Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

Кафедра информационных систем и технологий

Дисциплина: Конструирование программного обеспечения

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

«Программная система для врачей и работников регистратуры поликлиники»

Студент: гр. 181075 Клименков Е.А.

Руководитель: Скудняков Ю.А.

Минск 2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc105368913)

[1 Постановка задачи 5](#_Toc105368914)

[2 Анализ существующих решений 6](#_Toc105368915)

[3 Выбор и обоснование среды разработки и языка программирования 8](#_Toc105368916)

[3.1 Выбор и обоснование среды разработки 8](#_Toc105368917)

[3.2 Выбор и обоснование языка программирования 9](#_Toc105368918)

[4 Проектирование модели Программы 10](#_Toc105368919)

[5 Проектирование интерфейса 14](#_Toc105368920)

[6 Реализация программного средства 16](#_Toc105368921)

[6.1 Требования к программному продукту 16](#_Toc105368922)

[6.2 Руководство пользователя 16](#_Toc105368923)

[6.3 Тестирование и отладка 19](#_Toc105368924)

[ЗАКЛЮЧЕНИЕ 22](#_Toc105368925)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 23](#_Toc105368926)

[ПРИЛОЖЕНИЕ А (обязательное) Текст программы 24](#_Toc105368927)

## ВВЕДЕНИЕ

Одним из важнейших процедур обработки структурированной информации является сортировка и поиск. Сортировкой называют процесс перегруппировки заданной последовательности объектов в некотором определенном порядке. Определенный порядок (например, упорядочение в алфавитном порядке, по возрастанию или убыванию количественных характеристик, по классам, типам и.т.п.) в последовательности объектов необходимо для удобства работы с этим объектом. В частности, одной из целей сортировки является облегчение последующего поиска элементов в отсортированном множестве. Под поиском подразумевается процесс нахождения в заданном множестве объекта, обладающего свойствами или качествами задаваемого шаблона.

Алгоритмов сортировки существует огромное множество и зачастую не понятно, чем О(n) отличается от О(log (n)). Для решения этой проблемы можно визуализировать алгоритм сортировки, что позволяет не только увидеть скорость сортировки, но и визуально увидить каким образом сортируется какой-либо массив данных.

Задачи выполнения данного проекта:

* проанализировать техническую и специализированную литературу по теме курсового проекта;
* рассмотреть особенности различных алгоритмов сортировки;
* изучить раличные сложности алгоритмов сортировки;
* ознакомиться с технологиями визуализации данных;
* реализовать технологический проект в соответствии с изученными технологиями.

Исходя из актуальности проблемы, была сформулирована тема исследования: «Создание визуализатора сортировки»

электронный справочник учебное пособие

## 1 Постановка задачи

Задачей данной работы была разработка «Программная система для врачей и работников регистратуры поликлиники». Веб-сайт предназначен для эксплуатации сотрудниками и обычными гражданами поликлиник.

Веб-сайт должен обеспечивать:

• Хранение сведений об участках, к которым относятся к поликлиники, о расписании работы участковых врачей, информацию о врачах, а также карточки пациентов;

• Карточка имеет номер, в нее заносятся сведения о каждом посещении поликлиники пациентом: дата посещения, жалобы, предварительный диагноз, назначения, выписан или нет больничный лист, и если выписан, то на какой срок, имя врача. В карточке на первой странице указаны также фамилия, имя, отчество пациента, его домашний адрес, пол и возраст, номер страхового полиса, дата заполнения карточки;

• В расписании работы врачей указывается, на каком участке работает врач, дни и часы приема, номер кабинета;

• Врач может обслуживать более одного участка. В случае увольнения врача его участок передается другим врачам. Данные о враче, которые хранятся в БД, это фамилия, имя отчество, категория, стаж работы, дата рождения. В карточку больного при каждом его посещении поликлиники врачом заносится очередная запись;

•Работники регистратуры регистрируют пациента, заполняя первую страницу его карточки. При увольнении врача его больных передают другому врачу;

Программа должна отображать:

• Сведения об участках;

• Карточки пациентов;

• Сведения о врачах.

Требования к надежности:

• программа устойчива к ошибкам;

• бесперебойное функционирование в процессе эксплуатации.

## 2 Анализ существующих решений

На текущем рынке существует довольно большое количество программных решених для поликлиник, преобладающее количество решений ялвяется частным решений крупных предприятий, предоставляющих своё программное обеспечение на тестовый период, либо же как полноценный продукт за подписку.

Не все из представленных решений предоставляют многофункциональный сайт.

Почему именно сайт? Сайт не требователен к программному оборудованию, т.е. убавится меньше жалоб, что программное обеспечение не работает или зависает. Персонал и граждане смогут открыть сайт и воспользоваться удобным интерфейсом.

К тому же этому сайту нет конкуренции, потому что он объединяет все возможные участки и их поликлиники.

## 3 Выбор и обоснование среды разработки и языка программирования

### 3.1 Выбор и обоснование среды разработки

[React](https://ru.reactjs.org/) — это JavaScript-библиотека для создания пользовательских интерфейсов. Обратите внимание, что это именно библиотека, а не фреймворк. React часто называют фреймворком, но это ошибка. Во-первых, его использование ни к чему вас не обязывает, не формирует «фрейм» проекта. Во-вторых, React выполняет единственную задачу: показывает на странице компонент интерфейса, синхронизируя его с данными приложения, и только этой библиотеки в общем случае недостаточно для того, чтобы полностью реализовать проект.

React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость разработки, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и GraphQL.

TypeScript **—** [язык программирования](https://ru.wikipedia.org/wiki/Язык_программирования), представленный [Microsoft](https://ru.wikipedia.org/wiki/Microsoft)в [2012 году](https://ru.wikipedia.org/wiki/2012_год)и позиционируемый как средство разработки [веб-приложений](https://ru.wikipedia.org/wiki/Веб-приложение), расширяющее возможности [JavaScrip](https://ru.wikipedia.org/wiki/JavaScript)t.

JSON Server — это простой проект, который поможет вам очень быстро настроить REST API с операциями CRUD.

Redux — это библиотека JavaScript с открытым исходным кодом для управления и централизации состояния приложения. Чаще всего он используется с такими библиотеками, как React или Angular, для создания пользовательских интерфейсов.

### 3.2 Выбор и обоснование языка программирования

React это библиотека для создания пользовательских интерфейсов. Одной из ее отличительных особенностей является возможность использовать JSX, язык программирования с близким к HTML синтаксисом, который компилируется в JavaScript. Разработчики могут добиваться высокой производительности приложений с помощью Virtual DOM. C React вы можете создавать [изоморфные приложения](https://xbsoftware.ru/blog/react-seo-izomorphnost-react-odnostrannoe-prilozhenie/), которые помогут вам избавиться от неприятной ситуации, когда пользователь с нетерпением ожидает, когда же наконец завершится загрузка данных и на экране его компьютера наконец появится что-то помимо анимации загрузки. Созданные компоненты могут быть с легкостью изменены и использованы заново в новых проектах. Высокий процент переиспользования кода повышает покрываемость тестами, что, в свою очередь, приводит к более высокому уровню контроля качества. Используя [React Native](https://facebook.github.io/react-native/) мобильные приложения для Android и iOS, используя опыт JavaScript и React разработки.

* Virtual DOM может повысить производительность высоконагруженных приложений, что может снизить вероятность возникновения возможных неудобств и улучшает пользовательский опыт;
* Использование изоморфного подхода помогает производить рендеринг страниц быстрее, тем самым позволяя пользователям чувствовать себя более комфортно во время работы с вашим приложением. Поисковые системы индексируют такие страницы лучше. Поскольку один и тот же код может быть использован как в клиентской, так и в серверной части приложения, нет необходимости в дублировании одного и того же функционала. В результате время разработки и затраты снижаются;
* Благодаря переиспользованию кода стало гораздо проще создавать мобильные приложения. Код, который был написан во время создания сайта, может быть снова использован для создания мобильного приложения. Если вы планируете использовать не только сайт, но и мобильное приложение, нет необходимости нанимать две большие команды разработчиков.

TypeScript отличается от JavaScript возможностью [явного](https://ru.wikipedia.org/wiki/Явная_типизация)[статического](https://ru.wikipedia.org/wiki/Статическая_типизация)назначения типов, поддержкой использования полноценных классов (как в традиционных [объектно-ориентированных](https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование)языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, [рефакторинг](https://ru.wikipedia.org/wiki/Рефакторинг)и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ.

