geekbrains

Программист Python Цифровые профессии

Разработка web-приложения для учета компьютерной техники на предприятии

Перемышленников Алексей Сергеевич

Барнаул

2024

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc155881806)

[ВВЕДЕНИЕ 5](#_Toc155881807)

[ГЛАВА 1. Основы разработки web-приложений с использованием Django 8](#_Toc155881808)

[1.1 Введение в Django и его преимущества 8](#_Toc155881809)

[1.2 Архитектура MVC (Model-View-Controller) и паттерн MTV (Model-Template-View) 10](#_Toc155881810)

[1.3 Основы работы с базами данных в Django (ORM) 13](#_Toc155881811)

[1.4 Создание и управление веб-приложением в Django 15](#_Toc155881812)

[ГЛАВА 2. Анализ и проектирование системы учета компьютерной техники 17](#_Toc155881813)

[2.1 Анализ требований к web-приложению для учета компьютерной техники 17](#_Toc155881814)

[2.2 Проектирование базы данных для хранения информации о компьютерной технике 20](#_Toc155881815)

[2.3 Определение функциональных возможностей приложения 22](#_Toc155881816)

[ГЛАВА 3. Разработка web-приложения с использованием Django 24](#_Toc155881817)

[3.1 Создание основных моделей данных при помощи Django ORM 24](#_Toc155881818)

[3.2 Разработка пользовательского интерфейса с использованием шаблонов (Templates) 34](#_Toc155881819)

[3.3 Реализация функциональности учета компьютерной техники: добавление, редактирование, удаление 45](#_Toc155881820)

[3.4 Обеспечение безопасности данных и аутентификации пользователей 51](#_Toc155881821)

[ГЛАВА 4. Тестирование и отладка приложения 53](#_Toc155881822)

[4.1 Планирование тестирования и выбор тестовых сценариев 53](#_Toc155881823)

[4.2 Отладка и исправление выявленных ошибок 55](#_Toc155881824)

[ГЛАВА 5. Заключение 56](#_Toc155881825)

[ПРИЛОЖЕНИЯ 58](#_Toc155881826)

[СПИСОК ЛИТЕРАТУРЫ 59](#_Toc155881828)

ВВЕДЕНИЕ

В современном мире компьютерная техника является неотъемлемой частью эффективной работы любого предприятий. Актуальность данной темы обусловлена рядом факторов, связанных с современной динамикой развития бизнеса и зависимостью от информационных технологий. Вот некоторые из них:

1. **Зависимость от информационных технологий:** Современные предприятия не могут обойтись без компьютеров и информационных систем. Компьютеры используются для организации бухгалтерии, управления производственными процессами, ведения клиентской базы данных и многих других задач. Поэтому эффективное управление компьютерной техникой становится ключевым фактором для успешной деятельности.
2. **Необходимость оптимизации ресурсов:** Компьютерная техника представляет собой значительный капитальный актив. Отсутствие эффективной системы учета и мониторинга может привести к излишним затратам, например, на покупку дополнительного оборудования, когда существующее можно было бы использовать более эффективно.
3. **Контроль над оборудованием:** Учет компьютерной техники важен для обеспечения безопасности данных и предотвращения утечек конфиденциальной информации. Неправильное использование техники или несанкционированный доступ к ней может повлечь за собой серьезные последствия.
4. **Упрощение процессов обслуживания и ремонта:** Правильный учет и мониторинг компьютеров позволяют оперативно реагировать на неисправности и проводить плановое обслуживание. Это увеличивает срок службы оборудования и минимизирует потери времени из-за простоев.
5. **Требования законодательства и стандартов:** В некоторых отраслях существуют законодательные требования к учету и безопасности информационных ресурсов. Невыполнение этих требований может привести к юридическим и финансовым последствиям.
6. **Рост объемов данных:** С ростом объемов данных, связанных с учетом компьютерной техники, становится сложнее эффективно управлять информацией вручную. Web-приложение с учетом и анализом данных может значительно упростить этот процесс.

В связи с вышеизложенными факторами, разработка web-приложения для учета компьютерной техники имеет высокую актуальность и может принести значительную пользу для организаций любого масштаба и направления деятельности.

Данный проект представляет собой разработку web-приложения на основе фреймворка Django, направленного на автоматизацию процесса учета и мониторинга компьютерной техники на предприятии. Современная организация не может обойтись без эффективной работы с информационными ресурсами, включая компьютеры и периферийные устройства. Недостаточный контроль и учет данной техники могут привести к ухудшению производительности, недопустимым затратам и сложностям в обслуживании.

Целью данного проекта является создание удобного, функционального и безопасного web-приложения, которое позволит эффективно вести учет компьютерной техники, отслеживать её статус и историю обслуживания, а также проводить анализ данных.

В ходе выполнения проекта планируется реализовать следующие этапы:

1. Исследование и анализ требований к системе учета компьютерной техники, включая изучение существующих методов учета и мониторинга. Это позволит определить основные функциональности и требования, необходимые для разработки эффективной системы учета.
2. Проектирование базы данных и структуры приложения, включая разработку схемы для хранения информации о компьютерной технике и определение структуры пользовательского интерфейса. Это обеспечит удобное взаимодействие пользователей с приложением.
3. Реализация функциональности приложения, включая создание моделей данных для техники и пользователей, механизмы добавления, редактирования и удаления записей о технике, а также возможности просмотра и анализа данных.
4. Обеспечение безопасности и аутентификации, включая внедрение механизма аутентификации пользователей для обеспечения доступа только авторизованным пользователям.
5. Тестирование и отладка, включая разработку тестовых сценариев для проверки функциональности, интерфейса и безопасности приложения.
6. Заключение и будущие перспективы, включая подведение итогов разработки, оценку эффективности системы учета и предложение дальнейших шагов по развитию и усовершенствованию приложения.

Путем разработки данного web-приложения решается проблема неэффективного учета компьютерной техники, что позволит предприятию оптимизировать процессы управления ресурсами, минимизировать издержки и обеспечить более высокую производительность.

Для реализации проекта будет использован фреймворк Django, который предоставит мощный инструментарий для разработки web-приложений, включая ORM для работы с базами данных, механизмы аутентификации и безопасности.

На данном этапе проекта я буду работать индивидуально, выполняя роли разработчика, проектировщика и тестировщика.

Таким образом, разработка данного web-приложения имеет актуальную проблему и важное практическое значение для организаций, ориентированных на эффективное использование информационных ресурсов.

ГЛАВА 1. Основы разработки web-приложений с использованием Django

# Введение в Django и его преимущества

Для реализации поставленной цели и задач работы был выбран фреймворк Django. Django представляет собой мощный инструмент для разработки web-приложений на языке Python. Вот некоторые ключевые аспекты и значимость выбранного инструмента для проекта:

1. **Продуктивность и эффективность разработки:** Django предоставляет набор инструментов и готовых компонентов, которые значительно ускоряют процесс создания web-приложений. Вместо написания кода с нуля для обработки рутины, такой как управление базой данных и аутентификацией пользователей, разработчик может использовать встроенные функции Django, что позволяет сосредоточиться на уникальных аспектах приложения.
2. **ORM (Object-Relational Mapping):** Django предоставляет ORM, что облегчает работу с базами данных. ORM позволяет работать с данными как с объектами Python, а не писать SQL-запросы напрямую. Это сокращает вероятность ошибок и упрощает взаимодействие с базой данных.
3. **Механизмы аутентификации и безопасности:** Безопасность данных является приоритетом в любом web-приложении. Django предоставляет встроенные механизмы аутентификации и авторизации пользователей, а также множество инструментов для защиты от потенциальных угроз, таких как CSRF-атаки (межсайтовая подделка запроса) и инъекции SQL.
4. **Шаблонизация и представления:** Django использует систему шаблонов, которая позволяет разделить логику и представление. Это способствует созданию чистого и удобочитаемого кода, что важно для поддержки и развития приложения в будущем.
5. **Расширяемость и плагины:** Django обладает обширной документацией и активным сообществом разработчиков. Это позволяет легко находить решения для специфических задач, а также использовать готовые пакеты и плагины для расширения функциональности приложения.
6. **Административная панель:** Django предоставляет встроенную административную панель, которая позволяет управлять данными в базе данных без необходимости написания специфического кода. Это значительно облегчает процесс управления и мониторинга компьютерной техники.
7. **Соответствие современным стандартам:** Django активно поддерживается и развивается, что обеспечивает совместимость с последними стандартами и технологиями web-разработки.

В целом, выбор фреймворка Django для данного проекта оправдан его возможностями в области быстрой разработки, безопасности и управления данными. Он позволит значительно ускорить процесс создания функционального и надежного web-приложения для учета компьютерной техники на предприятии.

# Архитектура MVC (Model-View-Controller) и паттерн MTV (Model-Template-View)

Архитектурные паттерны MVC (Model-View-Controller) и MTV (Model-Template-View) являются основой для организации кода во фреймворке Django. В то время как MVC широко используется во многих других фреймворках, Django вводит свою вариацию на этот паттерн под названием MTV. Вот как они работают:

MVC (Model-View-Controller):

**Модель (Model):** отвечает за обработку данных и логику бизнес-приложения. Это может быть класс, представляющий таблицу в базе данных или другие структуры данных.

**Представление (View):** отвечает за отображение данных пользователю. Оно получает данные от модели и решает, как их представить в виде HTML, JSON или других форматов.

**Контроллер (Controller):** осуществляет взаимодействие между моделью и представлением. Обрабатывает пользовательский ввод, вызывает соответствующие методы модели и выбирает подходящее представление.

MTV (Model-Template-View):

**Модель (Model):** Основной компонент для работы с данными. Модель описывает структуру данных и правила их хранения и манипулирования. В Django это напрямую связано с базой данных и обработкой данных.

**Шаблон (Template):** отвечает за отображение данных в виде, понятном для пользователей. Шаблоны Django используют специфический синтаксис, который позволяет интегрировать данные в HTML-код.

**Представление (View):** отвечает за обработку запросов пользователя и взаимодействие с моделью и шаблонами. Возвращает пользователю результирующую HTML-страницу.

Сравнение:

В MVC контроллер более активно управляет потоком данных между моделью и представлением, в то время как в MTV представление играет более активную роль.

В MTV, представление и шаблоны работают вместе для генерации финальной HTML-страницы, тогда как в MVC представление и представленные данные отделены друг от друга.

В контексте Django, MTV близок к реализации MVC, но с упором на более тесное взаимодействие между представлениями и шаблонами. Это помогает разделить логику отображения и представления данных, что облегчает разработку и поддержание приложений.

В контексте данной работы, использование архитектурного паттерна MTV (Model-Template-View) предоставляет значимые преимущества и соответствует целям разработки web-приложения для учета компьютерной техники на предприятии.

Преимущества применения паттерна MTV в данной работе:

**Разделение логики и отображения:** Модель (Model) позволяет описать структуру данных и обеспечить их целостность, что важно для учета компьютерной техники. Шаблоны (Template) позволяют эффективно формировать пользовательский интерфейс, а представления (View) обеспечивают связь между данными и отображением, что способствует удобочитаемости кода и его поддержке.

**Ускорение разработки:** Встроенные функции Django для работы с моделью и шаблонами сокращают время разработки и позволяют сосредоточиться на бизнес-логике. Это особенно важно при разработке приложения для учета и мониторинга, где скорость внедрения играет ключевую роль.

**Безопасность и аутентификация:** Паттерн MTV включает в себя готовые механизмы аутентификации и безопасности, что облегчает обеспечение конфиденциальности данных и предотвращение несанкционированного доступа.

**Гибкость и масштабируемость:** Разделение компонентов приложения по MTV позволяет легко вносить изменения в каждую из частей без влияния на остальные. Это упрощает масштабирование приложения в будущем.

**Поддерживаемость и расширяемость**: Разделение на модель, шаблоны и представления способствует созданию чистого и структурированного кода, который легко поддерживать и расширять.

# 1.3 Основы работы с базами данных в Django (ORM)

Одним из ключевых аспектов разработки web-приложений является эффективное взаимодействие с базами данных. В рамках разрабатываемого web-приложения для учета компьютерной техники на предприятии, важной составляющей становится выбор инструмента для работы с данными. В этом контексте, фреймворк Django предоставляет мощное средство для взаимодействия с базой данных - Object-Relational Mapping (ORM).

1. Определение моделей

Центральным элементом ORM в Django является определение моделей. Модель — это класс Python, который описывает структуру и атрибуты данных, которые будут храниться в базе данных. Определяя поля модели, определяем структуру таблицы в базе данных. Такой подход позволяет абстрагировать базу данных от кода приложения и работать с данными в объектно-ориентированной парадигме.

1. Создание миграций

Для внесения изменений в структуру базы данных, определенную моделями, в Django используется концепция миграций. Миграции представляют собой скрипты, которые автоматически применяют изменения к базе данных, обеспечивая синхронизацию с изменениями в моделях. Создание миграции после внесения изменений и последующее их применение к базе данных позволяют поддерживать целостность данных и структуры.

1. CRUD-операции и QuerySets

Основная функциональность работы с базой данных включает в себя операции CRUD - создание, чтение, обновление и удаление данных. В Django, эти операции реализованы через методы объектов моделей. Кроме того, Django ORM предоставляет мощный механизм для выполнения сложных запросов - QuerySets. QuerySets позволяют фильтровать, сортировать и агрегировать данные, предоставляя эффективные инструменты для получения требуемых данных.

1. Отношения между моделями

Организация данных в приложении может включать в себя отношения между различными моделями. Django ORM поддерживает разнообразные типы отношений, такие как ForeignKey (один-ко-многим), ManyToManyField (многие-ко-многим) и OneToOneField (один-к-одному). Эти отношения позволяют эффективно организовать данные и связи между ними.

В общем, работа с базами данных в Django через ORM предоставляет удобный и эффективный способ взаимодействия с данными. Использование моделей, миграций, административной панели и QuerySets упрощает организацию данных, обеспечивает целостность и ускоряет разработку web-приложения для учета компьютерной техники на предприятии.

