**Министерство образования Иркутской области**

Государственное бюджетное профессиональное образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
| **ДП. 09.02.07–5.25.212.05. ПЗ** | УТВЕРЖДАЮ  Зам. директора по УР, к.т.н.  \_\_\_\_\_\_\_\_\_\_\_Коробкова. Е.А |

**Информационная система**

**«Медицинская система учета»**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Нормоконтролер: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Е.С. Кудрявцева) |
| Консультант по экономической части: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.П. Юргина) |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.И. Тирский) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.С. Мурзин) |

Иркутск 2025

Содержание

[Введение 3](#_Toc199106348)

[1 Предпроектное исследование 4](#_Toc199106349)

[2 Анализ инструментов для разработки ПО 6](#_Toc199106350)

[3 Техническое задание 18](#_Toc199106351)

[4 Проектирование ИС 19](#_Toc199106352)

[4.1 Структурная схема ИС 19](#_Toc199106353)

[4.2 Проектирование базы данных 23](#_Toc199106354)

[4.3 Проектирование интерфейса 28](#_Toc199106355)

[5 Разработка ИС 30](#_Toc199106356)

[5.1 Разработка интерфейса ИС 30](#_Toc199106357)

[5.2 Разработка базы данных ИС 33](#_Toc199106358)

[5.3 Разработка серверной части 37](#_Toc199106359)

[5.4 Тестирование информационной системы 39](#_Toc199106360)

[6 Документирование информационной системы 43](#_Toc199106361)

[6.1 Руководство пользователя информационной системы 43](#_Toc199106362)

[7.2 Руководство администратора информационной системы 51](#_Toc199106363)

[7.3 Руководство службы поддержки информационной системы 55](#_Toc199106364)

[Заключение 56](#_Toc199106365)

[Список использованных источников 58](#_Toc199106366)

[Приложение А – Техническое задание 60](#_Toc199106367)

[Приложение Б – Листинг таблицы пользователей 66](#_Toc199106368)

Введение

Современный мир невозможно представить без информационных технологий, которые проникают во все сферы человеческой деятельности. Одной из таких сфер является управление и организация данных, что особенно актуально для учреждений здравоохранения. В данном контексте разработка информационных систем становится неотъемлемой частью оптимизации бизнес-процессов, повышения эффективности работы и улучшения оборудования.

Дипломный проект посвящён разработке информационной системы «Медицинской система учёта», которая предназначена для автоматизации процессов учёта, хранения и обработки данных в медицинском учреждении. Актуальность данной работы обусловлена необходимостью упрощения рутинных операций, снижения вероятности ошибок при обработке информации об оборудовании и повышения скорости доступа к данным.

Целью проекта является создание информационной системы, которая позволит эффективно управлять данными об оборудовании. Для достижения поставленной цели необходимо решить следующие задачи:

* Провести анализ предметной области и определить требования к системе.
* Разработать структуру базы данных и интерфейс пользователя.
* Реализовать функционал системы с использованием современных технологий программирования.
* Протестировать систему на предмет корректности работы и удобства использования.

В результате выполнения дипломного проекта будет создана информационная система, которая сможет стать полезным инструментом для медицинских учреждений, а также послужит примером успешного применения современных технологий в решении реальных задач в области здравоохранения.

1 Предпроектное исследование

**1.1 Описание предметной области**

Предметной областью дипломного проекта является разработка информационной системы для медицинского учреждения, специализирующегося на учёте и управлении данными об оборудовании. В условиях современного здравоохранения эффективное управление информацией становится ключевым фактором для повышения качества обслуживания пациентов и оптимизации работы медицинского персонала. Учитывая специфику работы учреждения, возникает необходимость в автоматизации процессов учёта, обработки и хранения данных для оборудований.

Основные процессы, которые охватывает предметная область, включают:

1. Учёт оборудования: создание базы данных о медицинских устройствах, включая их характеристики (тип, модель, производитель), состояние, дату приобретения и срок службы.
2. Мониторинг состояния: отслеживание технического состояния оборудования, планирование профилактических осмотров и ремонтов, а также управление запасными частями.
3. Управление запасами: контроль наличия и распределения медицинского оборудования по отделениям, а также учет его использования.
4. Аналитика и отчётность: формирование отчетов по использованию оборудования, затратам на обслуживание и ремонты, а также анализ эффективности работы.
5. Разработка информационной системы «Медицинская система учета» направлена на автоматизацию перечисленных процессов, что позволит:

* сократить время на выполнение рутинных операций;
* минимизировать ошибки, связанные с человеческим фактором;
* повышение эффективности управления медицинским оборудованием.
* улучшение взаимодействия между отделами, ответственными за техническое обслуживание и эксплуатацию.