Механизм локального хранилища компонента, который поставляется вместе с базовой библиотекой (React) неудобен тем, что такое хранилище изолировано. К примеру, если вы хотите, чтобы разные независимые компоненты реагировали на какое-либо событие, вам придётся либо передавать локальное состояние в виде пропсов дочерним компонентам, либо поднимать его вверх до ближайшего родительского компонента. В обоих случаях делать это не удобно. Код становится более грязным, трудночитаемым, а компоненты зависимыми от их вложенности. Redux снимает эту проблему так как всё состояние доступно всем компонентом без особых трудностей.Redux является универсальным средством разработки и может быть использован в связке с различными библиотеками и фреймворками. В этой же статье будет рассматривается использование Redux в React приложениях.

## 4 **разработка архитектуры программного средства**

* 1. Выбор технологий:

Доступ к данным осуществляется с использованием архитектуры, состоящей из трех слоев. UI, BLL, DAL уровни.

UI (User Interface)

В области промышленного дизайна взаимодействия человека с компьютером пользовательский интерфейс — это пространство, в котором происходит взаимодействие между людьми и машинами.

В данном случае в роли UI используется React

BLL (Business Logic Layer)

Такие API называются системой удаленного вызова процедур. Клиент выполняет функцию (или процедуру) на сервере, и сервер отправляет результат обратно клиенту.

Websocket API

Websocket API – это еще одна современная разработка web API, которая использует объекты JSON для передачи данных. WebSocket API поддерживает двустороннюю связь между клиентскими приложениями и сервером. Сервер может отправлять сообщения обратного вызова подключенным клиентам, что делает его более эффективным, чем REST API.

REST API

На сегодняшний день это самые популярные и гибкие API-интерфейсы в Интернете. Клиент отправляет запросы на сервер в виде данных. Сервер использует этот клиентский ввод для запуска внутренних функций и возвращает выходные данные обратно клиенту. Давайте рассмотрим API REST более подробно ниже.

Каковы преимущества REST API?

REST API имеет четыре главных преимущества.

1. Интеграция

API используются для интеграции новых приложений с существующими программными системами. Это увеличивает скорость разработки, потому что каждую функцию не нужно писать с нуля. API можно использовать для усиления существующего кода.

2. Инновации

Целые отрасли могут измениться с появлением нового приложения. Компании должны быстро реагировать и поддерживать быстрое развертывание инновационных услуг. Они могут сделать это, внося изменения на уровне API без необходимости переписывать весь код.

3. Расширение

API-интерфейсы предоставляют компаниям уникальную возможность удовлетворять потребности своих клиентов на разных платформах. Например, карты API позволяет интегрировать информацию о картах через веб-сайты, Android, iOS и т. д. Любая компания может предоставить аналогичный доступ к своим внутренним базам данных, используя бесплатные или платные API.

4. Простота обслуживания

API действует как шлюз между двумя системами. Каждая система обязана вносить внутренние изменения, чтобы это не повлияло на API. Таким образом, любые будущие изменения кода одной стороной не повлияют на другую сторону.

Какие типы API существую?

API классифицируются как по архитектуре, так и по сфере применения. Мы уже рассмотрели основные типы архитектур API, поэтому мы предлагаем рассмотреть сферы применения.

Частные API

Это внутренние API организаций, которые используются только для соединения систем и данных внутри бизнеса.

Общедоступные API

Это API с общим доступом и могут быть использованы кем угодно. С этими типами API может быть (но не обязательно) сопряжена некоторая авторизация и стоимость.

Партнерские API

Эти API доступны только авторизованным сторонним разработчикам для содействия партнерским отношениям между предприятиями.

Составные API

Эти API объединяют два или более разных API для решения сложных системных требований или поведения.

Что такое адрес API и почему он важен?

Адреса API – это конечные точки взаимодействия в системе связи API. К ним относятся URL-адреса серверов, службы и другие конкретные цифровые местоположения, откуда информация отправляется и принимается между системами. Адреса API имеют решающее значение для предприятий по двум основным причинам.

1. Безопасность

Адреса API делают систему уязвимой для атак. Мониторинг API имеет решающее значение для предотвращения ненадлежащего использования.

2. Производительность

Адреса API, особенно с высоким трафиком, могут создавать узкие места и влиять на производительность системы.

Как основной framework к Python был выбрал Django Rest Framework.

Django Rest Framework (DRF) — это библиотека, которая работает со стандартными моделями Django для создания гибкого и мощного API для проекта. Эта статья поможет понять структуру DRF и дать вводные данные для начала его использования

API DRF состоит из 3-х слоев: сериализатора, вида и маршрутизатора.

Сериализатор: преобразует информацию, хранящуюся в базе данных и определенную с помощью моделей Django, в формат, который легко и эффективно передается через API.

Вид (ViewSet): определяет функции (чтение, создание, обновление, удаление), которые будут доступны через API.

Маршрутизатор: определяет URL-адреса, которые будут предоставлять доступ к каждому виду.

Сериализаторы

Модели Django интуитивно представляют данные, хранящиеся в базе, но API должен передавать информацию в менее сложной структуре. Хотя данные будут представлены как экземпляры классов Model, их необходимо перевести в формат JSON для передачи через API.

Сериализатор DRF производит это преобразование. Когда пользователь передает информацию (например, создание нового экземпляра) через API, сериализатор берет данные, проверяет их и преобразует в нечто, что Django может сложить в экземпляр модели. Аналогичным образом, когда пользователь обращается к информации через API, соответствующие экземпляры передаются в сериализатор, который преобразовывает их в формат, который может быть легко передан пользователю как JSON.

Виды (ViewSets)

Сериализатор анализирует информацию в обоих направлениях (чтение и запись), тогда как ViewSet - это тот код, в котором определены доступные операции. Наиболее распространенным ViewSet является ModelViewSet, который имеет следующие встроенные операции:

• Создание экземпляра: create ()

• Получение / чтение экземпляра: retrieve ()

• Обновление экземпляра (все или только выбранные поля): update () или partial\_update ()

• Уничтожение / Удаление экземпляра: destroy ()

• Список экземпляров (с разбивкой по страницам по умолчанию): list ()

Каждая из этих функций может быть переопределена, если требуется другое поведение

Маршрутизаторы (роутеры)

И наконец, маршрутизаторы: они предоставляют верхний уровень API. Чтобы избежать создания бесконечных URL-адресов вида: «списки», «детали» и «изменить», маршрутизаторы DRF объединяют все URL-адреса, необходимые для данного вида в одну строку для каждого ViewSet.

Базой данных для хранения пользовательских данных был выбран PostgreSQL.

PostgreSQL — это объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных.

Ранние версии системы были основаны на старой программе POSTGRES University, созданной университетом Беркли: так появилось название PostgreSQL. И сейчас СУБД иногда называют «Постгрес». Существуют сокращения PSQL и PgSQL — они тоже обозначают PostgreSQL.

СУБД позволяет гибко управлять базами данных (БД). С ее помощью можно создавать, модифицировать или удалять записи, отправлять транзакцию — набор из нескольких последовательных запросов на особом языке запросов SQL.

Кто пользуется PostgreSQL

• Бэкенд-разработчики, которым приходится взаимодействовать с базами данных при работе с «серверной» частью сайта.

• Администраторы и разработчики баз данных — специалисты, основная задача которых заключается в обслуживании и поддержке работоспособности базы.

• DevOps-инженеры, обслуживающие рабочую инфраструктуру проекта.

Для чего нужна PostgreSQL

• Гибкий доступ к базам данных, их организация и хранение.

• Управление записями в базах данных: создание, редактирование и удаление, обновление версий и так далее.

• Просмотр нужной информации из базы по запросу, например, для ее отправки на сайт или в интерфейс приложения.

• Отправка транзакций, последовательных запросов, собранных в подобие скрипта.

• Настройка и контроль доступа к той или иной информации, группировка пользователей по уровню прав.

• Контроль версий и организация одновременного доступа к базе из разных источников так, чтобы предотвратить сбои.

• Защита информации от возможных утечек и потерь.

• Контроль состояния базы в целом.

* 1. Архитектурная диаграмма

Архитектурная диаграмма — диаграмма, описывающая архитектуру приложения и взаимодействие пользователей/сервисов друг с другом.

Общая диаграмма программы и коммуникации пользователя и сервисов предоставлена на рисунке 4.1.