# 1.4 Создание и управление веб-приложением в Django

Разработка web-приложения в Django включает не только создание моделей и работы с базой данных, но и организацию структуры приложения, маршрутизацию запросов, обработку пользовательских данных и представление результатов на веб-страницах. В этом разделе рассмотрим основные шаги по созданию и управлению веб-приложением в Django.

1. Структура проекта и приложений

Django рекомендует организовывать проекты в виде набора приложений. Приложение - это независимый компонент, который может иметь свою собственную модель, представление, шаблоны и статические файлы. Организация проекта в виде приложений обеспечивает модульность, улучшает поддерживаемость и позволяет повторно использовать компоненты.

1. Маршрутизация URL-запросов

Маршрутизация URL-запросов в Django осуществляется через файлы маршрутов (urls.py). В этих файлах определяются соответствия между URL-путями и функциями представлений. Django использует регулярные выражения для определения соответствий, что позволяет гибко настраивать маршруты.

1. Представления (Views)

Представления в Django - это функции или классы, которые обрабатывают HTTP-запросы и возвращают HTTP-ответы. Представления могут получать данные из моделей, обрабатывать пользовательский ввод, формировать контекст для шаблонов и возвращать рендеринг HTML-страницы.

1. Шаблоны (Templates)

Шаблоны в Django используются для генерации HTML-кода на основе данных. Шаблоны могут включать переменные, условия, циклы и другие элементы, что позволяет генерировать динамические страницы. Django предоставляет свой синтаксис для вставки данных в HTML.

1. Работа с формами

Для обработки пользовательского ввода, такого как отправка данных на сервер, Django предоставляет механизм работы с формами. Формы позволяют создавать и валидировать HTML-формы на основе моделей, обрабатывать отправленные данные и сохранять их в базе данных.

1. Обработка статических файлов

Стаические файлы, такие как CSS, JavaScript и изображения, играют важную роль в визуальной составляющей веб-приложения. Django предоставляет специальные пути для управления статическими файлами и обеспечения их доступности для клиентов.

1. Работа с пользователями и аутентификация

Django предоставляет инструменты для управления пользователями, аутентификации и авторизации. Встроенная система пользователей позволяет регистрировать новых пользователей, аутентифицировать существующих, обрабатывать сессии и управлять доступом к различным частям приложения.

1. Тестирование приложения

Тестирование является важной частью разработки в Django. Фреймворк предоставляет инструменты для написания автоматизированных тестов, которые позволяют проверить корректность работы различных компонентов приложения, включая модели, представления, формы и другие.

В общем, создание и управление веб-приложением в Django требует организации кода в виде приложений, настройки маршрутизации, создания представлений и шаблонов, обработки пользовательских данных и обеспечения безопасности. Фреймворк предоставляет множество инструментов, упрощающих разработку, тестирование и поддержку web-приложений, что делает его эффективным выбором для проекта учета компьютерной техники на предприятии.

ГЛАВА 2. Анализ и проектирование системы учета компьютерной техники

# 2.1 Анализ требований к web-приложению для учета компьютерной техники

На момент создания данной работы существует множество бесплатных приложений как полностью закрытых коммерческих продуктов, так и бесплатных с открытым исходным кодом, разработанных на разных платформах и языках программирования. Например:

1. GLPI (Gestionnaire Libre de Parc Informatique). Сайт - https://glpi-project.org
2. Snipe-IT. Сайт - https://snipeitapp.com
3. OCS Inventory NG. Сайт https://ocsinventory-ng.org/
4. Lansweeper. Сайт https://www.lansweeper.com

И многие другие.

Существует много готовых программ для учета компьютерной техники, предоставляющих широкий спектр функциональности и возможностей. Однако, в некоторых случаях разработка собственной системы учета может оказаться более выгодной и эффективной стратегией для компании. Вот несколько аргументов в пользу разработки своей системы учета несмотря на многообразие готовых решений:

1. **Уникальные потребности бизнеса**: Каждая компания уникальна, и ее потребности в учете компьютерной техники могут отличаться. Разработка собственной системы позволяет точно адаптировать функциональность под конкретные требования организации.
2. **Интеграция с другими системами:** В случае, если компания уже использует специфическое программное обеспечение или системы управления, разработка своей системы может обеспечить более глубокую интеграцию и совместимость с существующими решениями.
3. **Более гибкий контроль и обновления:** Разработка собственной системы учета предоставляет компании полный контроль над функциональностью и обновлениями. Это позволяет оперативно внедрять изменения, адаптируя систему к изменяющимся потребностям бизнеса.
4. **Экономическая эффективность в долгосрочной перспективе:** возможно, вложение в разработку своей системы учета в начале может казаться затратным. Однако в долгосрочной перспективе это может оказаться экономически эффективным, особенно если предполагается долгосрочное использование и постоянные изменения в бизнес-процессах.
5. **Соблюдение уникальных стандартов безопасности:** когда речь идет о хранении и обработке конфиденциальной информации, разработка собственной системы позволяет более тщательно настраивать меры безопасности, соответствуя уникальным требованиям компании.

В целом, несмотря на наличие готовых решений, решение о разработке своей системы учета компьютерной техники стоит принимать на основе конкретных потребностей и стратегических целей организации.

При разработке системы учета компьютерной техники, важно признать, что создание универсальной системы, полностью удовлетворяющей всем требованиям, может оказаться сложной и затратной задачей в рамках ограниченных ресурсов, доступных для данной работы. В связи с этим, разумным шагом может быть ограничение функционала до базовых возможностей.

Фокусировка на базовом функционале позволит удовлетворить основные потребности организации в учете компьютерной техники. Это обеспечит минимальные, но необходимые функции для эффективного контроля и управления активами. Ограничение функционала упрощает процесс разработки и внедрения системы. Это позволяет быстрее достичь рабочей версии и минимизировать временные затраты, что важно при ограниченных сроках проекта. Создание сложной системы с обширным функционалом требует значительных ресурсов в виде времени, финансов и человеческих ресурсов. Ограничение функционала поможет сохранить эти ресурсы и сосредоточиться на ключевых аспектах проекта. Построение системы с базовым функционалом не исключает возможность постепенного расширения возможностей в будущем. Это позволяет компании адаптировать систему по мере роста бизнеса и появления новых потребностей.

Ограничение функционала в начальной стадии разработки может служить разумным компромиссом, обеспечивая баланс между функциональностью и ограниченными ресурсами проекта.

# 2.2 Проектирование базы данных для хранения информации о компьютерной технике

Проектирование базы данных для учета компьютерной техники представляет собой критически важный этап, определяющий эффективность и надежность системы управления IT-активами. В эпоху, где компьютеры и информационные технологии являются неотъемлемой частью организационной инфраструктуры, правильно спроектированная база данных становится фундаментом для эффективного мониторинга, учета и управления компьютерными ресурсами.

В данном контексте, необходимо учесть не только текущие потребности организации, но и перспективы ее развития, адаптируя базу данных для возможного роста и изменений в бизнес-процессах. Наша задача - предоставить надежный фундамент для создания базы данных, которая будет отвечать не только текущим, но и будущим потребностям вашей организации.

ER-диаграмма (ERD) — важный инструмент при проектировании баз данных. Она позволяет визуализировать структуру данных и взаимосвязи между ними. Процесс ее создания включает определение сущностей, их атрибутов и взаимосвязей. ERD является мощным инструментом для моделирования связей между сущностями в базе данных.

Создадим ERD нашей будущей базы.

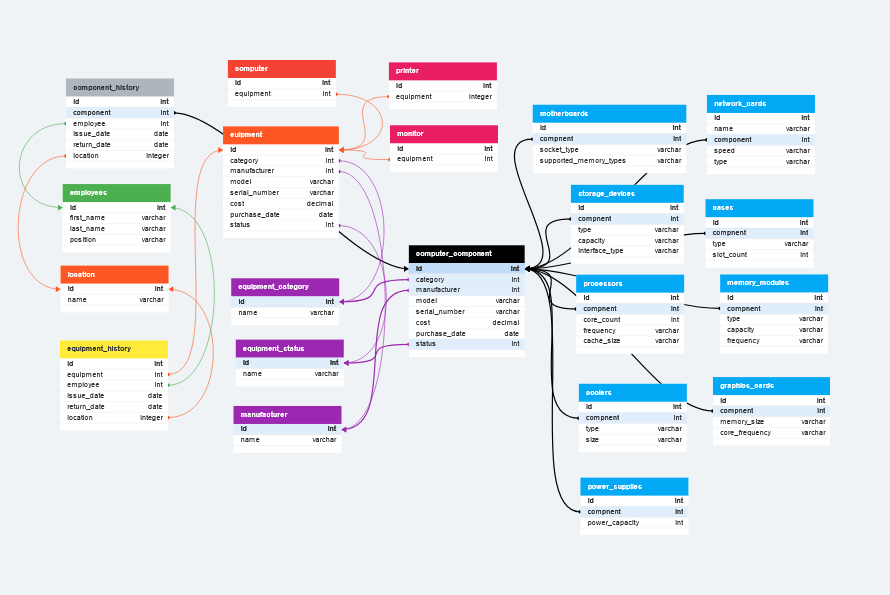


Рисунок 1

# 2.3 Определение функциональных возможностей приложения

Целью данного раздела является четкое определение функциональных возможностей приложения, предназначенного для учета компьютерной техники, ее компонентов, а также информации о местоположении и пользователях. Приложение разработано с учетом потребностей в эффективном и надежном управлении IT-активами.

1. Учет Компьютерной Техники: приложение предоставляет возможность подробного учета компьютеров и другой офисной техники, принтеров, телефонов и т.д. включая модель, серийный номер, характеристики и дату приобретения. Модуль для внесения и обновления данных о компьютерной технике.
2. Учет Компонентов: возможность регистрации и отслеживания компонентов компьютерной техники, таких как процессоры, жесткие диски, видеокарты и другие. Привязка компонентов к конкретным компьютерам для точного учета использования и обновлений.
3. Местоположение Техники: функциональность для указания и изменения физического местоположения компьютеров. Отслеживание перемещений компьютеров между разными подразделениями или офисами.
4. Учет Пользователей: список пользователей, использующих компьютерную технику. Привязка компьютеров к конкретным пользователям для более точного мониторинга и учета ответственных лиц.
5. Базовые справочники: кроме функциональности по учету компьютеров, их компонентов, местоположения и пользователей, приложение также предоставляет возможности для ведения справочников. Эти справочники служат ключевой составляющей для дополнительного контроля, управления и анализа данных, а также оптимизации процессов учета и администрирования.
6. Привязка и Удаление Компонентов: возможность привязки компонентов (например, замена жесткого диска) к конкретному компьютеру. Механизм удаления компонентов из системы в случае необходимости замены или апгрейда.
7. Интерфейс и Управление: интуитивно понятный пользовательский интерфейс для удобного внесения и просмотра данных.

ГЛАВА 3. Разработка web-приложения с использованием Django

# 3.1 Создание основных моделей данных при помощи Django ORM

В этой главе начнем разработку основных моделей данных для нашего web-приложения с использованием Django ORM. определим структуру базы данных, которая будет поддерживать учет компьютеров, компонентов, сотрудников и другого оборудования.

Приложение CAP (Computer Accounting Program) разрабатывается как комплексное решение, состоящее из четырех отдельных Django-приложений, каждое из которых фокусируется на определенном функциональном аспекте системы, предоставляя полный инструментарий для эффективного учета и управления IT-активами.

Приложение "Справочники" (Catalogs):

Это приложение предоставляет удобный интерфейс для ведения и обновления справочников, содержащих ключевую информацию для других модулей. Ведение справочника моделей компьютеров, компонентов, пользователей и местоположений.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| Модель | Описание | Наследование | Дополнительные Поля |
| BaseCommonInfo | Базовая модель справочников | - | id (AutoField), name (CharField), created\_at (DateTimeField), updated\_at (DateTimeField) |
| Manufacturer | Регистрация производителей техники | BaseCommonInfo | - |
| EquipmentCategory | Классификация оборудования | BaseCommonInfo | - |
| EquipmentStatus | Отслеживание статусов оборудования | BaseCommonInfo | - |
| ComponentStatus | Отслеживание статусов компонентов | BaseCommonInfo | - |
| MemoryType | Учет различных типов памяти в компьютерах | BaseCommonInfo | - |
| StorageType | Учет различных типов накопителей | BaseCommonInfo | - |
| SocketType | Учет различных типов разъемов | BaseCommonInfo | - |
| Employee | Информация о сотрудниках | BaseCommonInfo | position (CharField) |
| Location | Информация о местоположении | BaseCommonInfo | address (CharField) |

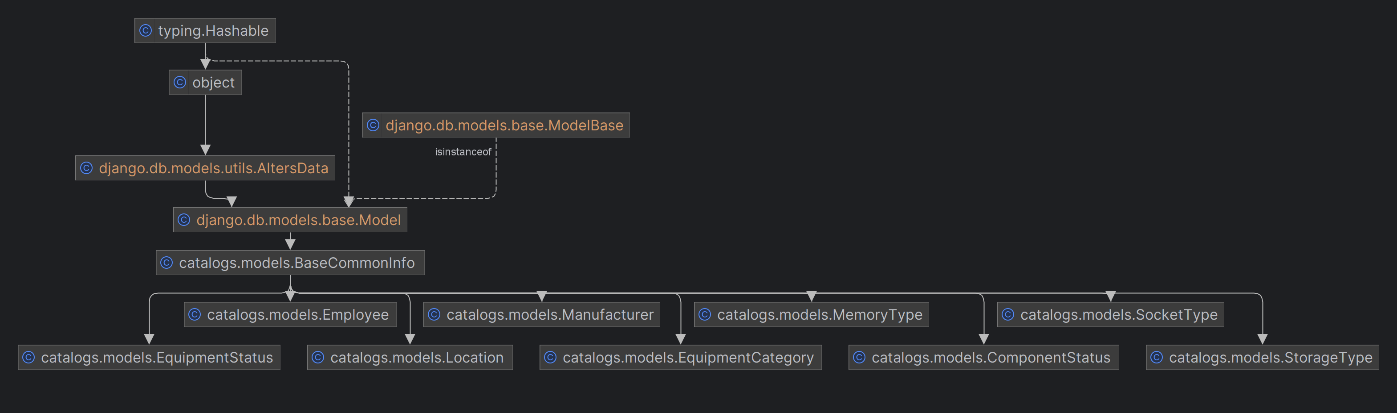


Рисунок 2

Приложение "Оборудование" (Equipments):

Это приложение предоставляет средства для учета и управления компьютерами. Регистрация компьютеров с указанием модели, серийного номера и статуса. Привязка компьютеров к пользователям и местоположению. Отслеживание перемещений компьютеров между разными местоположениями.