Выделены основные роли пользователей и их функционал:

1. Администратор:

* Добавление, редактирование и удаление учетных записей.
* Назначение ролей.
* Добавление, редактирование и удаление оборудования.
* Импортирование данных об оборудовании.
* Формирование отчетов об оборудовании.
* Добавление и удаление PDF файлов для оборудования.
* Просмотр истории оборудования.

1. Старшая медицинская сестра:

* Добавление и редактирование оборудования
* Импортирование данных об оборудовании.
* Формирование отчетов об оборудовании.
* Добавление и удаление PDF файлов для оборудования.

1. Медицинская сестра:

* Добавление и редактирования оборудования
* Формирование отчетов об оборудовании
* Просмотр таблицы оборудования.
* Просмотр карточки оборудования.

1. Пользователь:

* Формирование отчетов об оборудовании.
* Просмотр таблицы оборудования.
* Просмотр карточки оборудования.

# 2 Анализ инструментов для разработки ПО

Для разработки программного продукта в рамках производственной практики был проведён анализ инструментов, используемых на этапах проектирования, разработки, хранения данных и интеграции клиентской и серверной частей.

Для проектирования структуры и дизайна программного продукта использовался сервис Draw.io:

Draw.io – это бесплатное онлайн-приложение, предназначенное для создания диаграмм, схем и блок-схем. В проекте Draw.io используется для проектирования структурных диаграмм, контекстных диаграмм и создания прототипов страниц.

Программный продукт состоит из клиентской и серверной части. В клиентской части используются HTML, CSS и JavaScript, в серверной части будет использован язык Python с фреймворком Django, а также база данных MySQL.

HTML – это язык гипертекстовой разметки, который используется для создания структуры веб-страниц. Он позволяет разрабатывать элементы интерфейса и взаимодействия.

CSS – это каскадные таблицы стилей, которые используются для определения оформления и дизайна программного продукта.

JavaScript – это скриптовый язык программирования, который используется для создания динамических элементов на веб-страницах и взаимодействия с пользователем. JavaScript широко используется для улучшения интерактивности веб-приложений.

Для взаимодействия базы данных и клиентской части используется серверный язык. Для реализации серверной части проекта были рассмотрены два языка программирования: Python и PHP:

Python – высокоуровневый язык программирования, который применяется в различных областях, включая веб-разработку. Он обладает следующими характеристиками:

* Объектно-ориентированный подход: Python поддерживает объектно-ориентированное программирование, что позволяет структурировать код и облегчает его поддержку.
* Наличие библиотек: Python имеет множество библиотек и фреймворков для веб-разработки, таких как Django и Flask. Это позволяет быстро создавать веб-приложения.
* Простота синтаксиса: Python известен своей простотой и лаконичностью кода, что делает его понятным даже для новичков.
* Интеграция с базами данных: Python имеет инструменты для работы с различными базами данных, включая MySQL, PostgreSQL и SQLite.
* Более активное сообщество: Python имеет большое сообщество разработчиков, что позволяет быстро находить решения проблем и получать помощь.

Однако Python имеет и свои недостатки:

* Меньшая скорость работы: Python может быть медленнее по сравнению с компилируемыми языками, что может влиять на производительность веб-приложения.
* Большое потребление памяти: Python потребляет больше памяти, что может быть критичным при создании масштабируемых приложений.
* Необходимость фреймворка: для работы с backend в веб-приложениях Python требует фреймворка, что может усложнить разработку.

JavaScript– широко используемый язык программирования для создания как клиентской, так и серверной части веб-приложений. Его основные характеристики:

* Асинхронная модель: поддерживает асинхронные операции, что делает его эффективным для приложений с высокой нагрузкой.
* Использование одного языка: JavaScript используется как на клиентской, так и на серверной стороне, что упрощает разработку и поддержание кода.
* Интеграция с базами данных: JavaScript легко интегрируется с различными базами данных, включая NoSQL (MongoDB) и SQL (MySQL, PostgreSQL).
* Широкая экосистема: JavaScript и Node.js имеют обширную библиотеку пакетов через NPM, что ускоряет разработку приложений.

