geekbrains

Программист Python Цифровые профессии

Разработка web-приложения для учета компьютерной техники на предприятии

Перемышленников Алексей Сергеевич

Барнаул

2023

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc155717660)

[ВВЕДЕНИЕ 3](#_Toc155717661)

[ГЛАВА 1. Основы разработки web-приложений с использованием Django 7](#_Toc155717662)

[1.1 Введение в Django и его преимущества 7](#_Toc155717663)

[1.2 Архитектура MVC (Model-View-Controller) и паттерн MTV (Model-Template-View) 9](#_Toc155717664)

[1.3 Основы работы с базами данных в Django (ORM) 12](#_Toc155717665)

[1.4 Создание и управление веб-приложением в Django 14](#_Toc155717666)

[ГЛАВА 2. Анализ и проектирование системы учета компьютерной техники 17](#_Toc155717667)

[2.1 Анализ требований к web-приложению для учета компьютерной техники 17](#_Toc155717668)

[2.2 Проектирование базы данных для хранения информации о компьютерной технике 20](#_Toc155717669)

[2.3 Определение функциональных возможностей приложения 26](#_Toc155717670)

[ГЛАВА 3. Разработка web-приложения с использованием Django 28](#_Toc155717671)

[3.1 Создание основных моделей данных при помощи Django ORM 28](#_Toc155717672)

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.

Обоснование темы проекта заключается в растущей важности компьютерной техники в современных бизнес-процессах. Чтобы обеспечить эффективную работу и минимизировать риски, необходимо иметь учет и контроль над всей компьютерной техникой на предприятии.

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. Оценка производительности и оптимизация, включая анализ производительности приложения и применение оптимизационных методов для улучшения скорости и отклика.
7. 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, мы определяем структуру таблицы в базе данных. Такой подход позволяет абстрагировать базу данных от кода приложения и работать с данными в объектно-ориентированной парадигме.

2 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.

3 Административная панель Django

Для управления данными, хранящимися в базе данных, Django предоставляет готовую административную панель. Она автоматически создается на основе определенных моделей и позволяет администраторам приложения удобно выполнять операции CRUD (Create, Read, Update, Delete). Административная панель упрощает мониторинг и управление данными, что особенно важно для системы учета компьютерной техники.

4 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.

5 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 мы 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.

2 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.

3 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.

4 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.

5 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.

6 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.

7 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.

8 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.

# 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 мы начнем разработку основных моделей данных для нашего web-приложения с использованием Django ORM. Мы 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) |