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| Модель | Описание | Наследование | Дополнительные Поля |
| Equipment | Оборудование | - | id (AutoField), name (CharField), category (ForeignKey), manufacturer (ForeignKey), model (CharField), serial\_number (CharField), inventory\_number (CharField), cost (DecimalField), purchase\_date (DateTimeField), equipment\_status (ForeignKey) |
| Computer | Компьютер | Equipment | - |
| Monitor | Монитор | Equipment | - |
| Printer | Принтер | Equipment | - |
| NetworkDevice | Сетевое устройство | Equipment | - |
| Phone | Телефон | Equipment | - |
| OtherEquipment | Другое оборудование | Equipment | - |

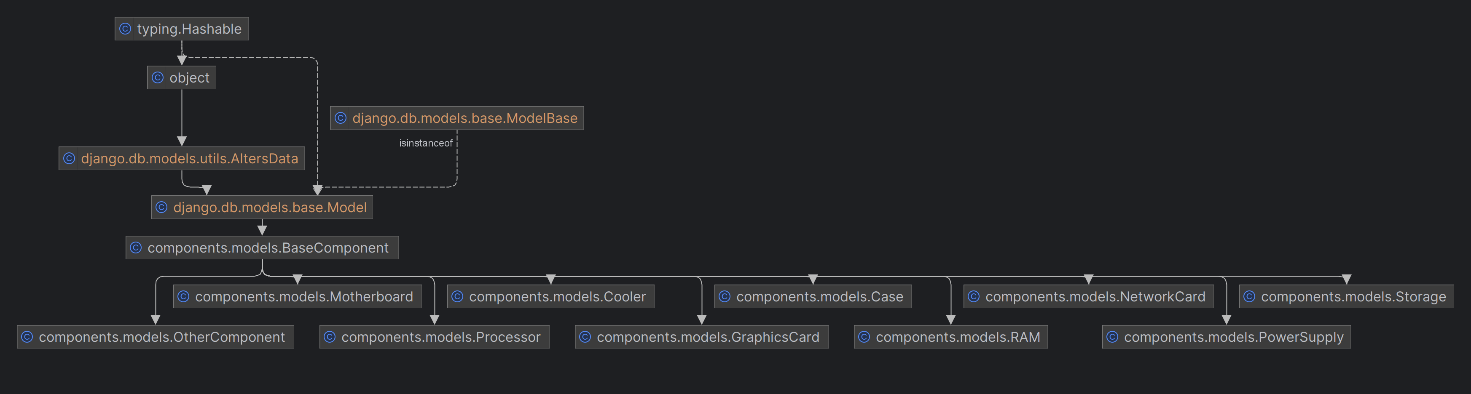


Рисунок 2

Приложение "Компоненты компьютера" (Components):

Это приложение специализируется на учете и управлении компонентами компьютеров. Ведение справочника компонентов, таких как процессоры, жесткие диски, оперативная память и т.д. Привязка компонентов к конкретным моделям и компьютерам.

Таблица 3

|  |  |  |  |
| --- | --- | --- | --- |
| Модель | Описание | Наследование | Дополнительные Поля |
| BaseComponent | Базовая модель компонентов оборудования | - | id (AutoField), name (CharField), serial\_number (CharField), inventory\_number (CharField), manufacturer (ForeignKey), cost (DecimalField), start\_date (DateTimeField), end\_date (DateTimeField), component\_status (ForeignKey) |
| Motherboard | Материнская плата | BaseComponent | socket\_type (ForeignKey), supported\_memory\_types (ManyToManyField), in\_computer (OneToOneField) |
| Processor | Процессор | BaseComponent | socket\_type (ForeignKey), num\_cores (IntegerField), frequency (IntegerField), in\_computer (ForeignKey) |
| RAM | Оперативная память | BaseComponent | memory\_type (ForeignKey), capacity (IntegerField), frequency (IntegerField), in\_computer (ForeignKey) |
| GraphicsCard | Видеокарта | BaseComponent | memory (CharField), frequency (IntegerField), in\_computer (ForeignKey) |
| Storage | Накопитель | BaseComponent | storage\_type (ForeignKey), capacity (IntegerField), interface (CharField), in\_computer (ForeignKey) |
| PowerSupply | Блок питания | BaseComponent | power (IntegerField), in\_computer (ForeignKey) |
| Cooler | Система охлаждения | BaseComponent | cooler\_type (CharField), size (CharField), in\_computer (ForeignKey) |
| Case | Корпус | BaseComponent | case\_type (CharField), num\_bays (IntegerField), in\_computer (ForeignKey) |

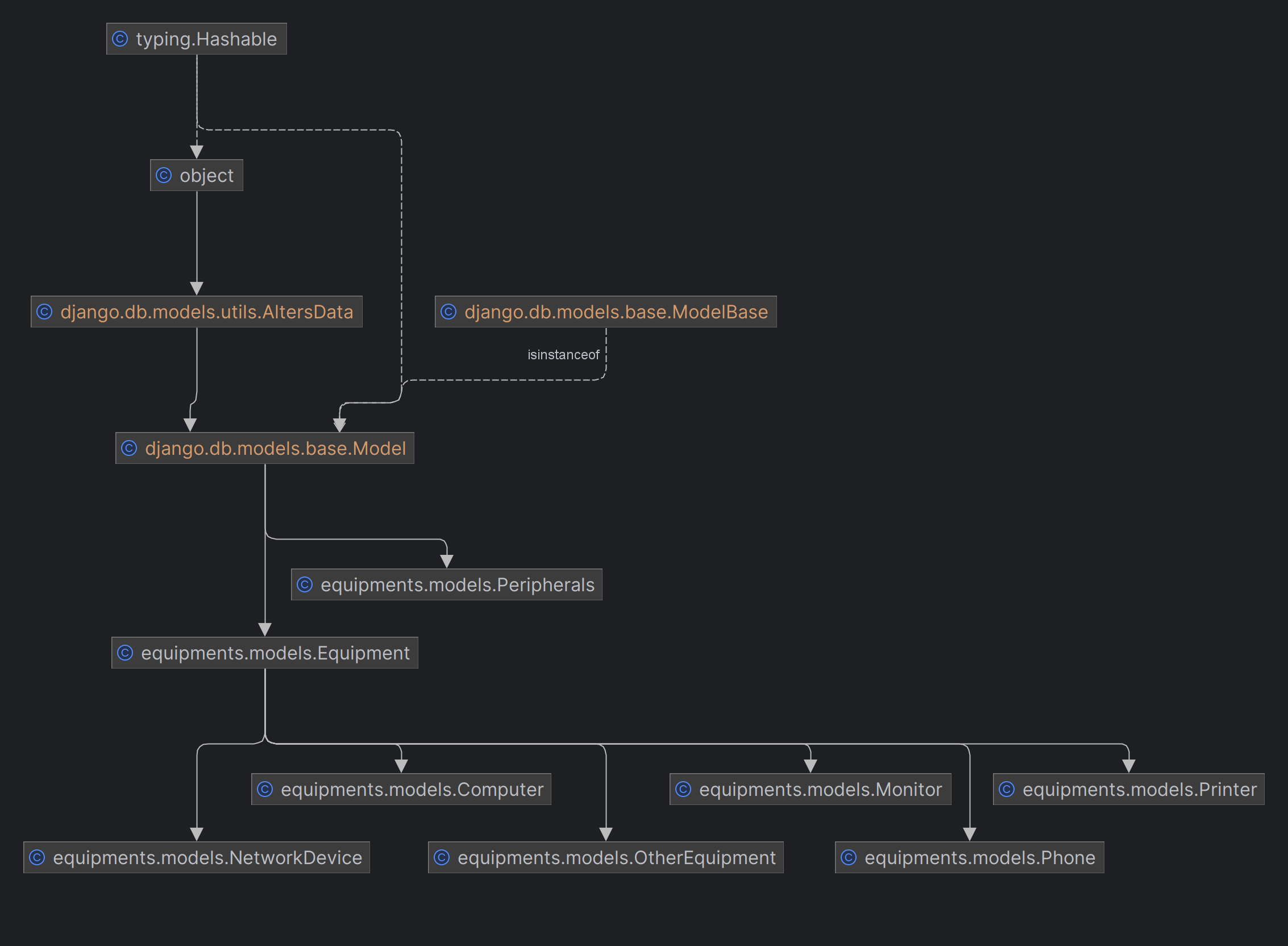


Рисунок 3

Приложение "История" (History):

Это приложение обеспечивает полную видимость истории перемещений компьютеров и другой техники, изменений владельцев.

Таблица 4

|  |  |  |  |
| --- | --- | --- | --- |
| Модель | Описание | Наследование | Дополнительные Поля |
| EquipmentHistory | История оборудования | - | employee (ForeignKey), location (ForeignKey), start\_date (DateTimeField), end\_date (DateTimeField) |
| ComputerHistory | История компьютера | EquipmentHistory | computer (ForeignKey) |
| MonitorHistory | История монитора | EquipmentHistory | monitor (ForeignKey) |
| PrinterHistory | История принтера | EquipmentHistory | printer (ForeignKey) |
| PhoneHistory | История телефона | EquipmentHistory | phone (ForeignKey) |
| NetworkDeviceHistory | История сетевого оборудования | EquipmentHistory | networkdevice (ForeignKey) |
| OtherEquipmentHistory | История другого оборудования | EquipmentHistory | otherequipment (ForeignKey) |
| ComputerComponentHistory | История компонента компьютера | - | computer (ForeignKey), start\_date (DateField), end\_date (DateField) |
| MotherBoardHistory | История материнской платы | ComputerComponentHistory | motherboard (ForeignKey) |
| ProcessorHistory | История процессора | ComputerComponentHistory | processor (ForeignKey) |
| RAMHistory | История оперативной памяти | ComputerComponentHistory | memory (ForeignKey) |
| StorageHistory | История жесткого диска | ComputerComponentHistory | storage (ForeignKey) |
| GraphicsCardHistory | История видеокарты | ComputerComponentHistory | graphicscard (ForeignKey) |
| CoolerHistory | История системы охлаждения | ComputerComponentHistory | cooler (ForeignKey) |
| PowerSupplyHistory | История блока питания | ComputerComponentHistory | powersupply (ForeignKey) |
| CaseHistory | История корпуса | ComputerComponentHistory | case (ForeignKey) |
| NetworkCardHistory | История сетевой карты | ComputerComponentHistory | networkcard (ForeignKey) |
| OtherComponentHistory | История другого компонента | ComputerComponentHistory | othercomponent (ForeignKey) |

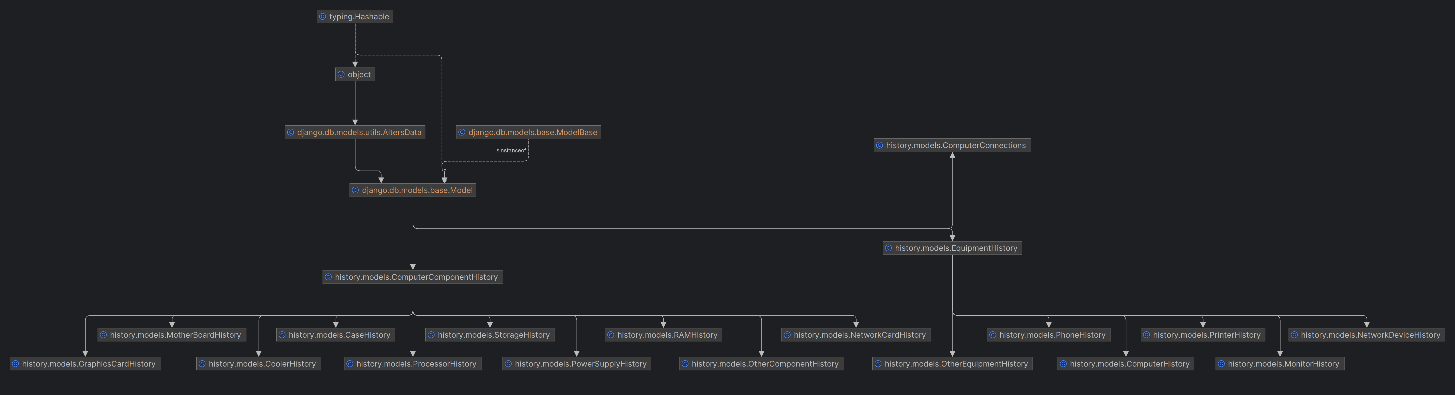


Рисунок 4

# 3.2 Разработка пользовательского интерфейса с использованием шаблонов (Templates)

Разработка пользовательского интерфейса в web-приложении на Django включает в себя создание шаблонов, которые определяют внешний вид страниц и способ представления данных. В данной главе будет представлен процесс разработки пользовательского интерфейса для системы учета компьютерной техники на предприятии.

Для создания внешнего вид страниц я буду использовать Bootstrap. Bootstrap - это мощный фреймворк для разработки веб-приложений, предоставляющий множество преимуществ. Страницы приложений, построенные с использованием Bootstrap, выгодно обладают отзывчивым и адаптивным дизайном, имеют единообразный стиль благодаря готовым компонентам, обеспечивают кроссбраузерность, легкость разметки через сеточную систему, добавляют интерактивность с помощью JavaScript-плагинов, поддерживаются активным сообществом, обеспечивают высокую производительность и гибкость расширения для создания уникальных дизайнов.

Создадим базовый шаблон, который будет предоставлять структуру для создания различных страниц веб-приложения, а блоки контента позволяют внедрять специфичный контент на каждой отдельной странице.