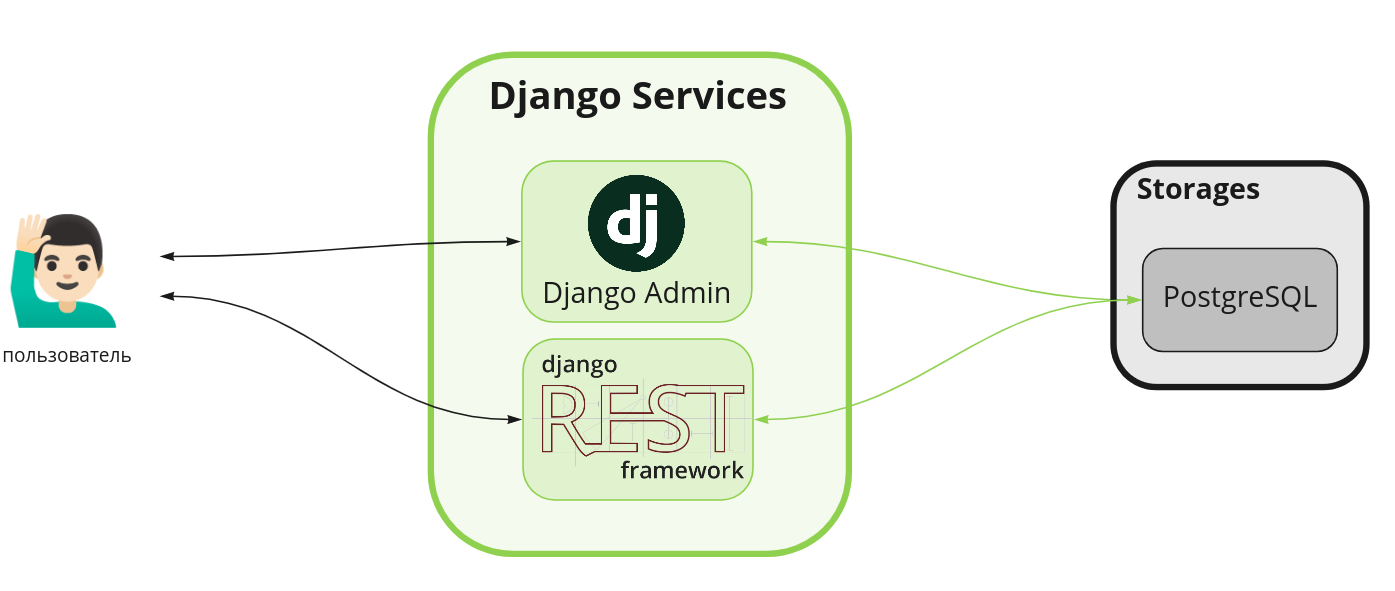


Рисунок 4.1 – Общая диаграмма приложения

Диаграмма взаимодействия пользователя через административную панель предоставлена на рисунке 4.2.

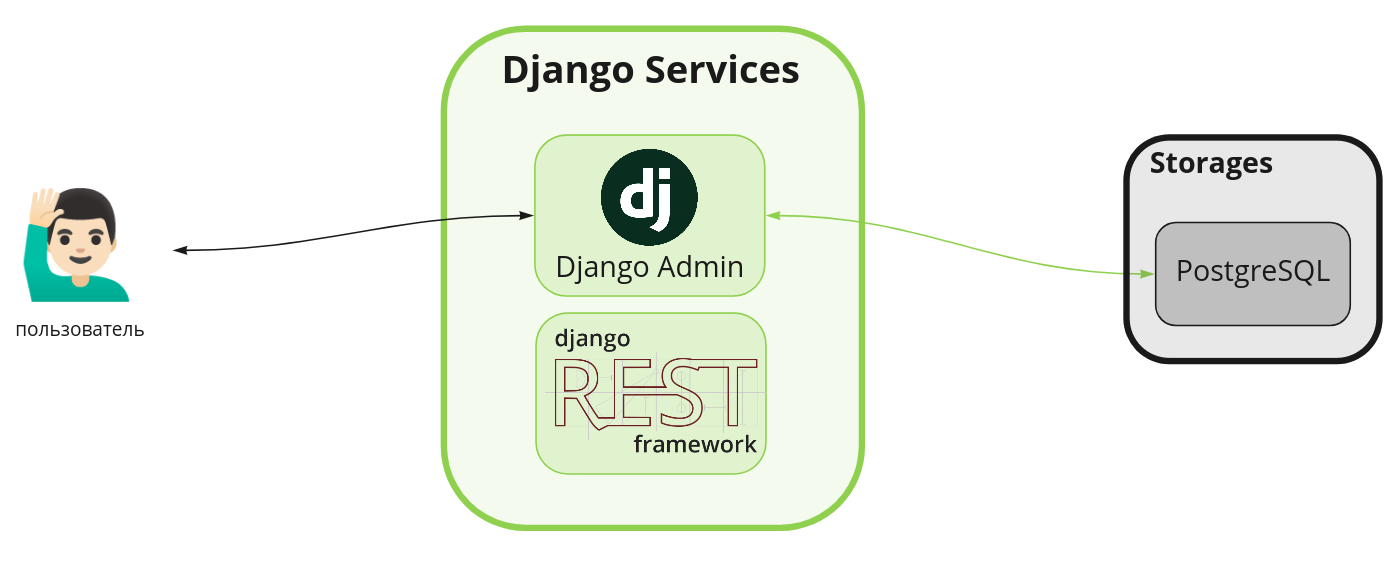
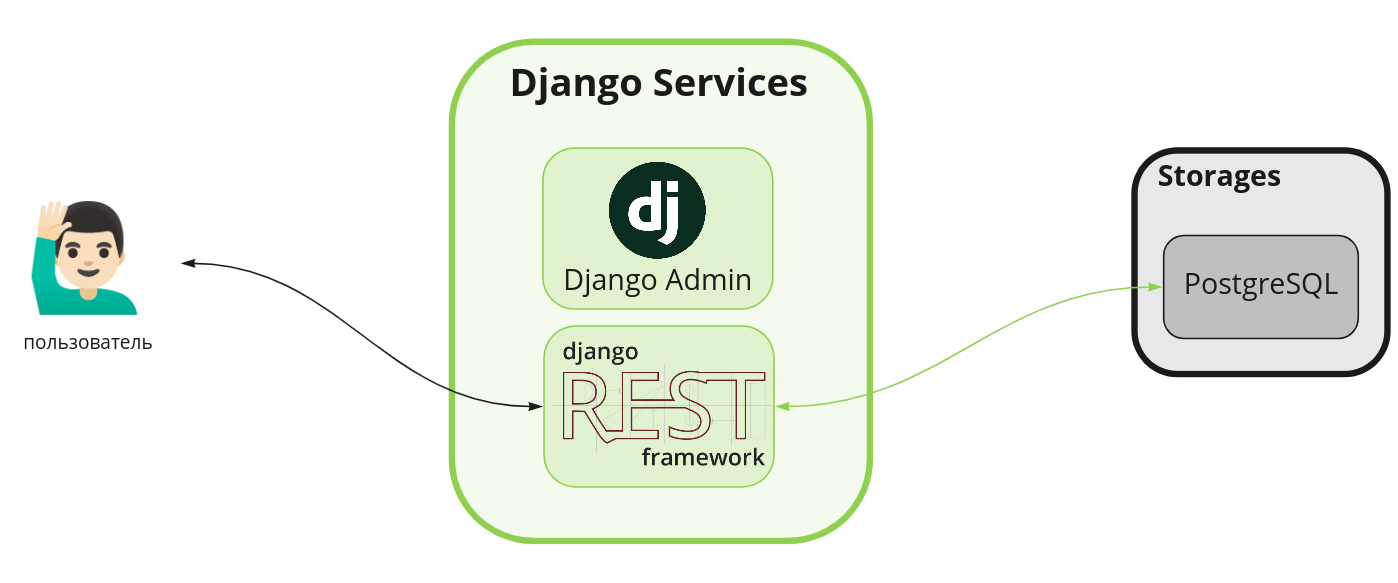
Рисунок 4.2 – Диаграмма взаимодействия через административную панель

Диаграмма взаимодействия пользователя через REST API предоставлена на рисунке 4.3.

Рисунок 4.3 – Диаграмма взаимодействия через REST API

## 5 Проектирование интерфейса

API – Application Programming Interface, что значит программный интерфейс приложения. В контексте API слово «приложение» относится к любому ПО с определенной функцией. Интерфейс можно рассматривать как сервисный контракт между двумя приложениями. Этот контракт определяет, как они взаимодействуют друг с другом, используя запросы и ответы. Документация API содержит информацию о том, как разработчики должны структурировать эти запросы и ответы.