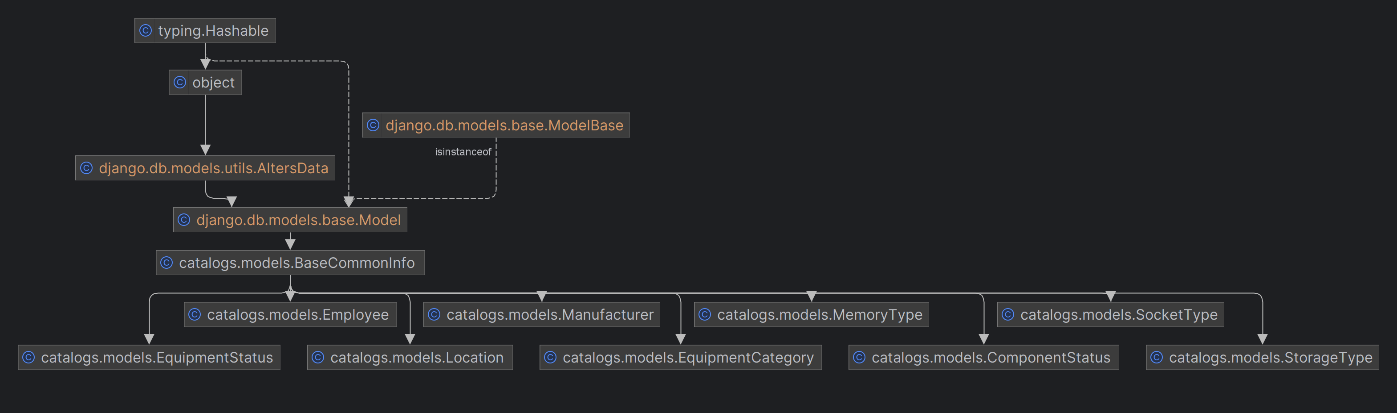


Рисунок 1

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 | - |

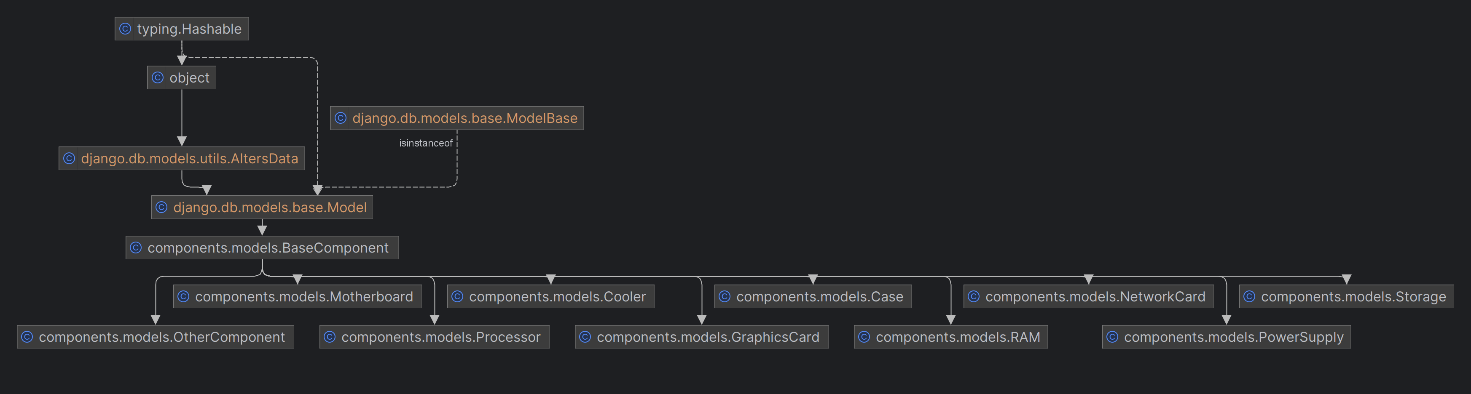


Рисунок 2