Шаблон base.html состоит из:

1. Метаинформация и подключение ресурсов:

Определение кодировки и настроек отображения. Подключение стилей Bootstrap, Font Awesome и пользовательских стилей.

1. Навигационное меню:

Создание навигационного меню с логотипом и кнопкой "бургер" для адаптивности. Включение ссылок на верхнем и боковом меню.

1. Структура страницы:

Разделение страницы на две колонки: левую для бокового меню и правую для основного контента, используя Bootstrap Grid System.

1. Блоки контента:

Использование Django-блоков для внедрения динамического контента на странице. Определение блоков для хлебных крошек, заголовка страницы, кнопок, основного контента и дополнительного контента.

1. Футер:

Включение футера с копирайтом и возможностью добавления дополнительной информации о проекте.

1. Подключение JavaScript:

Подключение JavaScript-скрипта Bootstrap для обеспечения интерактивности.

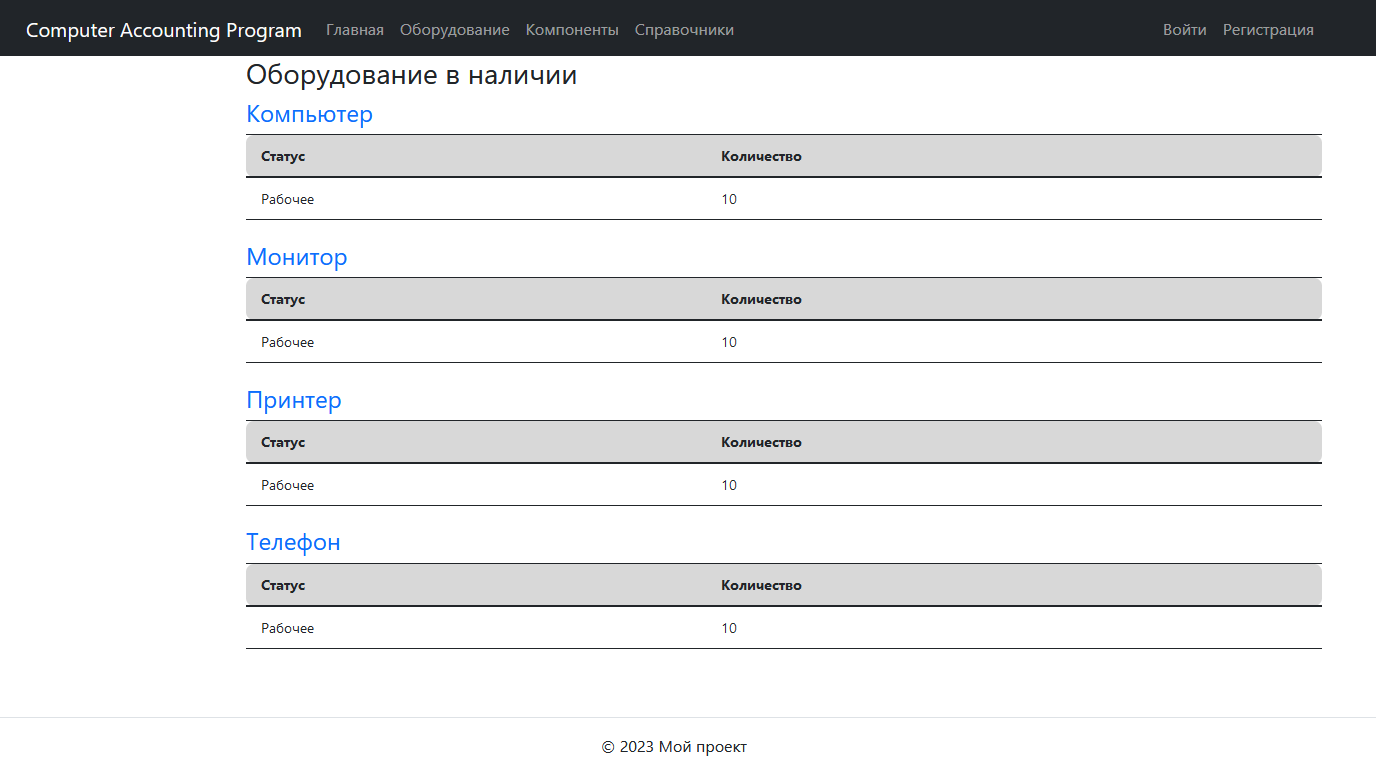


Рисунок 3

На странице "Оборудование", основанной на базовом шаблоне base.html, представлены таблицы с информацией о компонентах. В левой части страницы находится боковое меню, предоставляющее пользователю удобный способ перехода к персональным страницам каждого компонента. Боковое меню и таблицы создают единый интерфейс, обеспечивая структурированное и интуитивно понятное взаимодействие с данными оборудования. Адаптивный дизайн Bootstrap гарантирует, что страница корректно отображается на различных устройствах, обеспечивая удобство использования.

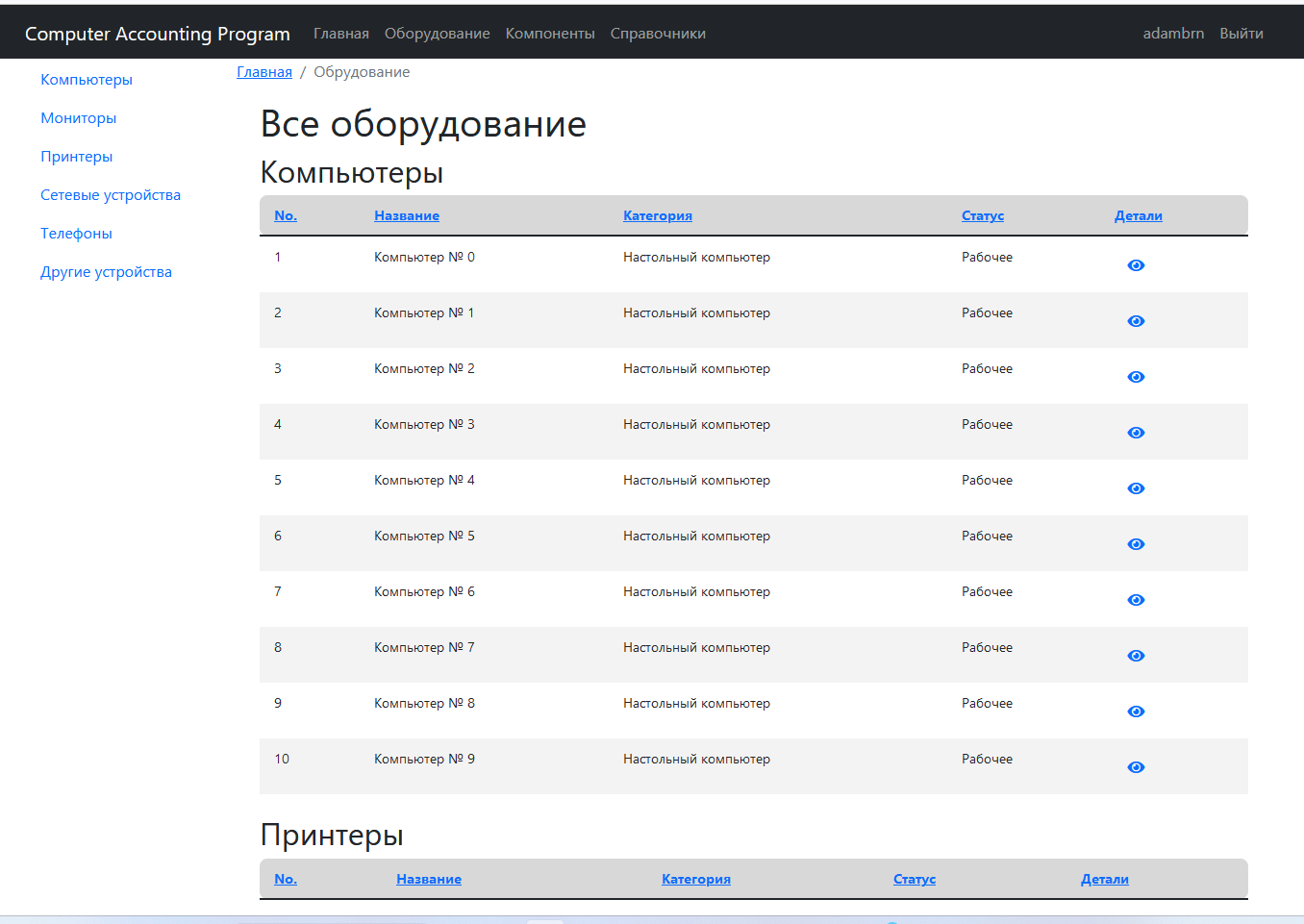


Рисунок 4

Страницы "Компоненты" и "Справочники" в нашем веб-приложении разработаны аналогичным образом с использованием базового шаблона base.html. На обеих страницах реализованы таблицы с информацией о компонентах и справочниках соответственно, а также боковое меню слева, предоставляющее удобный доступ к персональным страницам каждого элемента.

Благодаря применению отзывчивого дизайна Bootstrap, обеспечивается корректное отображение страниц на различных устройствах. Это делает взаимодействие с компонентами и справочниками максимально удобным и интуитивно понятным для пользователя. Все элементы интерфейса согласованы в стиле, что придает страницам профессиональный и единообразный вид. Такой подход обеспечивает легкость восприятия информации и повышает удовлетворенность пользователей при использовании веб-приложения.