API – это механизмы, которые позволяют двум программным компонентам взаимодействовать друг с другом, используя набор определений и протоколов. Например, система ПО метеослужбы содержит ежедневные данные о погоде. Приложение погоды на телефоне «общается» с этой системой через API и показывает ежедневные обновления погоды на телефоне.

REST – это Representational State Transfer, т. е. передача репрезентативного состояния. REST определяет набор функций, таких как GET, PUT, DELETE и т. д., которые клиенты могут использовать для доступа к данным сервера. Клиенты и серверы обмениваются данными по протоколу HTTP.

Главной особенностью REST API является то, что такая передача выполняется без сохранения состояния. Без сохранения состояния означает, что серверы не сохраняют клиентские данные между запросами. Клиентские запросы к серверу аналогичны URL-адресам, которые вы вводите в браузере для посещения веб-сайта. Ответ от сервера представляет собой простые данные без типичного графического отображения веб-страницы.

Пользовательский интерфейс представляет собой совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя и вычислительной системы.

ГОСТ «Эргономика взаимодействия человек-система», введенный в 2012 г., определяет пользовательский интерфейс (ПИ) как «компоненты интерактивной системы, предоставляющие пользователю информацию и являющиеся инструментами управления для выполнения определенных задач».

Проектирование пользовательского интерфейса – это создание тестовой версии приложения. Это начальный этап разработки пользовательского интерфейса, когда распределяются функции приложения по экранам, определяются макеты экранов, содержимое, элементы управления и их поведение.

Пользователь при обращении с интерфейсом должен представить себе, какая информация о выполняемой задаче у него существует, и в каком состоянии находятся средства, с помощью которых он будет решать данную задачу. Эффективность работы пользователя и его интерес обеспечивает правильно сформулированная методика разработки и проектирования пользовательского интерфейса.

Именно поэтому необходимо большое внимание уделять процессу построения пользовательских интерфейсов (UI) и выстраи- 5 ванию пользовательского опыта в целом (UX). Проектирование UI – это не разовая фаза проекта, это непрерывный итерационный процесс, в который вовлечены бизнес-пользователи, UX-инженеры, дизайнеры и программисты. UI (User Interface – дословно «пользовательский интерфейс») – то, как выглядит интерфейс, и то, какие физические характеристики приобретает. Определяет, какого цвета будет ваше «изделие», удобно ли будет человеку попадать пальцем на кнопки, читабельным ли будет текст и тому подобное.

UI – это дизайн визуальной составляющей интерфейса. UI-дизайнер делает графическую и текстовую информацию приятной и привлекательной, тем самым создает эмоциональную связь пользователя с интерфейсом.

UX (User Experience – дословно: «опыт пользователя») – это то, какой опыт/впечатление получает пользователь от работы с вашим интерфейсом. Удается ли ему достичь цели и насколько просто или сложно это сделать. UX-проектирование пользовательского интерфейса – проектирование взаимодействия пользователя с интерфейсом. UX-дизайнер разрабатывает сценарии того, как и каких целей может достигать пользователь при взаимодействии с цифровым продуктом.

UX/UI-дизайн – это проектирование любых пользовательских интерфейсов, в которых удобство использования так же важно, как и внешний вид. UX и UI неразрывно связаны между собой, и являются неотъемлемой частью успешного цифрового продукта.

Визуально привлекательный и удобный пользовательский интерфейс – ключевой показатель качества сайта. В сочетании с грамотной структурой и логичной навигацией по разделам ресурса, он привлекает посетителей и улучшает функциональность сайта. Главная задача в такой работе, как проектирование интерфейсов, – максимально упростить жизнь пользователю, сделать так, чтобы он достигал желаемый результат, затрачивая минимум усилий.

Пользовательский интерфейс (UI – англ. user interface – пользовательский интерфейс) представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными устройствами и аппаратурой.

Иными словами, это тот набор кнопок, ссылок, форм, диалоговых окон, иконок, пиктограмм, баннеров, ползунков и лент прокрутки, с помощью которого пользователь управляет продуктом. Интерфейс – только половина во взаимодействии с системой, другая половина – человек, пользователь. Для хорошей работы интерфейса нужно точно знать, что именно в любой конкретный момент пользователь воспринимает в интерфейсе, о чем думает, чего хочет добиться.

Интерфейсы являются основой взаимодействия всех современных информационных систем. Если интерфейс какого-либо объекта (персонального компьютера, программы, функции) не изменяется (стабилен, стандартизирован), это дает возможность модифицировать сам объект, не перестраивая принципы его взаимодействия с другими объектами.

## Реализация программного средства

### 6.1 Требования к программному продукту

В состав технических средств должен входить персональный компьютер, включающий в себя:

* процессор с частотой 1.8 ГГц или более;
* оперативную память объемом не менее 10 Гб;
* монитор с минимальным разрешением 1280х720 пикселей;
* клавиатуру или мышь.

Требование к информационной и программной совместимости:

* операционная система «Windows 10» и выше, либо операционная система на основе ядра «Linux» для пользователей;
* операционная система на основе ядра «Linux» для развертывания приложения.

### 6.2 Руководство пользователя

При входе в административную панель пользователя встречает окно атворизации, которое содержит в себе два поля: email и пароль. Для входа пользователь должен быть добавлен в базу данных администратором. Пример показан на рисунке 5.1.

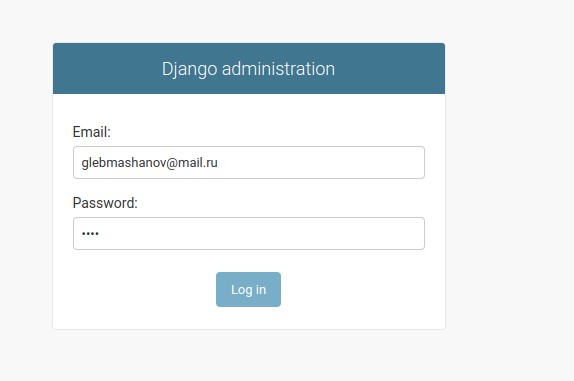
****

Рисунок 5.1 – Интерфейс окна авторизации

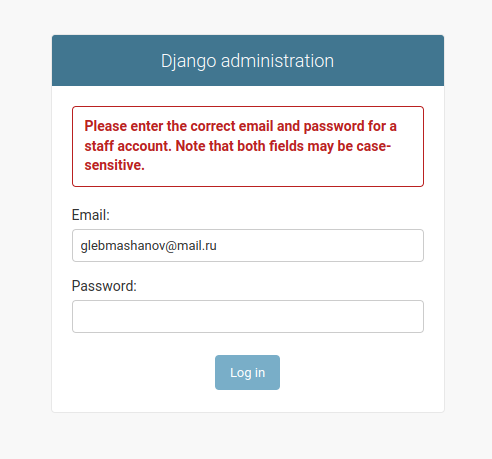
При вводе некорректных данных будет отображено сообщение об ошибке. Пример показан на рисунке 5.2.

Рисунок 5.2 – Интерфейс окна с некорректно введёнными данными

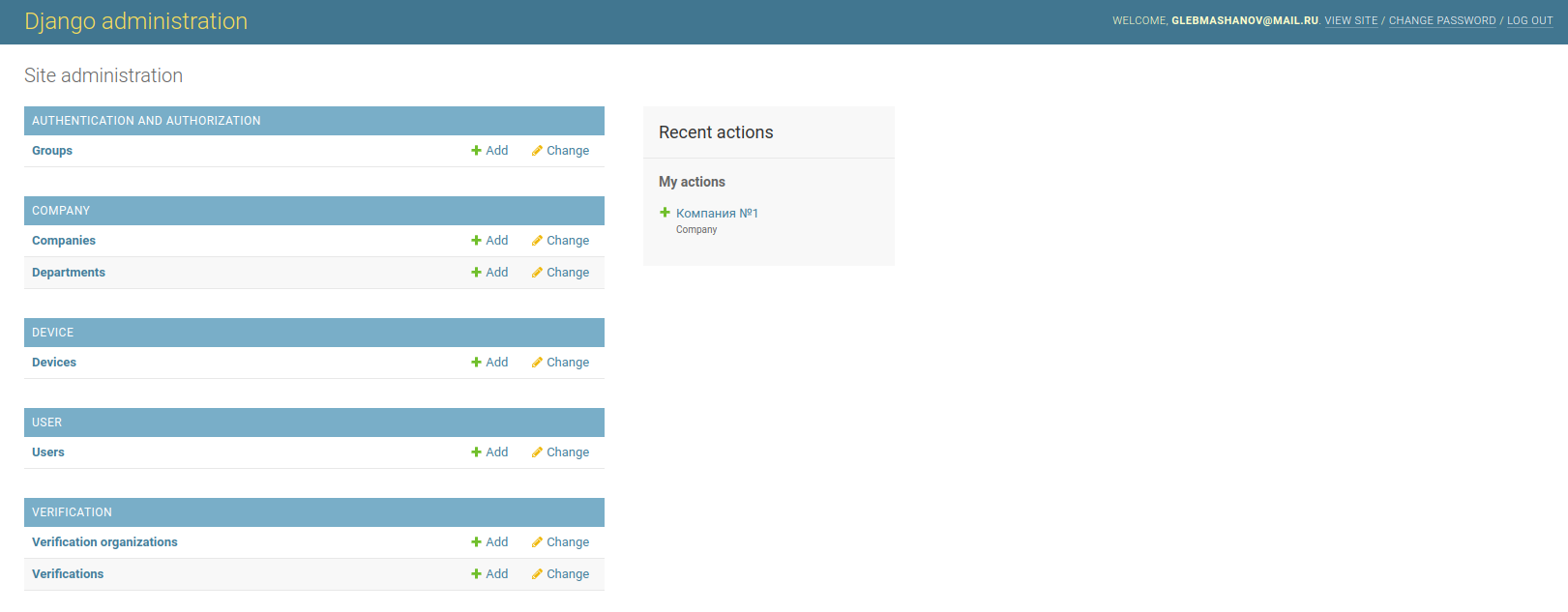
При успешно введённых данных будет отображено главное окно административной панель. Пример показан на рисунке 5.3.

