1. Общая концепция системы

Информационная система представляет собой **веб-агрегатор площадок и аудиторий для проведения мероприятий**. Основная задача — обеспечить удобный поиск, просмотр, бронирование и управление площадками как для клиентов, так и для организаторов, а также предоставить администратору полный контроль над системой.

Платформа работает по модели **трёх ролей**:

1. **Гость (неавторизованный пользователь)** — может просматривать информацию о площадках.
2. **Пользователь (клиент)** — регистрируется, ищет площадки, бронирует, оплачивает, оставляет отзывы.
3. **Администратор** — управляет всеми объектами системы, модерирует отзывы, добавляет площадки и контролирует бронирования.

---

2. Основные функции системы

2.1. Для пользователя (клиента)

- Регистрация и авторизация.
- Личный кабинет с историей бронирований.
- Поиск площадок по фильтрам:
  - тип помещения;
  - вместимость;
  - стоимость;
  - расположение;
  - доступные даты.
- Просмотр карточки площадки с описанием, фотографиями, картой и отзывами.
- Возможность выбрать свободную дату в календаре.
- Создание заявки на бронирование.
- Возможность онлайн-оплаты (в дипломе можно описать как интеграцию через API, а в реальности реализовать имитацию).
- Оставление отзыва после мероприятия.

2.2. Для администратора

- Вход в административную панель.
- Управление объектами:
  - площадки;
  - фотографии;
  - пользователи;
  - бронирования;
  - отзывы.
- Модерация отзывов.
- Управление доступностью площадок.
- Просмотр аналитики (в дипломе можно указать как перспективное расширение).

3. Архитектура и стек технологий (обоснование)

3.1. Клиентская часть: React

Почему React:

- высокая скорость разработки за счёт компонентов;
- большое количество готовых библиотек (карты, календари, ассинхронные запросы);
- удобная работа с REST API;
- модульность и переиспользуемость кода;
- возможность в дальнейшем сделать PWA.

С React создаются:

- страницы каталога площадок;
- карточки площадок;
- форма бронирования;
- личный кабинет;
- админ-панель (опционально через React или стандартную Django admin).

3.2. Серверная часть: Django + Django REST Framework

Почему Django:

- строгая структура MVC;
- высокая скорость разработки;
- встроенная система аутентификации;
- простая интеграция с PostgreSQL;
- удобный встроенный admin-panel.

Почему Django REST Framework:

- удобное создание REST API для фронтенда на React;
- сериализация данных;
- аутентификация токенами / JWT;
- гибкая работа с правами доступа (permissions);
- стандартизированный обмен данными.

Сервер обеспечивает:

- API для фронтенда;
- бизнес-логику бронирований;
- управление ролями (пользователь/администратор);
- хранение данных;

- ограничение доступа к операциям.

3.3. База данных: PostgreSQL

Почему PostgreSQL:

- надёжная реляционная БД;
- поддержка транзакций (важно для бронирований, чтобы не возникало "двойных" броней);
- мощные возможности по работе с JSON, GIS (в дальнейшем можно добавить карту);
- совместимость с Django.

Основные таблицы БД:

- Пользователи
- Площадки
- Фотографии площадок
- Бронирования
- Отзывы
- Категории площадок (если нужно)
- Роли пользователей / права доступа

3.4. Авторизация и роли

Используется Django Authentication + JWT-токены.

Роли:

- **Admin** — `is_staff = True`
- **User** — стандартная модель пользователя

Можно расширить модель пользователя (Django User model extension), добавив:

- телефон;
- организацию;
- номер документа.

---

4. Логика работы сайта (User Flow)

4.1. Гость

1. Заходит на сайт → видит каталог площадок.
2. Может применять фильтры и просматривать карточки.
3. Чтобы забронировать — проходит регистрацию/авторизацию.

4.2. Клиент

1. Авторизуется.
2. Выбирает площадку.
3. Нажимает «Забронировать».
4. Выбирает дату и время.
5. Система проверяет свободна ли дата (через запрос к БД).
6. Создаётся заявка **со статусом “Ожидает подтверждения”**.
7. Пользователь может:
    - отменить бронирование,
    - оплатить,
    - оставить отзыв после мероприятия.

4.3. Администратор

1. Входит в админ-панель.
2. Видит список площадок.
3. Может редактировать:
    - описание,
    - фото,
    - цены,
    - доступность.
4. Управляет бронированиями:
    - подтверждает,
    - отменяет,
    - вручную создаёт.
5. Модерирует отзывы.
6. Управляет пользователями.

---

5. Структура приложения (модули)

Модули Django:

1. **users**
    - регистрация
    - роли
    - профили
    - авторизация
2. **venues (площадки)**
    - модели площадок
    - фото
    - фильтры
3. **bookings**
    - бронирования
    - статусы (pending, confirmed, cancelled)

    - проверки доступности
4. **reviews**
    - отзывы
    - рейтинг
5. **api**
    - DRF endpoints

React приложения:

1. Главная страница
2. Каталог площадок
3. Карточка площадки
4. Форма поиска
5. Форма бронирования
6. Личный кабинет пользователя
7. Вход/регистрация
8. Админ-панель (если не использовать стандартную Django)

---

6. Почему такая архитектура подходит для диплома

- Разделение на **frontend + backend + database** показывает знание современной архитектуры.
- Django + DRF — стандарт в разработке REST-систем.
- React — самый востребованный современный фронтенд-фреймворк.
- PostgreSQL — промышленная БД, а не учебная.
- Чёткое разделение ролей (user/admin).
- Система является полностью рабочей и расширяемой.