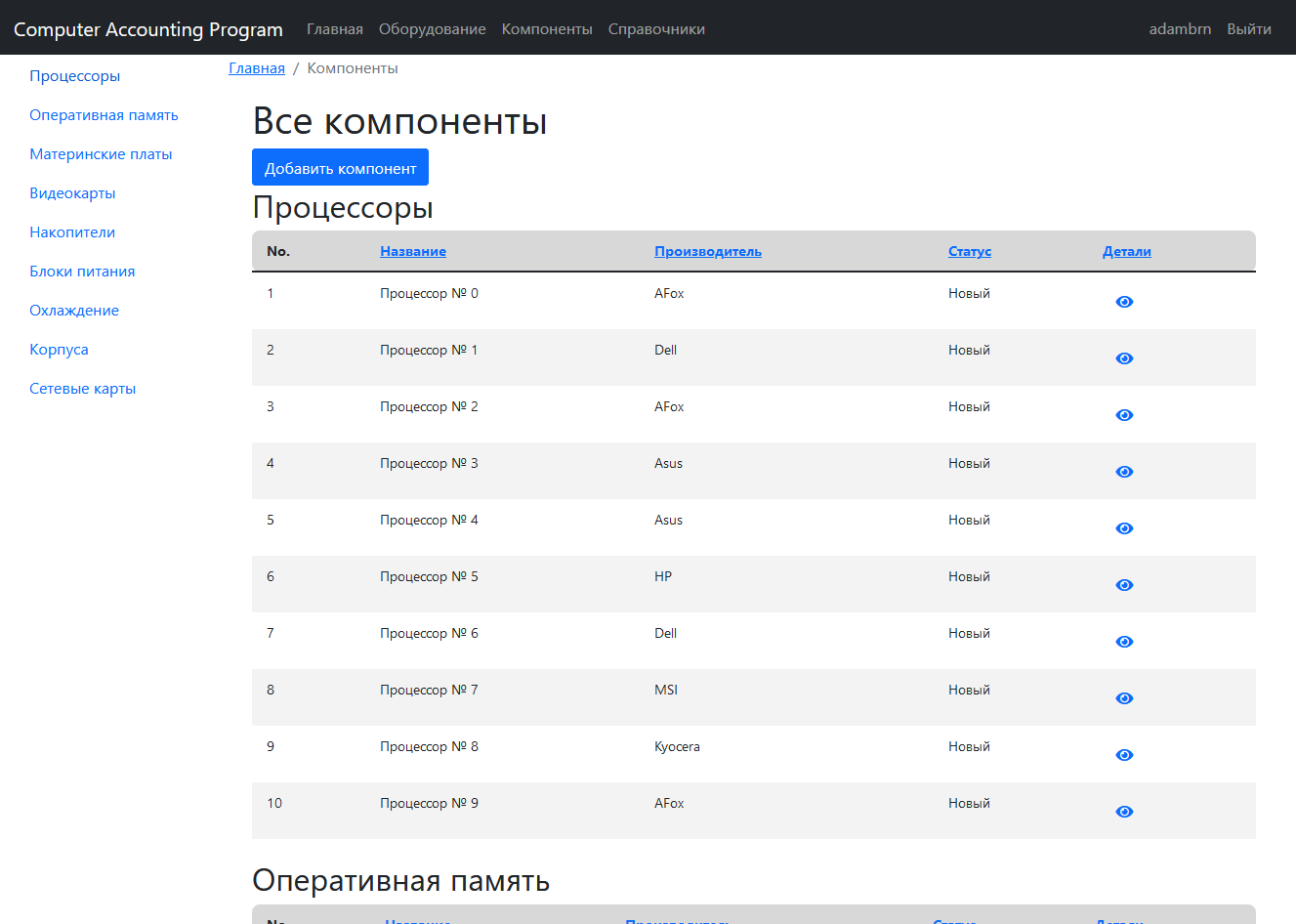


Рисунок 5

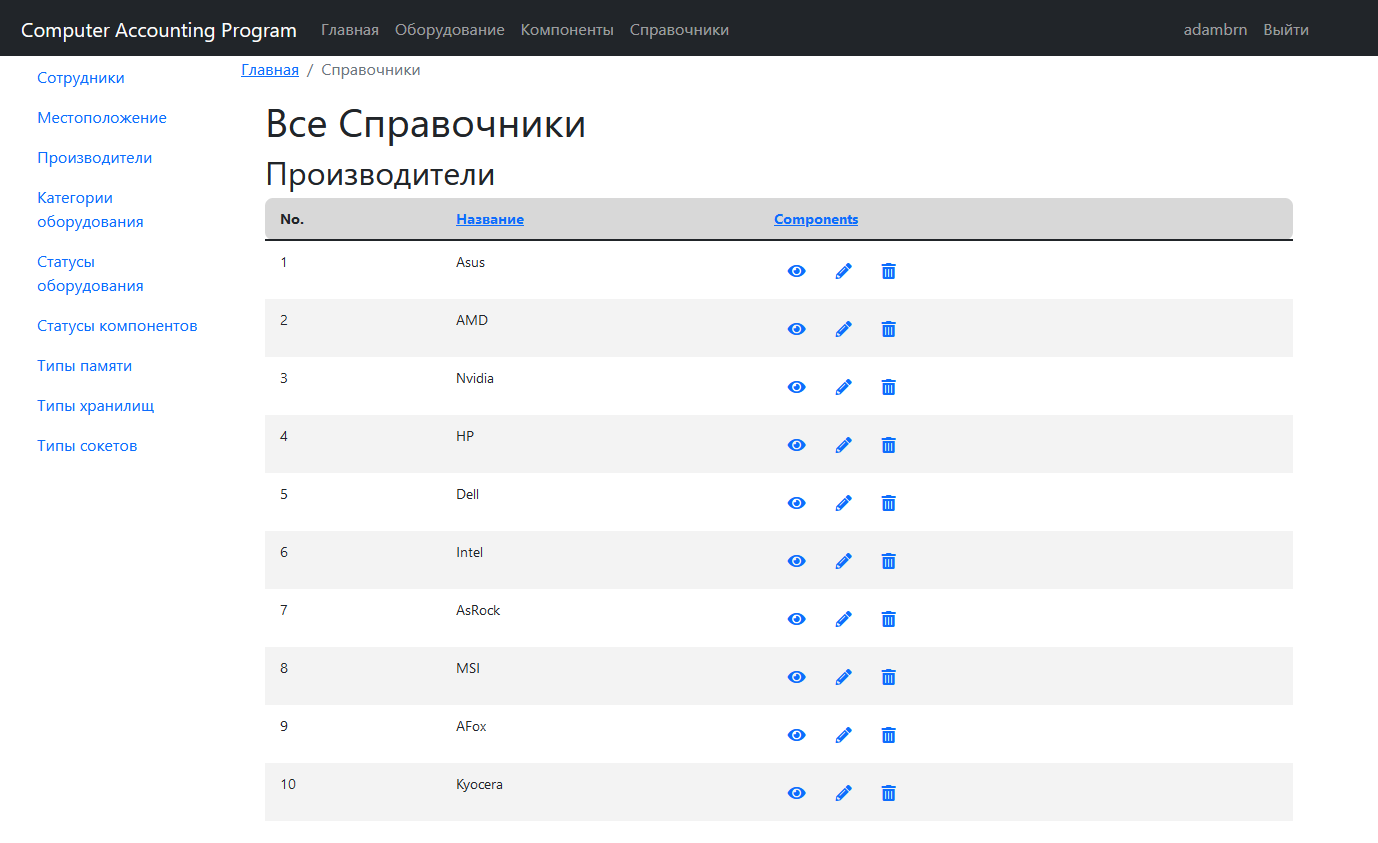


Рисунок 6

Для эффективного управления всем оборудованием в нашем веб-приложении разрабатываем базовые шаблоны, которые будут обеспечивать надежное и интуитивно понятное взаимодействие пользователя с системой.

equipment\_list - этот шаблон предназначен для отображения списка оборудования. предусматриваем информативную таблицу, включающую основные характеристики каждого компонента. Пользователи могут легко просматривать, фильтровать и сортировать данные, обеспечивая комфортное взаимодействие с обширным списком оборудования.

equipment\_form - этот шаблон разработан для предоставления пользователю возможности создавать новые записи об оборудовании. интегрируем удобные формы с валидацией данных, что обеспечивает точность ввода информации. Пользователи могут легко добавлять новые компоненты в систему, заполняя соответствующие поля.

equipment\_detail - данный шаблон предоставляет детальную информацию о конкретном компоненте оборудования. предусматриваем информативный макет, включающий ключевые характеристики, статус, и другие детали. Пользователи смогут подробно ознакомиться с информацией о каждом компоненте.

equipment\_confirm\_delete - этот шаблон предназначен для подтверждения удаления компонента. интегрируем подтверждающий макет, который предостерегает пользователя от случайного удаления. Этот шаблон поддерживает безопасное и контролируемое удаление записей из системы. С использованием этих базовых шаблонов стремимся обеспечить единообразие и интуитивно понятный пользовательский опыт в процессе взаимодействия с оборудованием в рамках нашего веб-приложения.

На основе наших базовых шаблонов, предпринимаем шаги по персонализации для каждого вида оборудования, обеспечивая уникальный и оптимизированный опыт взаимодействия с каждым компонентом. Создание наследуемых шаблонов для персонализации является ключевым аспектом нашего подхода к дизайну интерфейса.

Для списка компонентов (equipment\_list) создам наследуемый шаблон computer\_list, printer\_list, monitor\_list и так далее, в зависимости от вида компонента. Эти шаблоны адаптируют базовый computer\_list под уникальные требования каждого вида оборудования, отображая необходимые дополнительные характеристики и иконки для легкого распознавания.

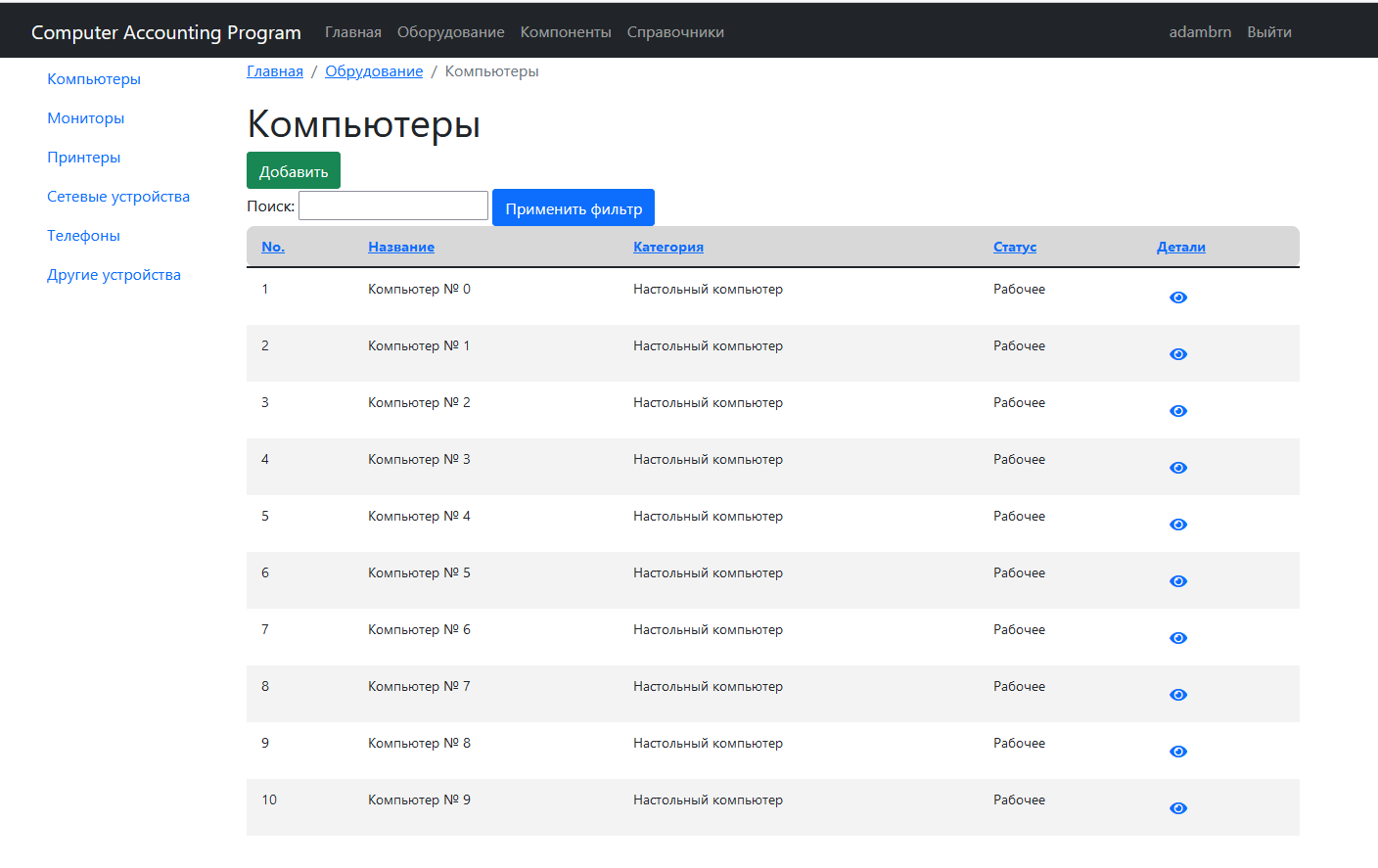


Рисунок 7

Для формы добавления и редактирования нового оборудования (equipment\_form) создам шаблоны computer\_form, printer\_form, monitor\_form и прочие, унаследованные от equipment\_form.

Эти наследуемые шаблоны могут включать специфические поля и опции ввода данных, соответствующие уникальным характеристикам каждого типа оборудования.

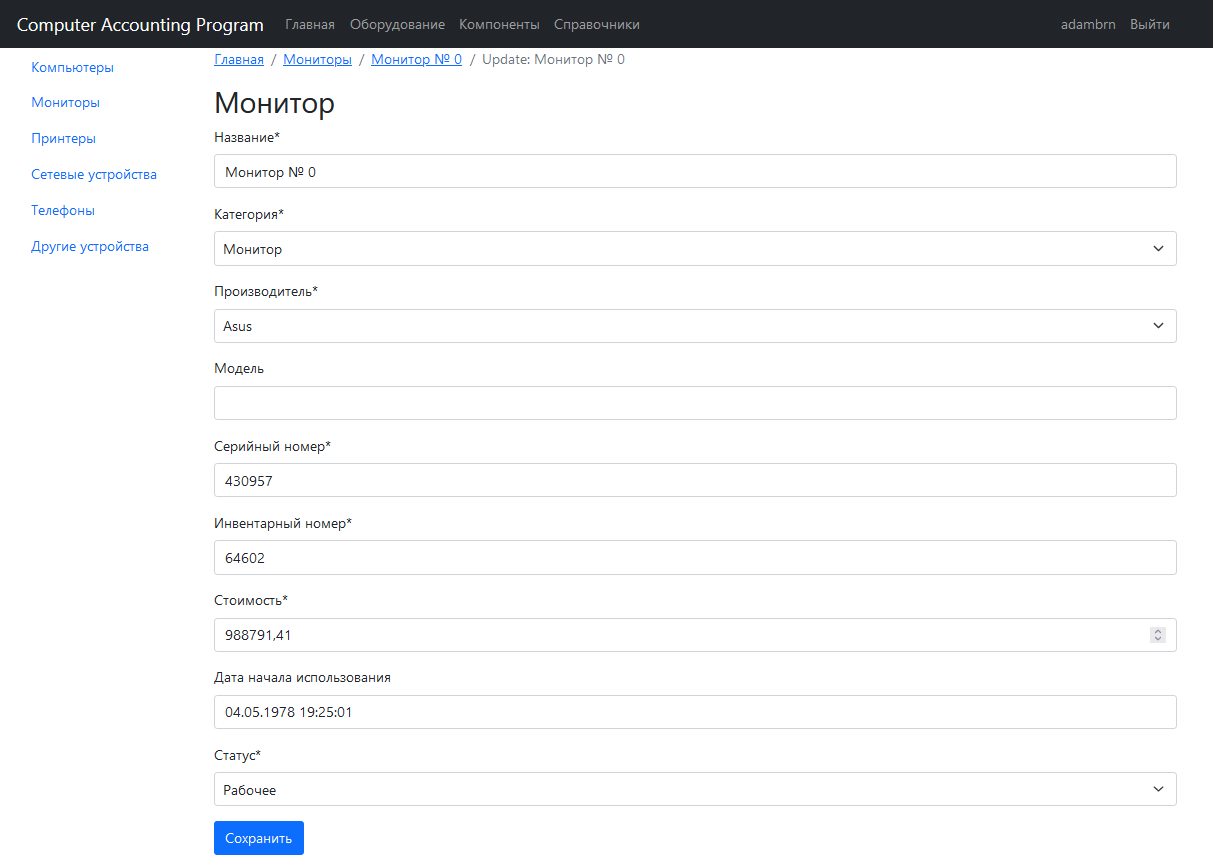


Рисунок 8

Для детальной страницы компонента (equipment\_detail) создаем computer\_detail, printer\_detail, monitor\_detail и так далее, унаследованные от equipment\_detail. На этих страницах адаптируем представление, чтобы отображать дополнительные детали и спецификации, специфичные для каждого вида оборудования.