Рисунок 5.3 – Интерфейс главного окна приложения

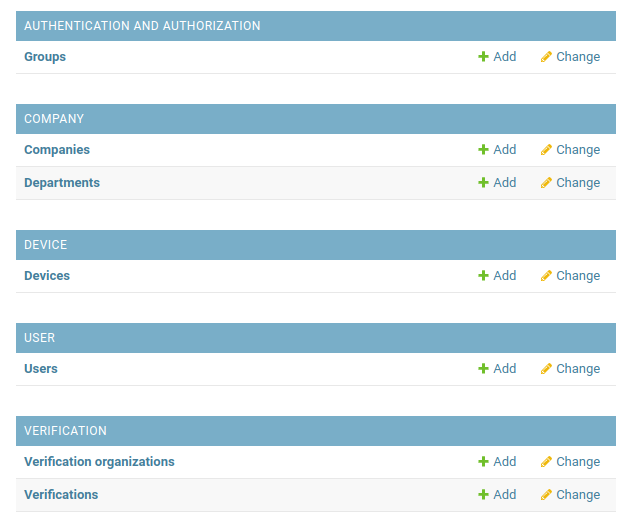
Слева предоставлена панель с выборов сущности для их просмотра, добавления, либо изменения. Пример показан на рисунке 5.4

Рисунок 5.4 – Панель с общим функционалом

При нажатии на кнопку «Add» где появятся поля для создания новой сущности. Пример показан на рисунке 5.5

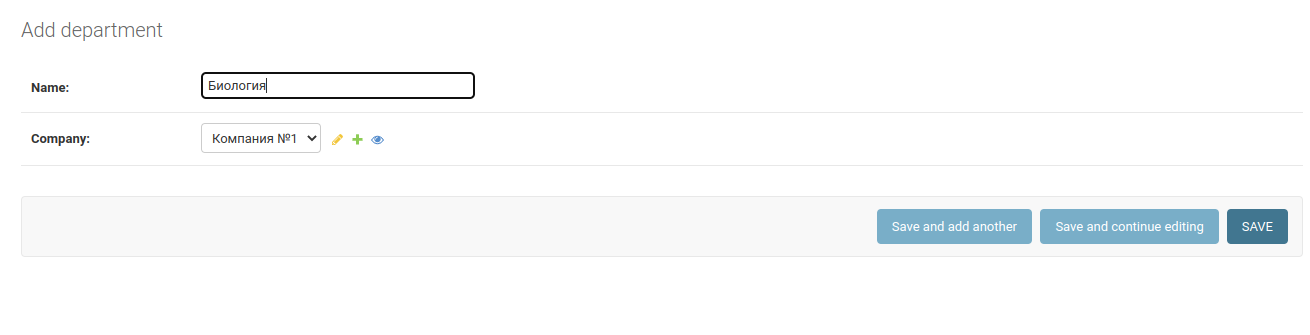


Рисунок 5.5 – Добавление отдела

При нажатии кнопки «Save» откроется окно со списком отделов. Пример показан на рисунке 5.6

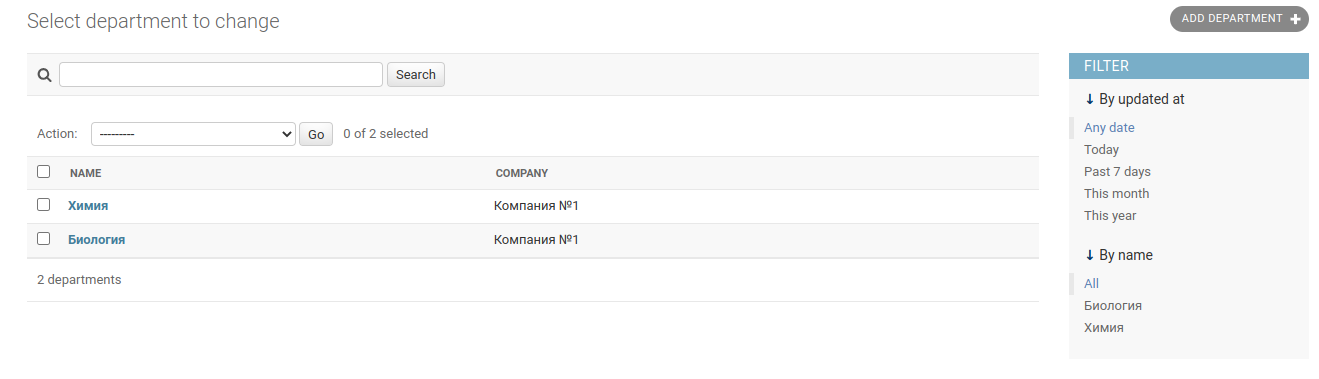


Рисунок 5.6 – Список отделов

По выбору «Device» отобразится список устройств с их текущим статусом. Пример показан на рисунке 5.7

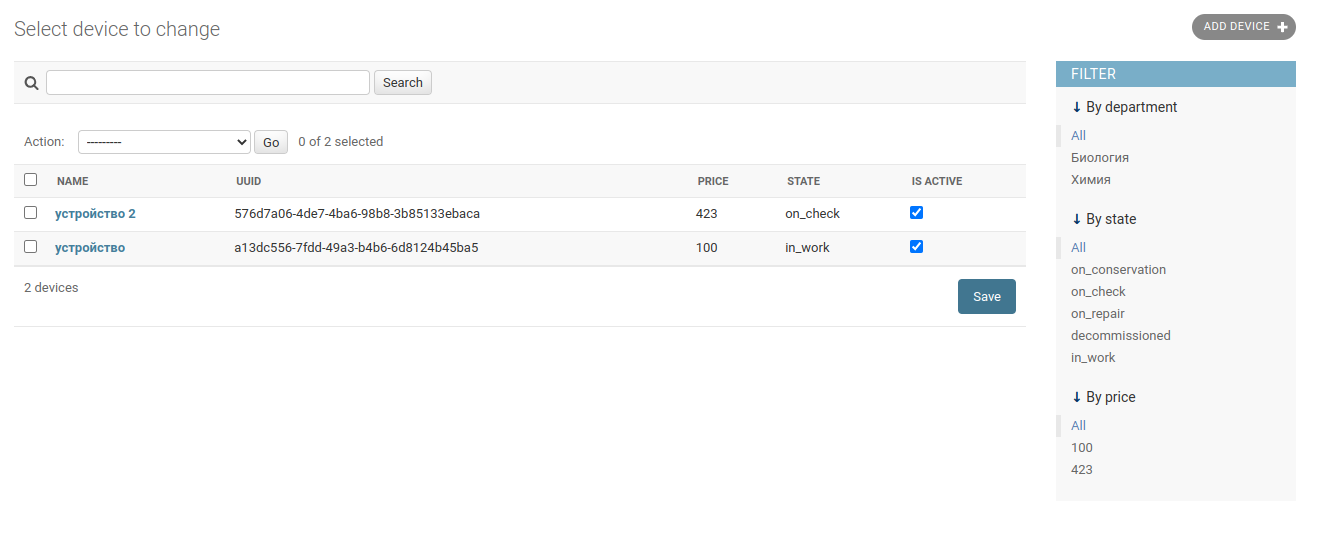


Рисунок 5.7 – Список устройств

После выбора устройства чекбоксом, действия удаления и кнопки «Save» - выбранное устройство удалится. По нажатию на кнопку «Add» откроется окно добавления нового девайса. Пример показан на рисунке 5.8

