoryBean.setPackagesToScan("pack");
    factoryBean.setJpaVendorAdapter(jpaVendorAdapter());
    factoryBean.setJpaProperties(properties());
    factoryBean.afterPropertiesSet();
    return factoryBean;
}

@Bean
public PlatformTransactionManager transactionManager() {
    JpaTransactionManager jpaTransactionManager = new JpaTransactionManager();
    jpaTransactionManager.setEntityManagerFactory(entityManagerFactory().getObjectType());
    return jpaTransactionManager;
}

@Bean
public PersistenceExceptionTranslationPostProcessor exceptionTranslation(){
    return new PersistenceExceptionTranslationPostProcessor();
}

```

Рисунок 6.3 - файл JPAconfig

- В папке entity хранятся классы для обращения к базе данных через JPA, так же классы для формирования Json ответов в рест контроллерах. На рисунке 6.4 пример класса для работы с базой данных. На рисунке 6.5 класс для формирования Json объекта.

```
@Entity
@Table(name = "users")
public class UserEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String login;
    private String password;
    private String role;

    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(name = "favorites",
               joinColumns = @JoinColumn(name = "users", referencedColumnName = "ID"),
               inverseJoinColumns = @JoinColumn(name = "toilet", referencedColumnName = "ID"))
    private List<ToiletEntity> favorite;

    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(name = "added",
               joinColumns = @JoinColumn(name = "users", referencedColumnName = "ID"),
               inverseJoinColumns = @JoinColumn(name = "toilet", referencedColumnName = "ID"))
    private Set<ToiletEntity> added;
```

Рисунок 6.4 – фрагмент класса UserEntity

```

public class BaloonPoint {

    @JsonProperty("name")
    private String name;
    @JsonProperty("latitude")
    private Double latitude;
    @JsonProperty("longitude")
    private Double longitude;
    @JsonProperty("mark")
    private int mark;
    @JsonProperty("blime")
    private int blime;

    public int getBlime() { return blime; }

    public void setBlime(int blime) { this.blime = blime; }

    public BaloonPoint(String name, Double latitude, Double longitude, int mark) {
        this.mark = mark;
        this.name = name;
        this.latitude = latitude;
        this.longitude = longitude;
    }

    public int getMark() { return mark; }

    public void setMark(int mark) { this.mark = mark; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public Double getLatitude() { return latitude; }

    public void setLatitude(Double latitude) { this.latitude = latitude; }
}

```

Рисунок 6.5 – фрагмент класса BaloonPoint

- Папка contr содержит классы контроллеров. В классе Contr прописаны контроллеры , которые работают с тэмплейтами. В классе RestContr находятся рест-контроллеры , которые отправляют Json ответы на фронтэнд. На рисунке 6.6 пример рест-контроллера.

```

@RequestMapping(@"/point/{name}")
public ToiletEntity getPoint(@PathVariable String name){
    ToiletEntity toiletEntity = toiletService.findByName(name);
    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
    if (!auth.getName().equals("anonymousUser")) {
        UserEntity userEntity = userService.findByLogin(auth.getName());
        toiletEntity.setFavorite(userEntity.isFavorite(toiletEntity));
    }
    ToiletIMGEntity toiletIMGEntity = imgRepo.findById(toiletEntity.getId()).get();
    toiletEntity.setImg("data:image/jpeg;base64," + DatatypeConverter.printBase64Binary(toiletIMGEntity.getImage()));
    return toiletEntity;
}

```

Рисунок 6.6 – фрагмент класса RestContr

- В папке `repo` содержатся JPARepository для работой с базой данных. На рисунке 6.7 представлен класс `ToiletRepo`

```
@Repository
public interface ToiletRepo extends JpaRepository<ToiletEntity, Long> {

    ToiletEntity findAllByName(String name);
    ToiletEntity findByName(String name);
}
```

Рисунок 6.7 – класс ToiletRepo

- В папке `service` содержатся классы сервисного слоя. На рисунке 6.8 представлен фрагмент класса `ToiletService`

```
@Service
public class ToiletService {

    @Autowired
    private ComplaintsRepo complaintsRepo;

    @Autowired
    private UserService userService;