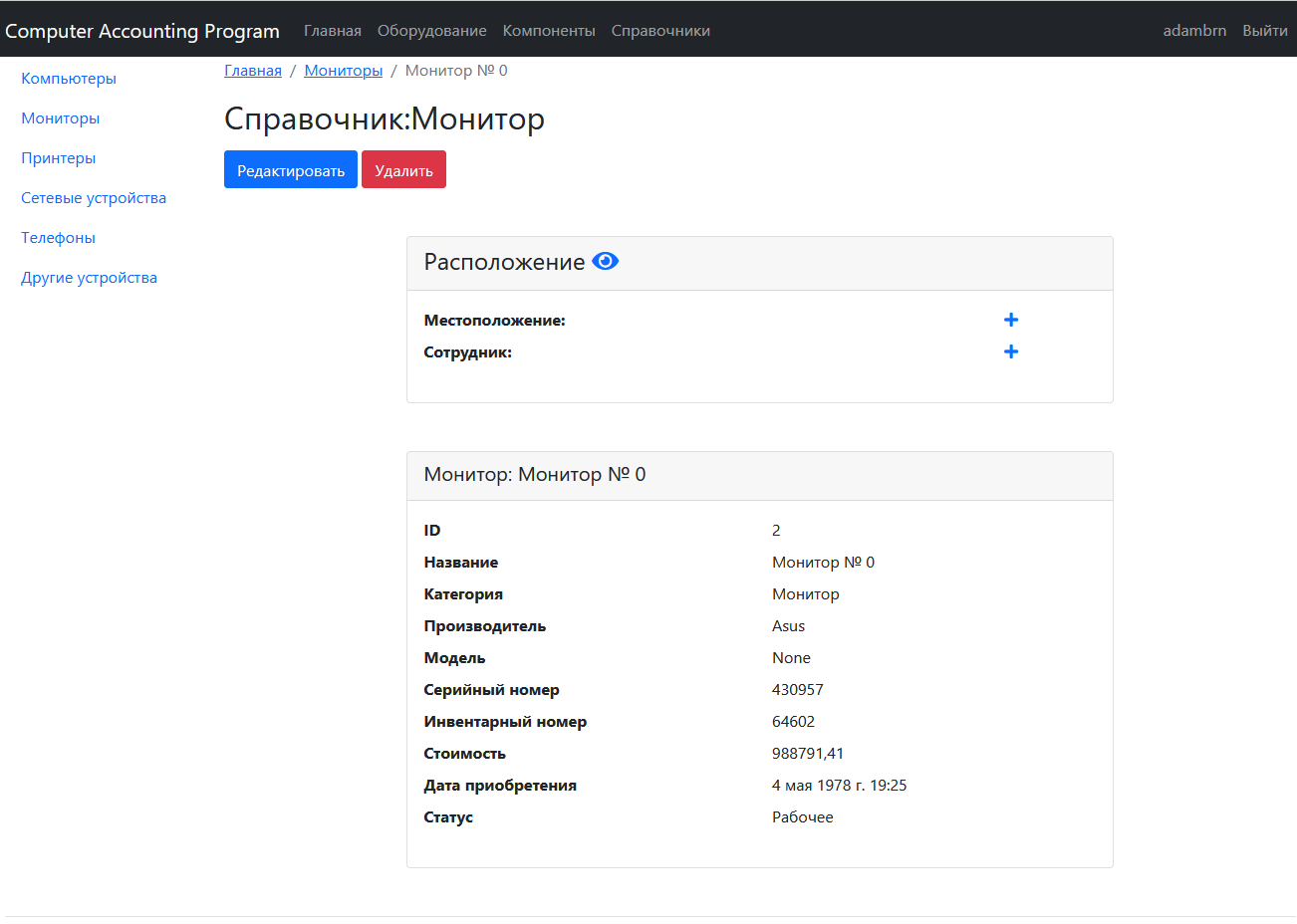


Рисунок 9

Для подтверждения удаления компонента (equipment\_confirm\_delete) cоздам шаблоны computer\_confirm\_delete, printer\_confirm\_delete, monitor\_confirm\_delete и другие, унаследованные от equipment\_confirm\_delete.

Эти шаблоны могут интегрировать специфические сообщения и подтверждения, соответствующие контексту удаления каждого компонента.

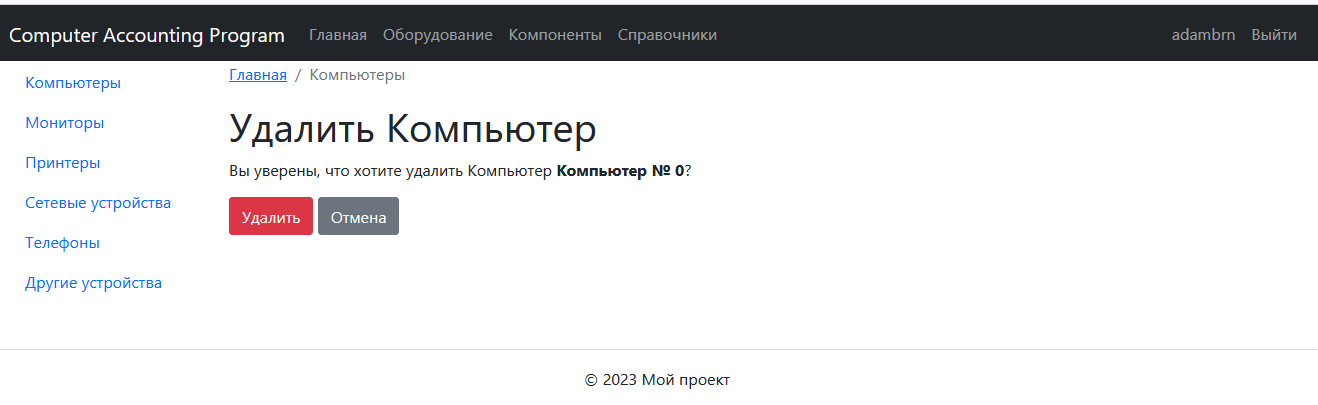


Рисунок 10

Такой подход к наследованию шаблонов обеспечивает гибкость и масштабируемость, позволяя нам эффективно адаптировать интерфейс для каждого типа оборудования, улучшая восприятие пользователем и обеспечивая более персонализированный опыт.

В рамках этого веб-приложения "Computer Accounting Program" придерживаемся единого и гибкого подхода к созданию шаблонов для приложений "Справочники" и "Компоненты". Этот подход включает в себя создание базовых шаблонов с последующим наследованием для каждого конкретного типа оборудования или справочника, обеспечивая единообразие и персонализацию пользовательского опыта.

Для приложения "Справочники":

Создаю базовые шаблоны catalog\_list, catalog\_form, catalog\_detail, и catalog\_confirm\_delete.

На основе этих базовых шаблонов создаем уникальные наследуемые шаблоны для каждого справочника, такие как manufacturer\_list, equipment\_category\_list, component\_status\_list, и другие.

Эти наследуемые шаблоны адаптируют базовые шаблоны, учитывая особенности и требования каждого справочника, отображая соответствующую информацию и предоставляя функционал для управления данными.

Для приложения "Компоненты":

Разрабатываю базовые шаблоны equipment\_list, equipment\_form, equipment\_detail, и equipment\_confirm\_delete.

На основе этих базовых шаблонов создаем уникальные наследуемые шаблоны для каждого типа оборудования, такие как motherboard\_list, processor\_list, ram\_list, и другие.

Эти наследуемые шаблоны адаптируют базовые шаблоны, предоставляя дополнительные детали и функциональность, соответствующие характеристикам каждого компонента.

Такой подход позволяет эффективно использовать общий дизайн, обеспечивая при этом гибкость и персонализацию для каждой категории оборудования и справочника. Результатом является удобный и единообразный пользовательский интерфейс, который улучшает восприятие и опыт работы с приложением.

# 3.3 Реализация функциональности учета компьютерной техники: добавление, редактирование, удаление

В данной главе реализуется функциональность учета компьютерной техники, охватывающая ключевые операции: добавление, редактирование и удаление оборудования и компонентов компьютера. В целях обеспечения удобства пользователя и эффективного взаимодействия, каждая из этих операций имеет свои уникальные особенности.

1. Добавление оборудования и компонента.

Пользователь имеет возможность добавить новый компонент, выбрав соответствующую опцию в интерфейсе. Форма добавления компонента предоставляет все необходимые поля ввода, учитывая характеристики конкретного типа оборудования.

1. Редактирование оборудования и компонента.

Пользователь может изменить данные о существующем компоненте, перейдя на страницу редактирования соответствующего компонента. Форма редактирования содержит текущие значения полей, предоставляя возможность внесения изменений. Система также обеспечивает проверку и валидацию введенных данных для предотвращения ошибок.

1. Удаление оборудования и компонента.

Пользователь имеет опцию удалить компонент, подтвердив свое решение в диалоговом окне подтверждения. Реализован механизм подтверждения, чтобы предотвратить случайное удаление важной информации.

Для улучшения удобства использования и функционала системы учета компьютерной техники, был разработан индивидуальный шаблон для модели "Компьютер". Этот шаблон предоставляет уникальные возможности по управлению компонентами, включая добавление и исключение из состава компьютера. Для модели "Компьютер" был специально создан шаблон computer\_detail, который предоставляет обзор всех характеристик компьютера. Этот шаблон также включает в себя дополнительные разделы для управления компонентами, обеспечивая легкий доступ к функционалу добавления и удаления.

Пользователь, находясь на странице деталей компьютера, может легко добавить новые компоненты, выбрав соответствующую опцию.

Реализована форма выбора компонентов с предварительным фильтром, учитывая только те компоненты, которые еще не включены в состав компьютера. Выбранные компоненты могут быть добавлены в компьютер одним кликом, и изменения моментально отражаются в системе учета.

Пользователь также имеет возможность удалить компоненты из состава компьютера, перейдя в соответствующий раздел. Реализована навигационная система для предоставления списка текущих компонентов, включенных в компьютер. Удаление компонентов также происходит в режиме реального времени, обновляя информацию на странице компьютера.

Такой подход позволяет пользователям удобно управлять конфигурацией компьютера, динамически добавляя и исключая компоненты в зависимости от потребностей. Это повышает гибкость и функциональность системы учета компьютерной техники.

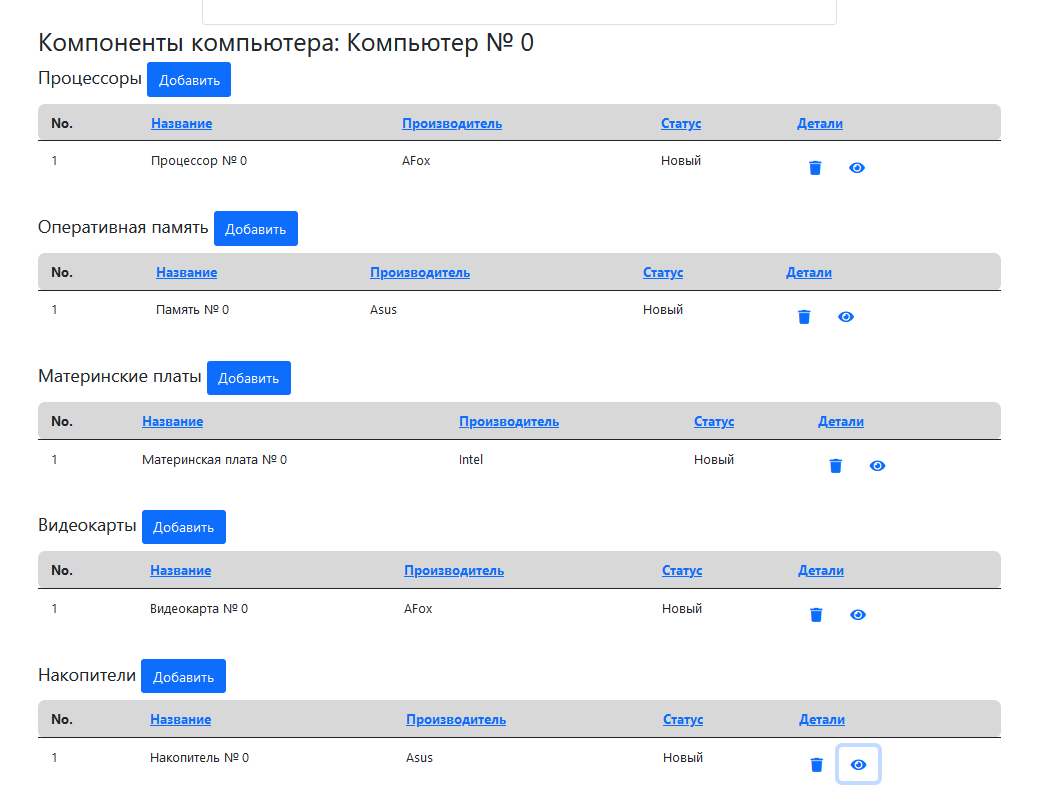


Рисунок 11

Реализована функциональность, позволяющая вести историю перемещений оборудования, что обеспечивает подробное отслеживание изменений и улучшает контроль над местоположением техники в рамках системы учета компьютерной техники.

**История перемещений:**

Для каждого элемента оборудования в системе ведется история его перемещений с использованием специальной модели данных.

Каждая запись в истории включает в себя детализированную информацию о дате и времени перемещения, предыдущем и новом местоположении, а также ответственном лице за перемещение.

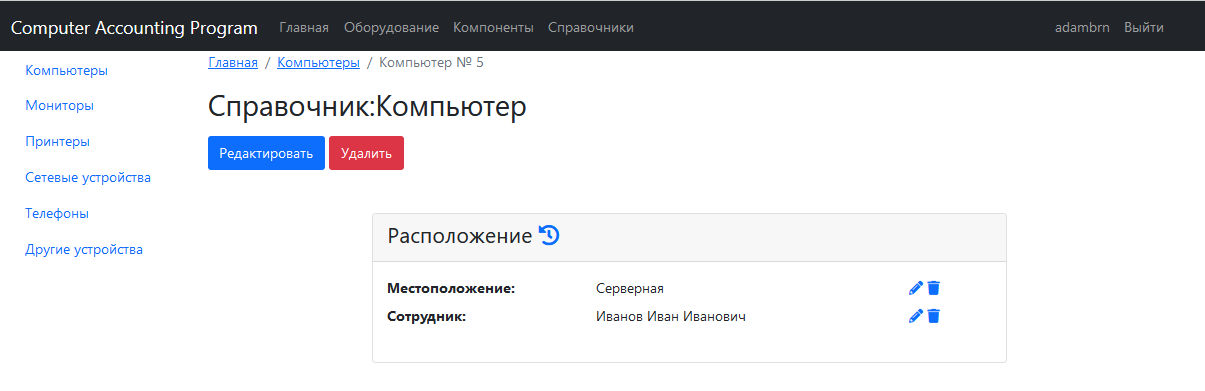


Рисунок 12

Эта информация сохраняется в базе данных, обеспечивая точный хронологический порядок перемещений и возможность анализа истории.

**Отображение истории:**

Пользователи могут просматривать историю перемещений для каждого оборудования, перейдя на соответствующую страницу.

Для повышения удобства, информация о перемещениях представлена в виде таблицы, отображающей дату, местоположение и ответственного за перемещение. Такой подход позволяет быстро получить всю необходимую информацию о местонахождении оборудования на различных этапах его эксплуатации.