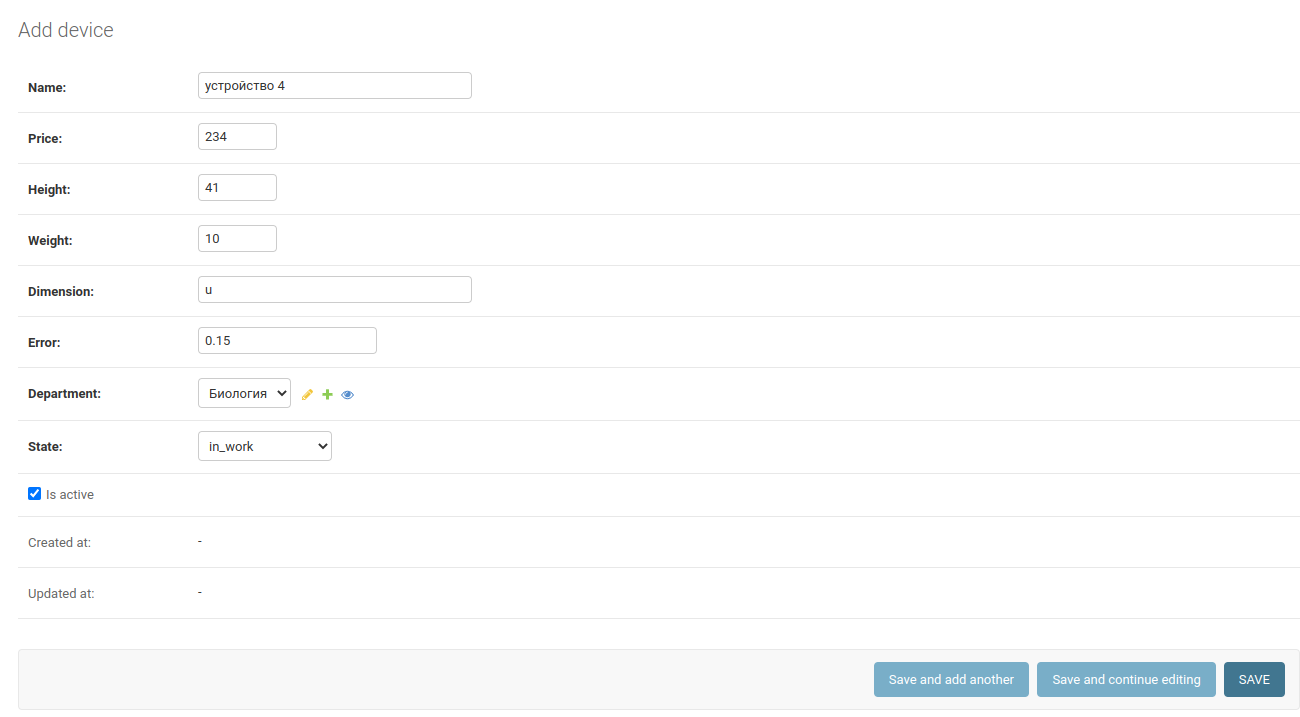
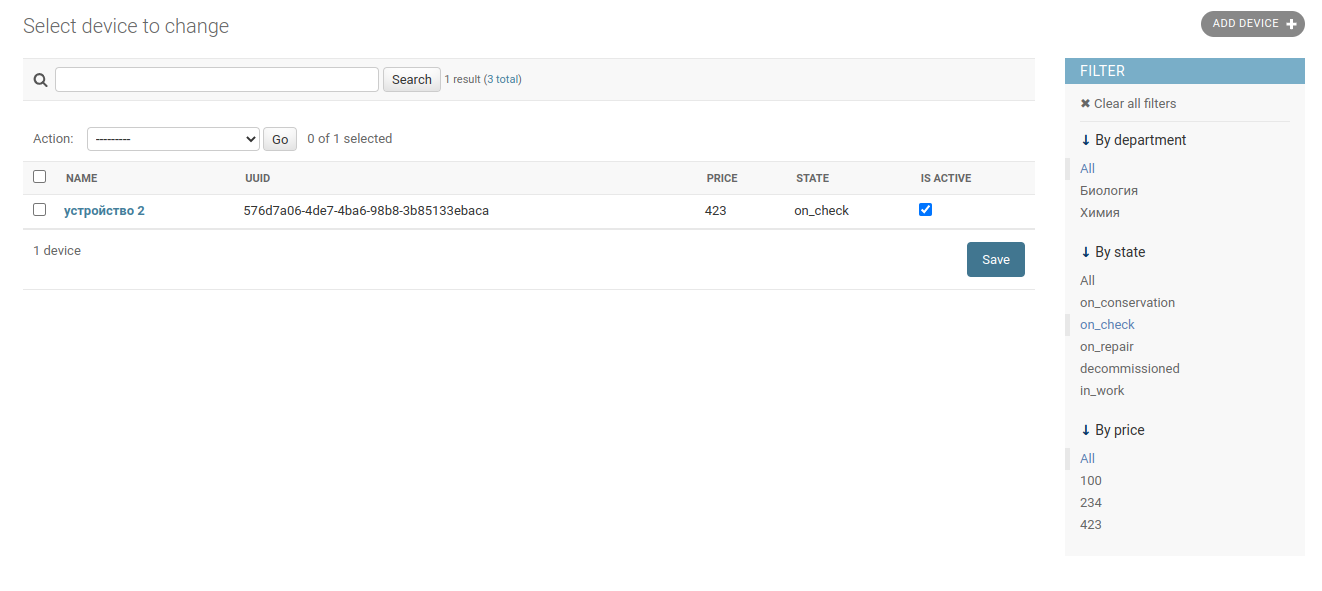


Рисунок 5.8 – Добавление устройства

При нажатии кнопки «Save» откроется окно со списком устройств. Справа можно отфильтровать девайсы по их статусу. Пример фильтрации по статусу показан на рисунке 5.9

Рисунок 5.9 – Фильтрация устройств

Также можно отправить определённое устройство на проверку. Пример показан на рисунке 5.10