    @Autowired
    private CommentRepo commentRepo;

    @Autowired
    private ImageRepo imageRepo;

    @Autowired
    private ToiletRepo toiletRepo;

    @Transactional
    public void toiletSave( NewJsonpoint jsonpoint) {
        ToiletEntity toiletEntity = new ToiletEntity();
        toiletEntity.setLongitude(jsonpoint.getLong());
        toiletEntity.setLatitude(jsonpoint.getLat());
        toiletEntity.setName(jsonpoint.getName());
        toiletEntity.setMark(jsonpoint.getMark());
        toiletEntity.setDescribe(jsonpoint.getComment());
        toiletEntity.setTime(jsonpoint.getStartWork() + " - " + jsonpoint.getEndTime());
        toiletEntity.setType(jsonpoint.getType());

        String base64Image = jsonpoint.getPhoto().split( regex ",") [1];
        byte[] convertByte = DatatypeConverter.parseBase64Binary(base64Image);
        ToiletIMGEEntity toiletIMGEEntity = new ToiletIMGEEntity();
        toiletIMGEEntity.setImage(convertByte);
        imageRepo.saveAndFlush(toiletIMGEEntity);
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        UserEntity userEntity = userService.findByLogin(auth.getName());
        userEntity.addToilet(toiletEntity);
        toiletRepo.saveAndFlush(toiletEntity);
        userService.save(userEntity);
    }
}
```

Рисунок 6.8 - фрагмент файла ToiletService

- Папка webapp содержит статические ресурсы приложения, скрипты в том числе.
- JS Скрипты устроены следующим образом – у них модульная структура. По сути в проект мы подключаем только один файл, а к этому файлу подключаем все функции, которые нам нужны.
- Основной файл содержит import необходимых функций, после чего используется функция init() для инициализации карты.

The screenshot shows a code editor interface with a sidebar on the left displaying the project structure and a main panel showing the content of the `mapbasics.js` file.

Project Structure:

```

    OPEN EDITORS 1 UNSAVED
    Get Started
    ● JS mapbasics.js src/main/we... 9+
    admin.css src/main/webapp/reso...
    JS Sidebar.js src/main/webapp/... 6

    WCHUNTER
    > .idea
    < src/main
    > java
    < webapp
        < resources
            < css
                admin.css
                mapContros.css
                moder.css
                regstyle.css
                styles1.css
            > fonts
            > images
        < Scripts
            < JSONS
                points.json
            JS imageUpload.js
            JS mapbasics.js 9+
            JS Sidebar.js 6
        < WEB-INF
            admin.html
            FirstPage.html
            lk.html
            loginka.html
            map.html
            moderlk.html
            reg.html
        > target
        .gitignore
        package.json
        pom.xml
        Procfile

```

File Content (mapbasics.js):

```

src > main > webapp > resources > Scripts > JS mapbasics.js > ...
1 import [
2     toggleMap,
3     addMode,
4     addPoints,
5     updatePoints,
6     toggleBar,
7     commentMode,
8     closeModal,
9     modal,
10    claim,
11    claimMode,
12    nameError,
13    comment_modal_mode,
14    createMultiRoute,
15    coordsArr
16 ] from '/resources/Scripts/Sidebar.js';
17 import {upload, preview} from '/resources/Scripts/imageUpload.js';
18
19 export var imageSrc="";
20 export function addsrc(img){
21     imageSrc = img;
22 }
23
24 window.addEventListener('DOMContentLoaded', function() {
25
26     function setHeiHeight() {
27         $('.ramka').css({
28             height: $(window).height() + 'px'
29         });
30     }
31     setHeiHeight();
32     $(window).resize( setHeiHeight );
33
34
35     ymaps.ready(init);
36
37
38     var username,
39         role,
40         myMap,
41         coords,
42         myCollection,
43         geolockmark,
44         validMark = false,
45         validCoords = false;
46
47     console.log($('.button_layout'));
48

```

Рисунок 6.9 – фрагмент файла mapbasics.js

- Также в `mapbasics.js` лежат основные обработчики событий кликов по точкам и кнопкам карты, так как они могут быть вызваны только непосредственно в функции `init()`.

The screenshot shows a code editor with two panes. The left pane displays the project structure:

```

Get Started
mapbasics.js src/main/webapp... 9+
admin.css src/main/webapp/res...
Sidebar.js src/main/webapp... 6
WCHUNTER
.idea
src/main
java
webapp
resources
css
admin.css
mapContros.css
moder.css
regstyle.css
styles1.css
fonts
images
Scripts
JSONS
points.json
imageUpload.js
JS mapbasics.js 9+
JS Sidebar.js 6
WEB-INF
admin.html
FirstPage.html
lk.html
loginka.html
map.html
moderk.html
reg.html
target
.gitignore
package.json
pom.xml
Procfile

```

The right pane shows a fragment of the `mapbasics.js` file with line numbers from 339 to 393. The code handles a click event on a collection of objects, performing an AJAX request to get information about a selected point.

```

myCollection.events.add('click', (e) => [
    const target = e.get('target');
    const bar = document.querySelector("#commentBar");
    const namepoint = document.querySelector('#discrName').innerHTML;
    console.log(document.querySelector('#sidebar'));

    function getInfo(){
        $.ajax({
            url: `point/${target.properties._data_hintContent}`,
            type: 'GET',
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            success: function (data) {
                currentLatitude = data.latitude;
                currentLongitude = data.longitude;
                document.querySelector('.favorite_btn').innerHTML="Добавить в избранное";
                document.querySelector('.favorite_btn').classList.remove('favorite_btn1');
                photoDiv.innerHTML = "";
                document.querySelector('#comment_pool').innerHTML = "";
                photoDiv.insertAdjacentHTML('afterbegin', `
                    <img class="preview-image" src='${data.img}' alt="Вот как-то так"/>
                `);

                console.log(data);
                document.querySelector('#discrD').innerHTML = data.describe;
                document.querySelector('#discrName').innerHTML = target.properties._data_hintContent;
                document.querySelector('#discrTime').innerHTML = data.time;
                document.querySelector('#discrMark').innerHTML = data.mark + "/10";
                document.querySelector('#discrType').innerHTML = data.type;
                console.log(data.comment);
                if(data.comment.length != 0){
                    for (let obj of data.comment) {
                        $('#comment_pool').append(`<div id='comments${obj.id}' class="one_comment">` +
                            `<div id="commentName">${obj.username}</div>` +
                            `<div id="commentRate">${obj.mark}/10</div>` +
                            `<div class="comment_text comment_text_small">${obj.comment}</div></div>`);
                    }
                }
                document.querySelector('#commentInsideBar').insertAdjacentElement('afterbegin', photoDiv);

                console.log(data.favorite);
                if(data.favorite){
                    document.querySelector('.favorite_btn').classList.add('favorite_btn1');
                    document.querySelector('.favorite_btn').innerHTML="Удалить из избранного";
                }
            },
            error: function (errMsg){
                document.querySelector('#discrName').innerHTML = target.properties._data_hintContent;
                document.querySelector('#discrD').innerHTML = target.properties._data_balloonContentHeader;
                console.log(errMsg);
            }
        });
    }
]);

```

Рисунок 6.10 – фрагмент файла `mapbasics.js` с пример обработчика нажатий по коллекции геообъектов, при клике на которую отправляется гет запрос на получение информации о точке.

- В файле `sidebar` лежат функции, которые могут функционировать и вне функции `init()`, например обработчик событий для кнопок сайдбара с информацией о точке.

The screenshot shows a fragment of the `mapbasics.js` file focusing on the logic for handling events on the sidebar's favorite button.