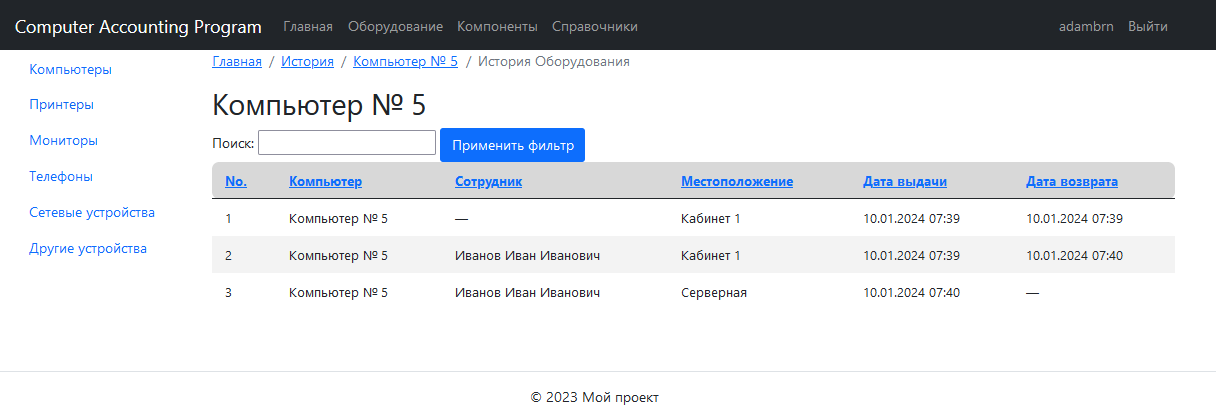


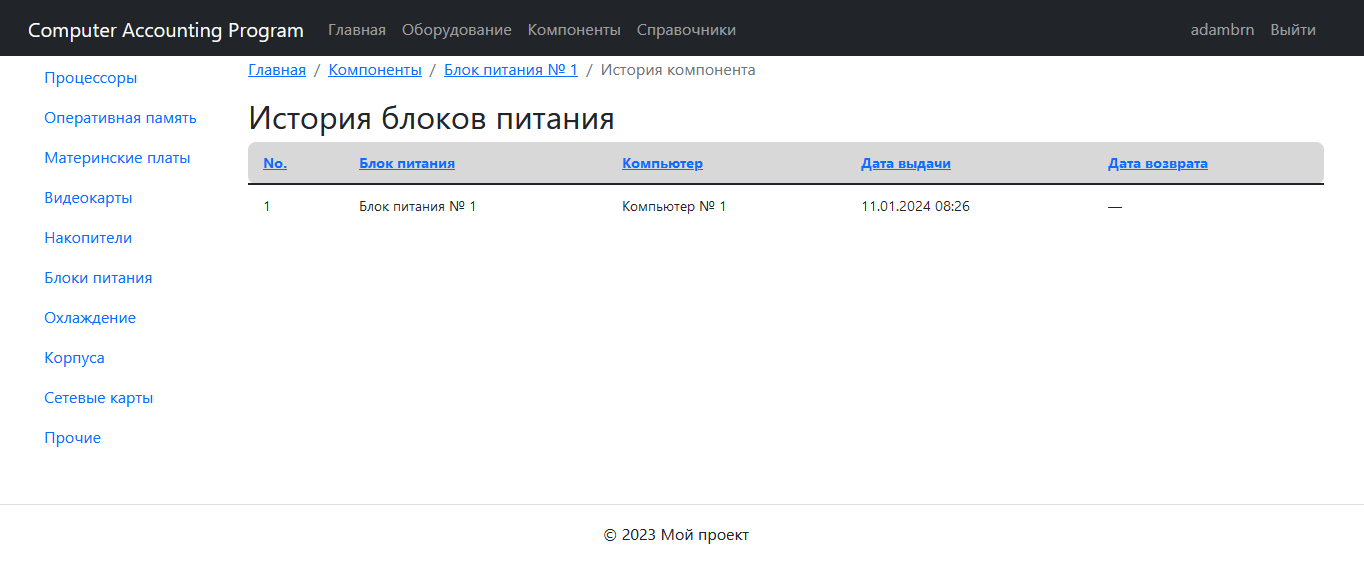
Рисунок 13

**Автоматическое обновление:**

Информация в истории обновляется автоматически при каждом изменении местоположения оборудования в системе. Это обеспечивает актуальность данных и точность отображения истории перемещений.

Реализация функциональности истории перемещений дополняет систему учета компьютерной техники, предоставляя пользователю полную картину о перемещениях оборудования в прошлом. Это важное дополнение, улучшающее прозрачность и контроль в управлении компьютерной техникой.

Так же аналогичным образом реализована история добавления и удаления компонентов в компьютер



Рисунок

В ходе разработки данного веб-приложения для учета компьютерной техники на предприятии были использованы следующие внешние модули и библиотеки:

**django\_tables2** — это Django приложение, предоставляющее возможность создания и отображения таблиц в ваших веб-приложениях. Он упрощает процесс создания динамических таблиц с функциональностью сортировки, фильтрации и другими возможностями. Модуль предоставляет абстракции для определения структуры таблиц и их отображения в шаблонах Django.

**crispy\_forms** — это пакет Django для красивого оформления форм. Модуль позволяет определять формы в Python и легко управлять их визуальным представлением в HTML. Он интегрируется с различными библиотеками стилей, такими как Bootstrap, что упрощает создание элегантных и отзывчивых форм.

**view\_breadcrumbs** — это Django приложение, предназначенное для управления "хлебными крошками" (breadcrumbs) в вашем веб-приложении. Хлебные крошки обеспечивают навигацию по иерархии страниц, предоставляя пользователям понятный путь к текущему местоположению. view\_breadcrumbs упрощает создание и отображение хлебных крошек в шаблонах Django.

**django\_filters** — это Django приложение, предоставляющее инструменты для фильтрации данных в вашем приложении. Модуль обеспечивает создание фильтров для моделей Django, что делает процесс фильтрации результатов запросов к базе данных более гибким и удобным. Он интегрируется с формами Django, обеспечивая простой способ добавления фильтрации к представлениям.

В ходе создания представлений (views) для компонентов были разработаны основные классы, которые обеспечивают общий функционал для обновления и создания компонентов. Данные представления наследуют необходимые миксины и классы модулей, используемых в приложении, чтобы обеспечить соответствие требованиям проекта. Далее были созданы классы-наследники для отдельных моделей каждого из приложений. Описание всей структуры созданных и используемых представлений можно увидеть в приложении.

# Обеспечение безопасности данных и аутентификации пользователей

Обеспечение безопасности данных и аутентификации пользователей в системе учета компьютерной техники осуществляется с использованием встроенных функций Django, что гарантирует надежную защиту конфиденциальности и целостности информации. Использование стандартных компонентов Django упрощает процесс аутентификации, обеспечивает актуальность безопасности и совместимость с последними обновлениями фреймворка.

Вот некоторый из возможностей:

1. Хранение паролей. Пароли пользователей хранятся в системе в зашифрованном виде с использованием хэширования, что предотвращает возможность несанкционированного доступа к учетным данным. Django обеспечивает стандартные методы хэширования паролей, что повышает уровень безопасности системы.
2. Защита от CSRF-атак. Встроенные механизмы защиты от атак на подделку межсайтовых запросов (CSRF) применяются для предотвращения возможных угроз безопасности. Django автоматически генерирует токены CSRF для каждой сессии, обеспечивая защиту от поддельных запросов.
3. Логирование и мониторинг. Встроенные средства Django для логирования событий и мониторинга активности пользователей предоставляют детальную информацию о действиях в системе. Логирование действий пользователей позволяет быстро выявлять и реагировать на потенциальные угрозы безопасности.
4. Защита от SQL-инъекций и других атак. Django предоставляет механизмы защиты от SQL-инъекций и других распространенных видов атак, предотвращая возможные угрозы безопасности базы данных и системы в целом. Встроенные методы работы с базой данных, такие как ORM, способствуют безопасной обработке запросов.
5. Обновления и поддержка. Постоянные обновления и поддержка фреймворка Django гарантируют актуальность и эффективность встроенных средств безопасности.

Внедрение последних версий Django позволяет минимизировать уязвимости и обеспечивает соответствие стандартам безопасности. Общее использование встроенных функций Django для аутентификации и обеспечения безопасности данных обеспечивает стабильную и защищенную среду в системе учета компьютерной техники.

ГЛАВА 4. Тестирование и отладка приложения

# Планирование тестирования и выбор тестовых сценариев

В рамках данной работы предпочтительным вариантом тестирования является комбинированный подход, включающий в себя функциональное и модульное тестирование.

Функциональное тестирование позволит проверить приложение в целом, оценить его работу в реальных сценариях использования, и удостовериться в соответствии функциональных возможностей заявленным требованиям. Это важно для обеспечения корректной работы приложения в конечной среде эксплуатации.

Модульное тестирование направлено на проверку отдельных компонентов и модулей приложения. Этот вид тестирования позволяет выявить и устранить потенциальные проблемы на ранних стадиях разработки, улучшая общую структуру кода и обеспечивая его легкость поддержки и расширения.

Совмещение функционального и модульного тестирования обеспечивает полный охват тестов, начиная от проверки отдельных блоков кода и заканчивая проверкой работы приложения в целом. Это повышает уверенность в качестве и стабильности приложения.

В целом, комбинированный подход к тестированию обеспечивает наилучший баланс между проверкой работы приложения в целом и обнаружением потенциальных проблем на уровне отдельных компонентов.

Тестирование будет охватывать следующие сценарии:

1. Добавление нового оборудования: Тестирование процесса корректного добавления нового оборудования через соответствующую форму, проверка валидации данных.
2. Редактирование информации об оборудовании: Проверка возможности успешного редактирования данных об уже существующем оборудовании, включая изменение статусов, местоположения и других характеристик.
3. Удаление оборудования: Тестирование корректного удаления оборудования из системы с подтверждением удаления.
4. Добавление и удаление компонентов: Проверка функционала добавления и удаления компонентов в составе компьютера.
5. История перемещения оборудования: Тестирование функционала отслеживания истории перемещения оборудования между различными местоположениями.
6. Аутентификация и безопасность: Проверка корректной работы системы аутентификации, защиты от несанкционированного доступа и обеспечение безопасности данных.
7. Тестирование формы поиска и фильтрации: Проверка правильности работы механизма поиска и фильтрации данных в системе.
8. Работа с справочниками: Тестирование функционала работы с каталогами и обеспечение их корректного отображения.

Выбор этих тестовых сценариев обеспечивает полное покрытие основных функций приложения, а также проверку его безопасности и удобства использования.

# Отладка и исправление выявленных ошибок

В процессе разработки приложения "Computer Accounting Program" одиночным разработчиком был применен непрерывный и постоянный подход к тестированию, который обеспечил высокий уровень качества и надежности приложения.

В течение всего процесса разработки проводилось непрерывное тестирование функциональности, включая добавление, редактирование и удаление данных об оборудовании, а также управление компонентами и историей перемещения. Регулярные тесты гарантировали поддержание стабильности функционала на каждом этапе разработки.

Непрерывное тестирование интерфейса направлено на обеспечение удобства пользовательского взаимодействия. В процессе разработки регулярно проверялась интуитивная понятность интерфейса, а также его соответствие заявленным требованиям. Исправления и улучшения вносились непосредственно после выявления недочетов.

Непрерывная проверка безопасности включала в себя аутентификацию пользователей, контроль доступа и защиту от потенциальных угроз. Регулярные тесты на безопасность обеспечивали своевременное выявление и устранение уязвимостей, гарантируя высокий уровень защиты данных.

Отладка проводилась непрерывно в тандеме с разработкой. Выявленные ошибки и недочеты исправлялись незамедлительно, что позволило обеспечить стабильность работы приложения на всех этапах разработки. Процесс отладки включал в себя анализ кода, выявление и устранение багов, а также оптимизацию производительности.

Регулярная оценка результатов тестирования позволяла поддерживать высокое качество приложения и своевременно реагировать на изменения требований. Такой непрерывный подход к тестированию стал ключевым элементом успешной разработки и обеспечил создание надежного и функционального продукта.

ГЛАВА 5. Заключение

В результате выполненной работы по разработке приложения "Computer Accounting Program" можно сделать следующие выводы. Разработанная система учета компьютерной техники успешно соответствует поставленным требованиям и целям, предоставляя пользователям начальный инструмент для управления оборудованием предоставляющий полную возможность дальнейшей разработки программы.

Пользователи имеют возможность легко добавлять, редактировать и удалять оборудование, а также проводить операции по управлению компонентами. Интуитивно понятный интерфейс и непрерывная проверка безопасности гарантируют удовлетворение требований безопасности данных.

Предложения по развитию приложения:

1. Для дальнейшего усовершенствования и расширения функциональности приложения предлагаются следующие шаги:
2. Добавление дополнительных отчетов и аналитики: Внедрение новых отчетов и аналитических инструментов для более детального мониторинга состояния и использования компьютерной техники.
3. Интеграция с внешними системами: Рассмотрение возможности интеграции с другими информационными системами предприятия для автоматизации обмена данными и улучшения взаимодействия.
4. Развитие мобильной версии: Создание мобильного приложения или адаптация существующего интерфейса для мобильных устройств для обеспечения удобства доступа к системе в любое время и из любой точки.
5. Внедрение дополнительных уровней доступа: Расширение системы управления доступом для более гибкой настройки прав и ролей пользователей.
6. Автоматизация резервного копирования и восстановления данных: Разработка механизма автоматического резервного копирования и восстановления данных для обеспечения надежности и безопасности информации.