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) |

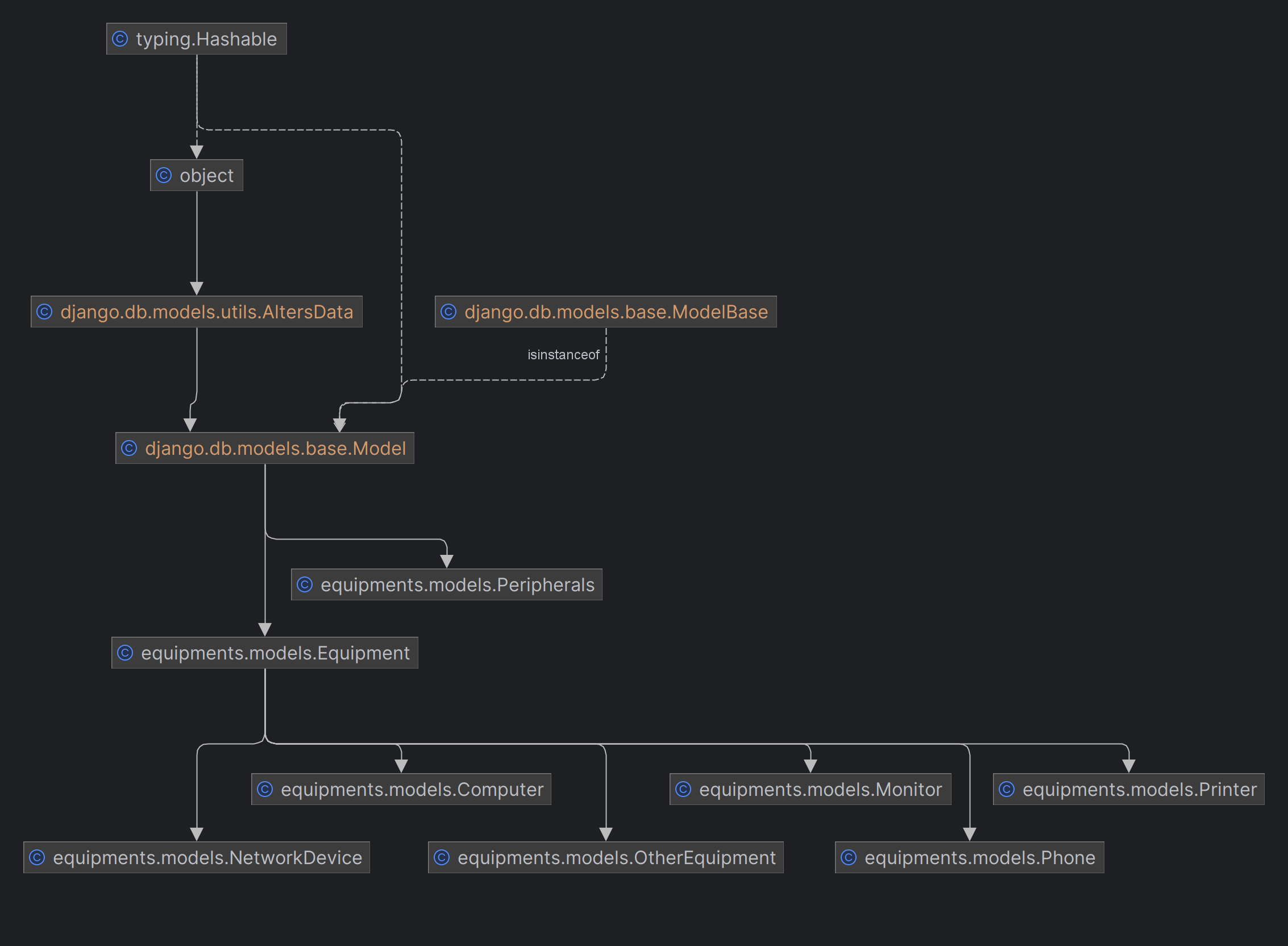


Рисунок 3

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) |

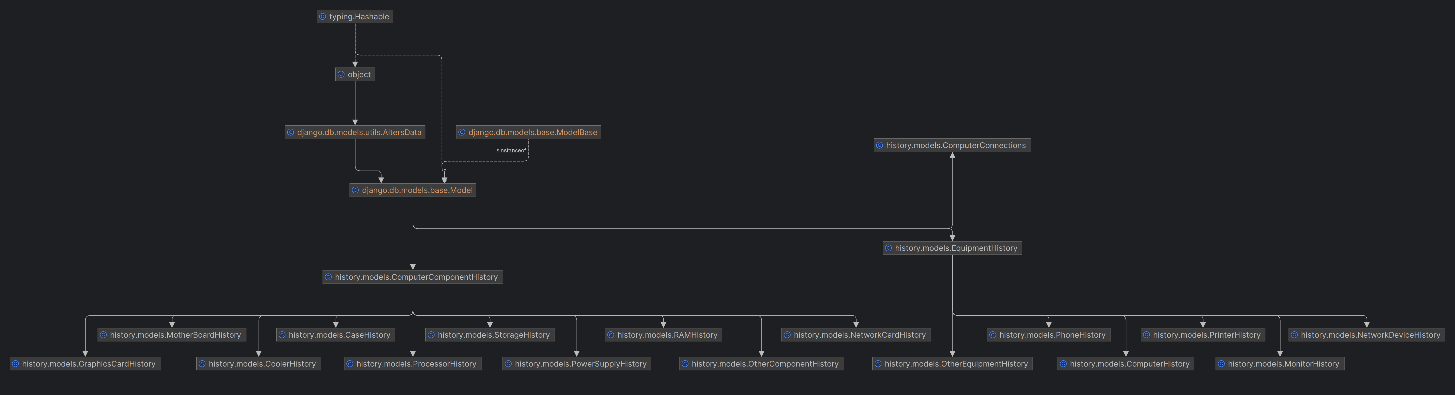


Рисунок 4