```

favoriteBtn.addEventListener('click', () =>{
    if(favoriteBtn.classList.contains('favorite_btn1')){
        $.post(`deleteFavorites/${document.querySelector('#discrName').innerHTML}`, function(data){
            favoriteBtn.classList.remove('favorite_btn1');
            favoriteBtn.innerHTML="Добавить в избранное";
        });
    }else{
        $.post(`addFavorites/${document.querySelector('#discrName').innerHTML}`, function(data){
            favoriteBtn.classList.add('favorite_btn1');
            favoriteBtn.innerHTML="Удалить из избранного";
        });
    }
});

```

Рисунок 6.11 – фрагмент файла `mapbasics.js` с пример обработчика нажатий по кнопке избранного

- В файле imageUpload содержится код, для того, чтобы загружать предпросмотр картинки и управлять input'ом файла(картинки)

```

import {addsrc} from '/resources/Scripts/mapbasics.js';
export const preview = document.createElement('div');
export function upload(selector, buttonSelector, options = {}){

    const input = document.querySelector(selector);
    const openButton = document.querySelector(buttonSelector);

    if(options.multi){
        input.setAttribute('multiple', true);
    }

    if(options.accept && Array.isArray(options.accept)){
        input.setAttribute('accept', options.accept.join(','));
    }

    input.insertAdjacentElement('afterend', preview);

    const triggerInput = () => input.click();

    const changeHandler = event => [
        // if(!event.target.files.length){
        //     return;
        // }

        console.log(event.target.files);
        const files = Array.from(event.target.files);

        preview.innerHTML = '';
        // files.forEach(file =>{
        //     if(!file.type.match('image')){
        //         return;
        //     }

        const reader = new FileReader();

        reader.onload = event => {
            const src = event.target.result;
            addsrc(src);
            preview.insertAdjacentHTML('afterbegin', `
                <div class="preview-image" style="background-image: url('${src}');">
                    <div class="preview-remove">&times;</div>

                    <div class="preview-info">
                        <span>${files[0].name}</span>
                        ${files[0].size}
                    </div>
                </div>
            `);
        };

        reader.readAsDataURL(files[0]);
        preview.classList.remove('hide');
        openButton.classList.add('hide');
        preview.classList.add('preview');
        // });
    ];
}

```

Рисунок 6.12 – фрагмент файла imageUpload.js

6.2 База данных

Наша база данных содержит 8 таблицы (Рисунок 7.0), из них 2 – таблицы связности.

>	added
>	comment
>	comments
>	complaints
>	favorites
>	toilet
>	toiletentity
>	users

Рисунок 7.0 - листы БД

На скриншотах можно увидеть пример данных на каждом из листов БД (Рисунки 7.1-7.4)

	id	login	password	role
1	1	geg	\$2a\$10\$ros08lmvGk8s.fPpuswpAe6Ef7xXx6s0sppdxPqDDdNJfF3C1yfke	USER
2	2	pop	\$2a\$10\$w1/c.tcswy7N0S5CyvnYezrXSruIxeXNQCWL86Ro5gifgb2aME6	USER

Рисунок 7.1 - лист users

	id	discribe	latitude	longitude	mark	name	time	type
1	1	ghkghk	59.96280857280104	30.300131937955456	6	Point	03:04 - 05:06	hgckjghk
2	2	Туалет	59.95672179779723	30.311598495642535	3	Теремок	09:07 - 09:59	TTT

Рисунок 7.2 - лист toilet

	id	comment	mark	username
1	1	sdfbvdf	5	pop
2	2	ва	10	pop
3	3	оставить комм	2	pop

Рисунок 7.4 - лист comment

	id	complain	username	toiletentity_id
1	1	kmealk['slfk v	pop	1
2	2	kjbwekdbacp	pop	2
3	3	;sadm ;	pop	2

Рисунок 7.5 – лист complaints

7. Вывод

В результате нашей работы мы создали задуманное рабочее веб-приложение. В ходе работы мы придумали идею веб-сайта, рассмотрели аналоги, спроектировали прецеденты использования, структуру базы данных, а также интерфейсы приложения. Для реализации мы выбрали язык программирования Java, JavaScript и другие технологии, которые помогли нам сверстать интерфейс, настроить работу баз данных и серверной части, обрабатывающей запросы от пользователей. По завершению работы мы запустили приложение на хостинге и проверили его работоспособность, на данный момент приложение доступно в интернете под доменным именем wchunter.tech, на котором работает, что позволяет сделать вывод об успешности проделанного проекта.

8. Список использованных источников

Яндекс API - <https://yandex.ru/dev/maps/jsapi/doc/2.1/quick-start/index.html?from=techmapsmain>