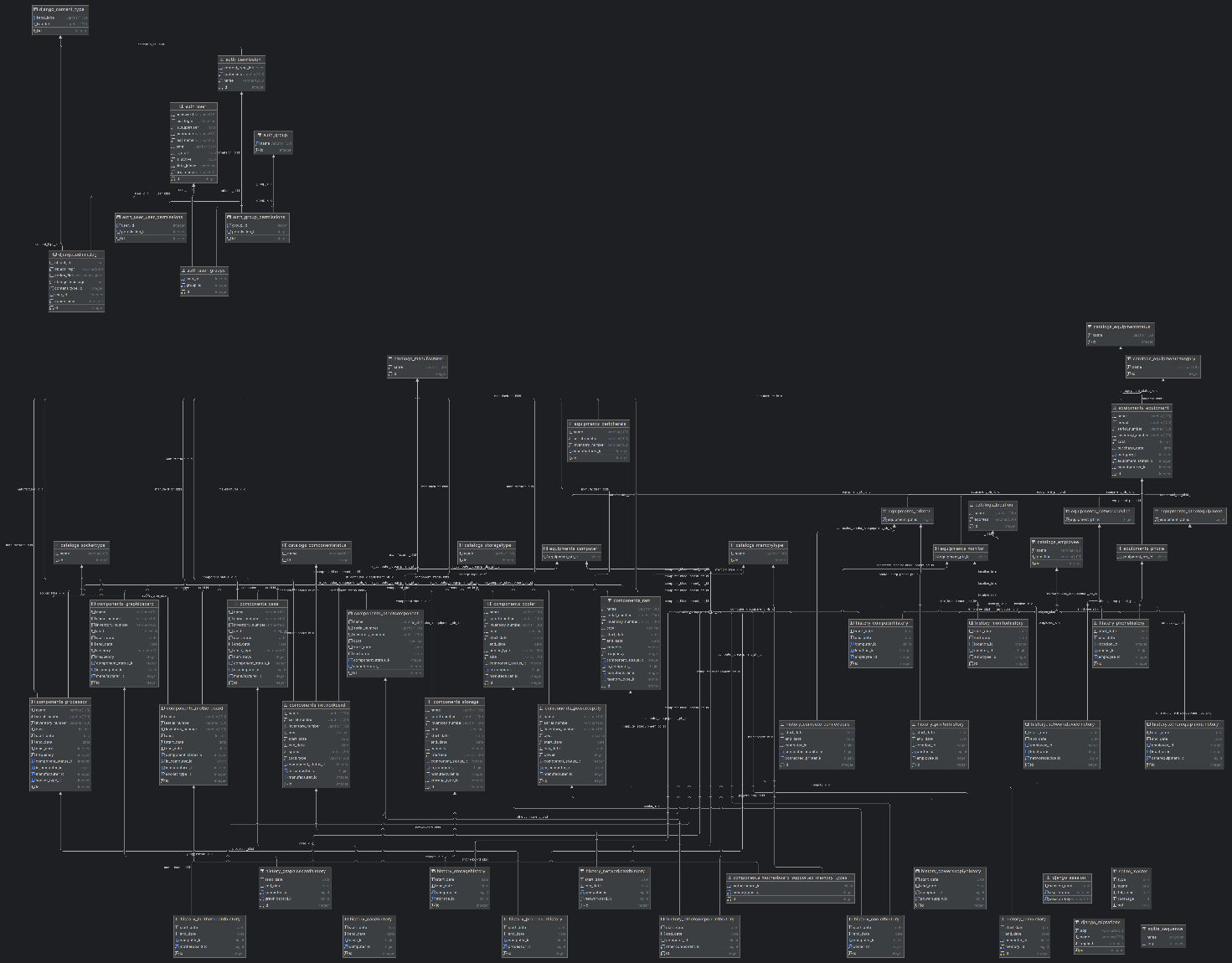
**Итоги**

Разработанная система учета компьютерной техники успешно решает начальные задачи управления оборудованием. Предложенные дальнейшие шаги по развитию системы направлены на улучшение ее возможностей и соответствие растущим потребностям предприятия в учете и мониторинге компьютерной техники.

ПРИЛОЖЕНИЯ

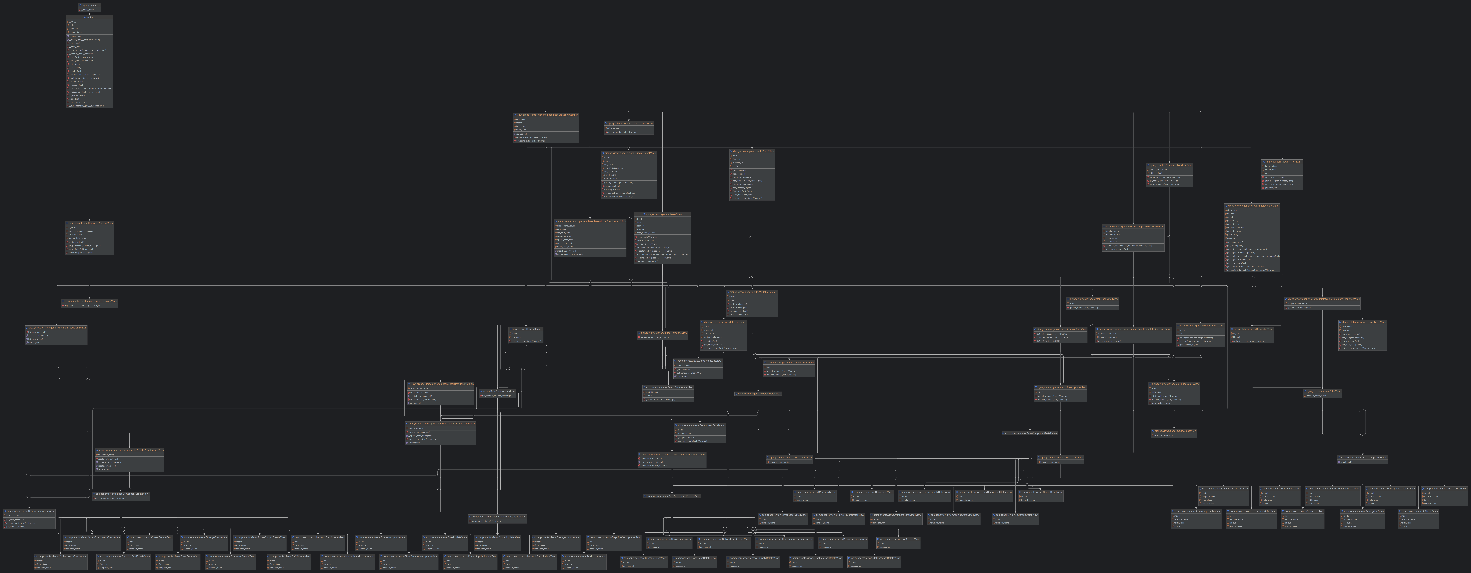
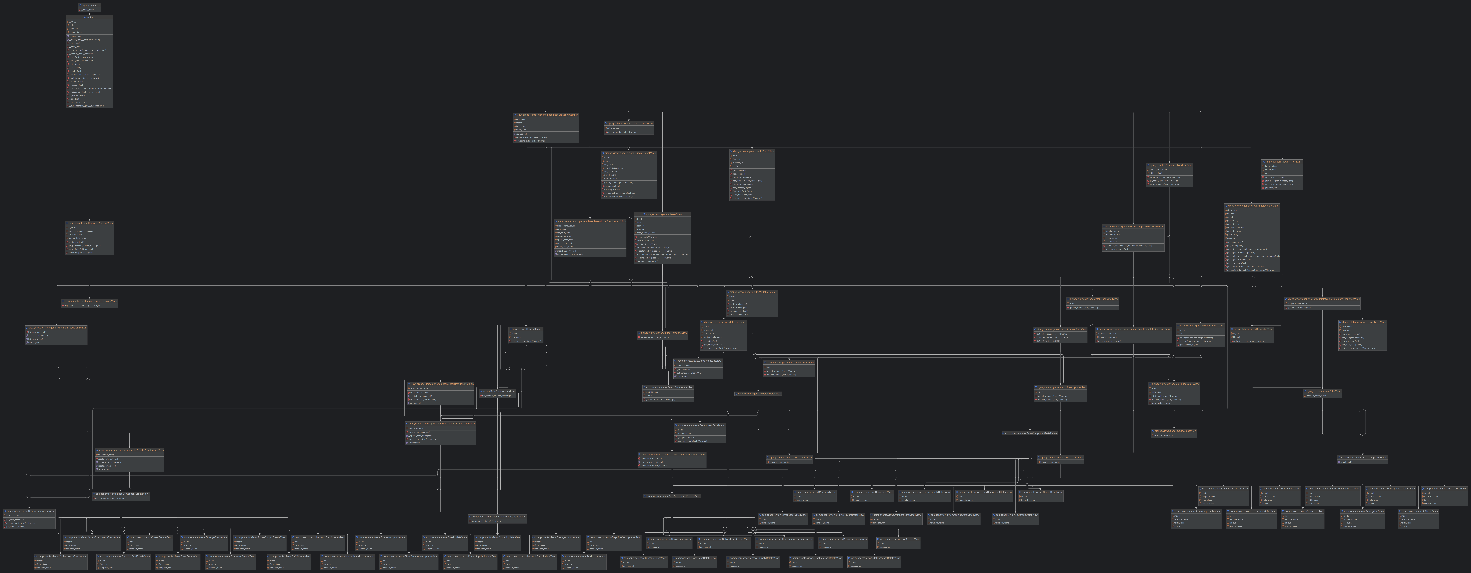
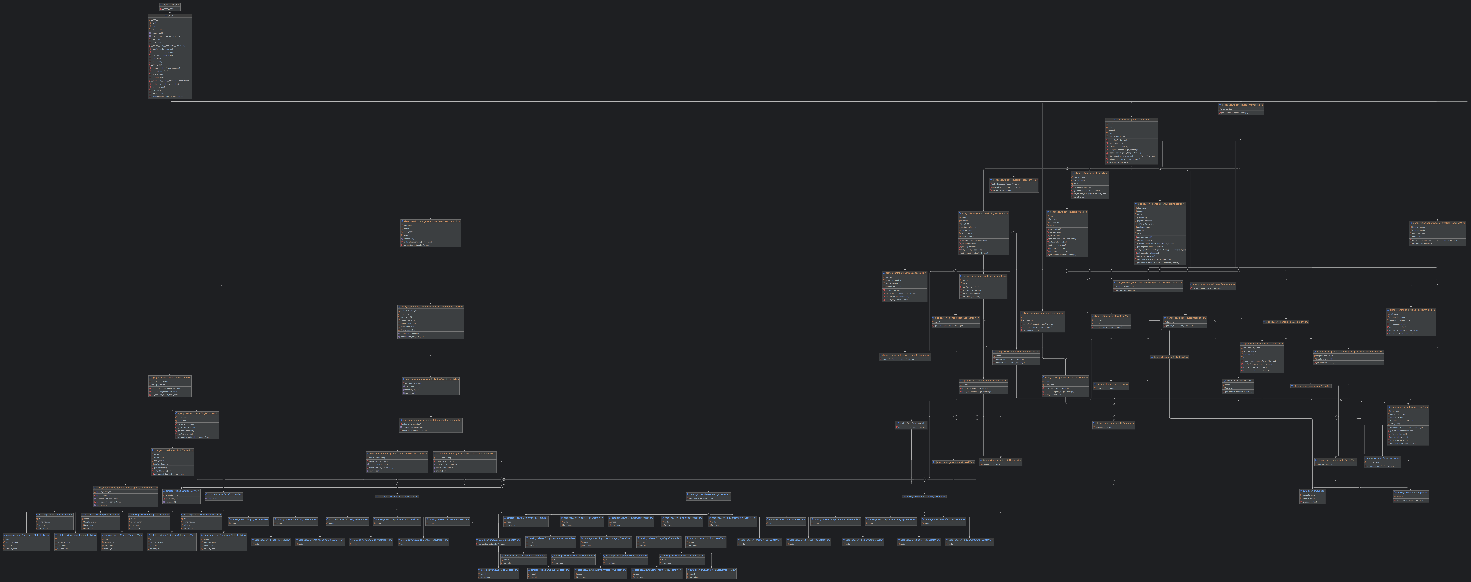
Приложение 1

ERD – базы данных



Приложение 2

UML – Представлений (View) приложений catalogs, components, equipments, history



СПИСОК ЛИТЕРАТУРЫ

1. Лутц М Изучаем Python Том 1 / М Лутц. – СПб : Вильямс, 2019. – 832 с.
2. Бхаргава Адитья, А.Б. Грокаем алгоритмы. Руководство. Иллюстрированное пособие для программистов и любопытствующих /

А.Б. Бхаргава Адитья. – Санкт-Петербург : Издательский Дом ПИТЕР, 2022. – 288 с.

1. Форсье Дж Django. Разработка веб-приложений на Python / Дж Форсье, П Биссекс, У Чан. – Санкт-Петербург : Символ-Плюс, 2018. – 456 с.
2. Постолит, А. Python, Django и PyCharm для начинающих /

А. Постолит. – Санкт - Петербург : БХВ, 2021. – 464 с.

1. Бейдер Д. Чистый Python. Тонкости программирования для профи / Д. Бейдер. – Спб : Питер, 2022. – 288 с.
2. Официальный сайт продукта Python [Электронный ресурс] /

https://www.python.org. – Режим доступа: <https://www.python.org/doc/>

1. Официальная документация Django 4 [Электронный ресурс] / https://docs.djangoproject.com/en/4.0/. – Режим доступа: <https://docs.djangoproject.com/en/4.0/>
2. Официальная документация django-tables2 [Электронный ресурс] / https://django-tables2.readthedocs.io/. – Режим доступа: <https://django-tables2.readthedocs.io/>
3. Официальная документация crispy-forms [Электронный ресурс] / https://django-crispy-forms.readthedocs.io/. – Режим доступа: <https://django-crispy-forms.readthedocs.io/>
4. Официальная документация crispy-bootstrap5 [Электронный ресурс] / https://django-crispy-bootstrap5.readthedocs.io/. – Режим доступа: <https://django-crispy-bootstrap5.readthedocs.io/>
5. Официальная документация view-breadcrumbs [Электронный ресурс] / https://django-view-breadcrumbs.readthedocs.io/. – Режим доступа: <https://django-view-breadcrumbs.readthedocs.io/>
6. Официальная документация django-filters [Электронный ресурс] / https://django-filter.readthedocs.io/. – Режим доступа: <https://django-filter.readthedocs.io/>
7. Официальная документация Bootstrap 5 [Электронный ресурс] / https://getbootstrap.com/– Режим доступа: <https://getbootstrap.com/>