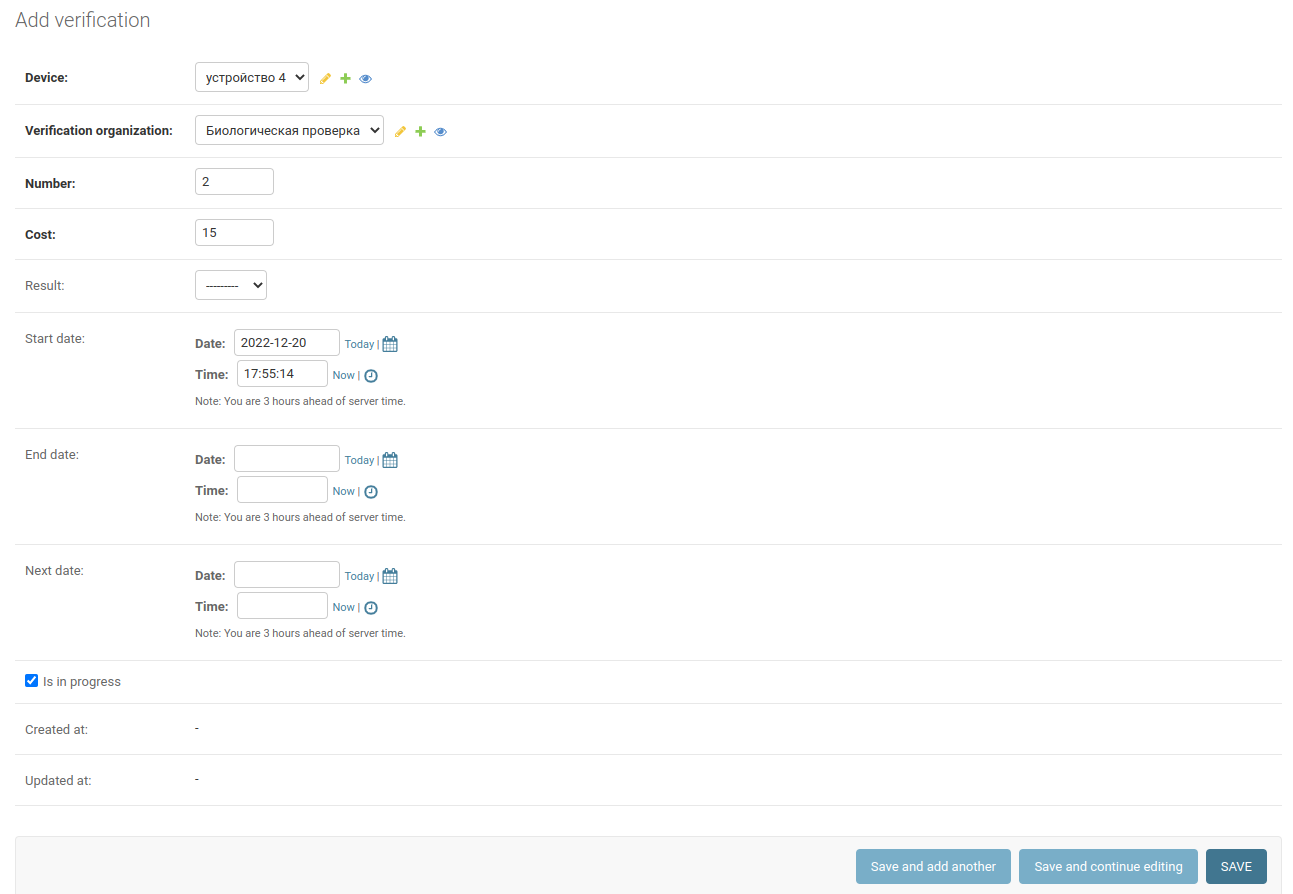


Рисунок 5.10 – Отправление устройства на проверку

Также можно изменить статус проверки. Пример показан на рисунке 5.11

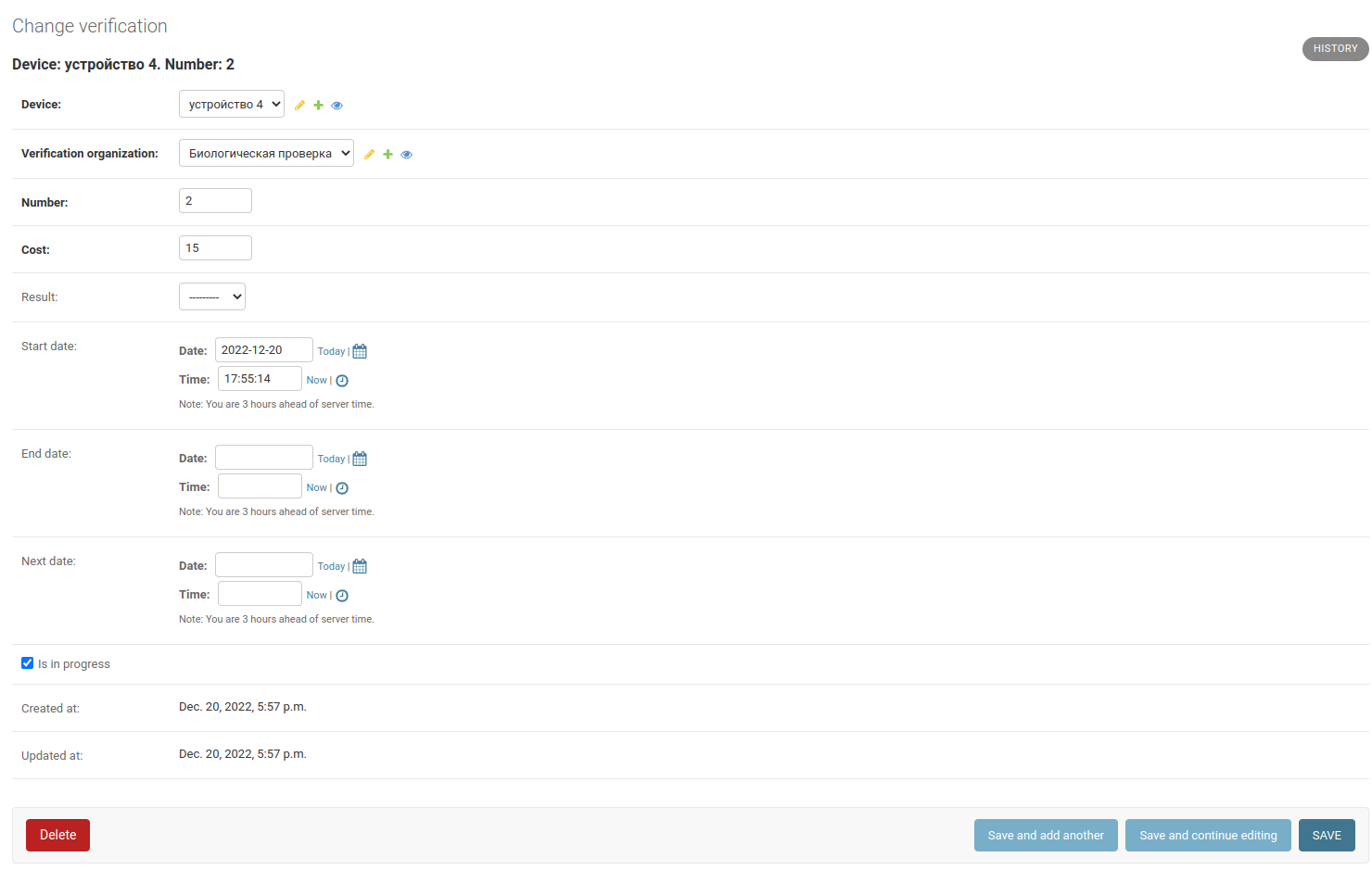


Рисунок 5.11 – Изменение статуса проверки устройства

Для отображения устройств с датой следующей проверки меньше 7 дней можно отфильтровать во вкладке с фильтрами. Пример фильтрации показан на рисунке 5.12

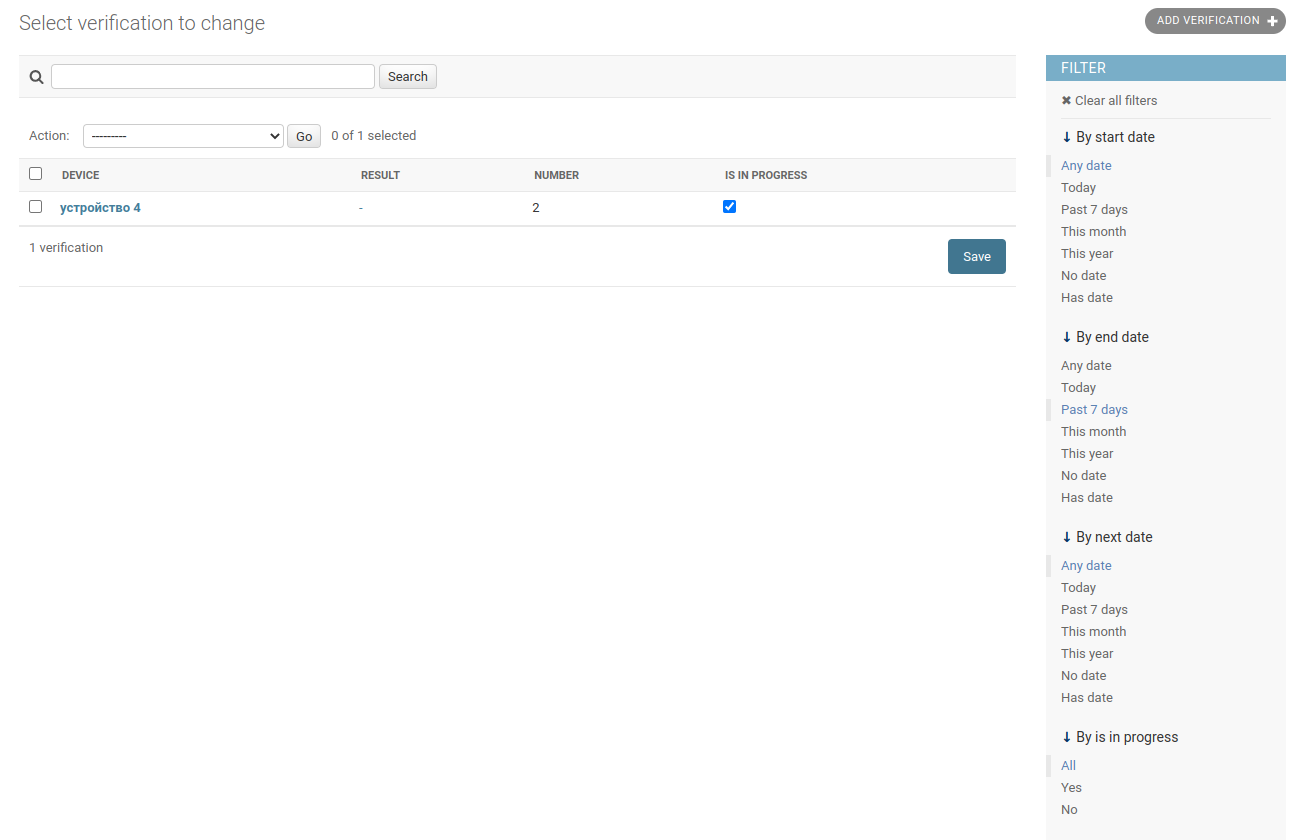


Рисунок 5.12 – Фильтрация проверок

### 6.3 Тестирование и отладка

Тестирование и отладка – два противоположных, но взаимосвязанных процесса, выполняются многократно.

Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта.

Отладка – это деятельность, направленная на обнаружение и исправление ошибок в программном продукте с использованием процессов выполнения его программ. Таким образом, отладку можно представить в виде многократного повторения трех процессов: тестирования, в результате которого может быть констатировано наличие в программном продукте ошибки, поиска места ошибки в программах и документации программного продукта и редактирования программ и документации с целью устранения обнаруженной ошибки [13].

Отладку программы начинают с составления плана тестирования. Такой план должен представлять себе любой программист. Составление плана опирается на понятие об источниках и характере ошибок. Основными источниками ошибок являются недостаточно глубокая проработка математической модели или алгоритма решения задачи; нарушение соответствия между схемой алгоритма или записью его на алгоритмическом языке и программой, записанной на языке программирования; неверное представление исходных данных на программном бланке; невнимательность при наборе программы и исходных данных на клавиатуре устройства ввода.

Нарушение соответствия между детально разработанной записью алгоритма в процессе кодирования программы относится к ошибкам, проходящим вследствие невнимательности программиста. Отключение внимания приводит и ко всем остальным ошибкам, возникающим в процессе подготовки исходных данных и ввода программы в ЭВМ. Ошибки, возникающие вследствие невнимательности, могут иметь непредсказуемые последствия, так как наряду с потерей меток и описаний массивов, дублированием меток, нарушением баланса скобок возможны и такие ошибки, как потеря операторов, замена букв в обозначениях переменных, отсутствие определений начальных значений переменных, нарушение адресации в массивах, сдвиг исходных данных относительно полей значений, определенных спецификациями формата.

Учитывая разнообразие источников ошибок, при составлении плана тестирования классифицируют ошибки на два типа: 1 – синтаксические; 2 – семантические (смысловые).

Синтаксические ошибки – это ошибки в записи конструкций языка программирования (чисел, переменных, функций, выражений, операторов, меток, подпрограмм).

Семантические ошибки – это ошибки, связанные с неправильным содержанием действий и использованием недопустимых значений величин.

Существующие на сегодня методы тестирования программ не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки разрабатываемого мобильного приложения.

Поэтому можно выделить три основные цели тестирования:

* повысить вероятность того, что приложение, предназначенное для тестирования, будет работать правильно при любых обстоятельствах;
* повысить вероятность того, что приложение, предназначенное для тестирования, будет соответствовать всем описанным требованиям;
* предоставление актуальной информации о состоянии продукта на данный момент.

Виды тестирования:

* функциональное тестирование;
* системное тестирование;
* тестирование производительности

1. нагрузочное;
2. стресс-тестирование.

* регрессионное;
* модульное;
* тестирование безопасности;
* тестирование локализации;
* юзабилити.

При тестировании разработанного программного продукта было использовано мануальное(ручное) тестирование пользовательского интерфейса, а также проверка соответствия написанного кода на корректность работоспособности.

### 6.4 Методы и средства защиты

JSON Web Token (JWT) — это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения подлинности аккаунта.

Токен JWT состоит из трех частей: заголовка (header), полезной нагрузки (payload) и подписи или данных шифрования. Первые два элемента — это JSON объекты определенной структуры. Третий элемент вычисляется на основании первых и зависит от выбранного алгоритма (в случае использования неподписанного JWT может быть опущен). Токены могут быть перекодированы в компактное представление (JWS/JWE Compact Serialization): к заголовку и полезной нагрузке применяется алгоритм кодирования Base64-URL, после чего добавляется подпись и все три элемента разделяются точками («.»).

Cross-Origin Resource Sharing (CORS) — механизм, использующий дополнительные HTTP-заголовки, чтобы дать возможность агенту пользователя получать разрешения на доступ к выбранным ресурсам с сервера на источнике (домене), отличном от того, что сайт использует в данный момент. Говорят, что агент пользователя делает запрос с другого источника (cross-origin HTTP request), если источник текущего документа отличается от запрашиваемого ресурса доменом, протоколом или портом.

Пример cross-origin запроса: HTML страница, обслуживаемая сервером с http://domain-a.com, запрашивает <img> src по адресу http://domain-b.com/image.jpg. Сегодня многие страницы загружают ресурсы вроде CSS-стилей, изображений и скриптов с разных доменов, соответствующих разным сетям доставки контента (Content delivery networks, CDNs).

В целях безопасности браузеры ограничивают cross-origin запросы, инициируемые скриптами. Например, XMLHttpRequest и Fetch API следуют политике одного источника (same-origin policy). Это значит, что web-приложения, использующие такие API, могут запрашивать HTTP-ресурсы только с того домена, с которого были загружены, пока не будут использованы CORS-заголовки.

Механизм CORS поддерживает кросс-доменные запросы и передачу данных между браузером и web-серверами по защищённому соединению. Современные браузеры используют CORS в API-контейнерах, таких как XMLHttpRequest или Fetch, чтобы снизить риски, присущие запросам с других источников.

CSRF-token — это максимально простой и результативный способ защиты сайта от CSRF-мошенников. Он работает так: сервер создаёт случайный ключ (он же токен) и отправляет его браузеру клиента. Когда браузер запрашивает у сервера информацию, сервер, прежде чем дать ответ, требует показать ключ и проверяет его достоверность. Если токен совпадает, сессия продолжается, а если нет — прерывается. Токен действителен только одну сессию — с новой сессией он обновляется.

Чтобы получить ответ от сервера, используются разные методы запроса. Условно они делятся на две категории: те, которые не изменяют состояние сервера (GET, TRACE, HEAD), и те, которые изменяют (PUT, PATCH, POST и DELETE). Последние имеют большую CSRF-уязвимость и поэтому должны быть защищены в первую очередь.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была достигнута поставленная цель – разработан визуализатор алгоритмов сортировки.

Были осуществлены поставленные задачи:

* анализ конкурирующих программных продуктов;
* построение функциональных моделей, отражающих работу разрабатываемого программного продукта;
* разработка алгоритма работы разрабатываемого программного продукта;
* разработка интерфейса.

В процессе выполнения курсового проекта:

1. был проведен анализ существующих решений;
2. установлена актуальность и необходимость в использовании программного продукта;
3. выбрана среда разработки и язык программирования, а также обоснование этого выбора;
4. разработаны функциональные модели;
5. разработан интерфейс программного продукта;
6. разработан сам программный продукт;
7. выполнена отладка и тестирования программного продукта;
8. разработано руководство пользователя.

Разработанный программный продукт позволяет визуально рассмотреть и сравнить различные алгоритмы сортировки.

Выбранная среда разработки способствовала созданию легкого и доступного для понимания интерфейса. Во время разработки были использованный внутренние библиотеки С#, что позволило упростить алгоритмы решения поставленных задач.

## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

* Luciano Ramalho . Fluent Python - O'Reilly Media; 1st edition, 2015. 790 с.
* Django документация [Электронный ресурс].

URL: <https://docs.djangoproject.com/en/4.1/> (дата обращения: 10.12.2022)

* Django REST Framework документация [Электронный ресурс].

URL: <https://www.django-rest-framework.org/> (дата обращения: 10.12.2022)

* Python документация [Электронный ресурс].

URL: <https://docs.python.org/3/> (дата обращения: 10.12.2022)

* PostgreSQL документация [Электронный ресурс].

URL: <https://www.postgresql.org/docs/> (дата обращения: 10.12.2022)

* Docker документация [Электронный ресурс].
* URL: <https://docs.docker.com/get-started/> (дата обращения: 10.12.2022)

# ПРИЛОЖЕНИЕ А (обязательное) Текст программы