Недостатки:

* Однопоточность: по умолчанию Node.js работает в однопоточном режиме, что может создавать сложности для некоторых задач.
* Необходимость фреймворка – при работе с JavaScript требуется использовать бэкенд-фреймворк.
* Меньшая распространенность в крупных корпоративных системах: хотя Node.js широко используется в веб-разработке, его часто избегают в корпоративных приложениях.

PHP - скриптовый язык, широко используемый в веб-разработке. Его ключевые особенности включают:

* Популярность в веб-разработке: PHP является одним из лидеров среди языков для создания динамических веб-сайтов.
* Встроенная поддержка HTTP Cookies: PHP позволяет легко работать с сессиями, что упрощает создание пользовательских сеансов и авторизации.
* Совместимость с большинством хостинг-провайдеров: PHP поддерживается практически всеми хостинг-провайдерами, что делает его универсальным.
* Простота использования: PHP интуитивно понятен и легко интегрируется с различными веб-технологиями.
* Лёгкий синтаксис: PHP имеет простой и понятный синтаксис, что облегчает разработку и поддержку веб-приложений.

Однако PHP также имеет свои недостатки:

* Ограниченные объектно-ориентированные возможности: хотя PHP поддерживает объектно-ориентированное программирование, его возможности могут быть ограничены по сравнению с Python.
* Отсутствие строгой типизации: PHP имеет динамическую типизацию, что может привести к ошибкам в коде.
* Ограниченные инструменты для работы с базами данных: PHP предлагает основные инструменты для работы с базами данных, но их функциональность может быть менее широкой, чем у Python.

Для наглядного сравнения языков была составлена таблица 1:

Таблица 1 – Сравнение языков программирования

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | Python | JavaScript | Php |
| Объектно-ориентированный подход | + | + | + |
| Наличие библиотек | + | + | - |
| Лёгкий синтаксис | + | + | + |
| Интеграция с базами данных | + | + | + |
| Встроенная поддержка HTTP Cookies | + | - | + |
| Поддержка большинства хостинг-провайдеров | + | + | + |
| Простота использования | + | + | + |
| Поддержка веб-фреймворков | + | + | + |

Для разработки серверной части проекта был выбран язык Python. Однако для более удобной и структурированной разработки необходимо выбрать фреймворк, который обеспечит упрощение работы с базой данных, маршрутизацией, аутентификацией и другими аспектами программного продукта. Рассмотрим три наиболее популярных Python-фреймворка: Django, Flask и FastAPI.

Для хранения данных в программном продукте необходимо выбрать систему управления базами данных (СУБД). В качестве возможных вариантов были рассмотрены MySQL, PostgreSQL и SQLite.

MySQL – реляционная система управления базами данных, известная своей простотой и высокой производительностью. Основные характеристики MySQL:

* Популярность: MySQL - одна из самых популярных СУБД в мире, что обеспечивает широкую поддержку и доступность ресурсов для разработчиков.
* Высокая производительность: MySQL оптимизирована для быстрого выполнения запросов и обработки больших объёмов данных.
* Поддержка различных типов данных: MySQL поддерживает множество типов данных, включая числовые, строковые, даты и другие.
* Отказоустойчивость: MySQL имеет встроенные механизмы резервного копирования и репликации, что обеспечивает высокую надёжность.
* Совместимость: MySQL совместима с большинством серверных языков, включая PHP, Python и другие.

Недостатки MySQL:

* Необходимость удалённого сервера: для работы MySQL требуется отдельный сервер, что может усложнить установку и настройку.
* Недостаточная гибкость: в некоторых случаях MySQL может быть менее гибкой, чем другие СУБД.

PostgreSQL – объектно-реляционная система управления базами данных, известная своей расширенной функциональностью. Основные характеристики PostgreSQL:

* Объектно-реляционный подход: PostgreSQL поддерживает объектно-реляционные функции, позволяющие наследовать таблицы и выполнять операции с объектами.
* Совместимость со стандартами SQL: PostgreSQL максимально соответствует стандарту SQL, что обеспечивает большую гибкость.
* Отказоустойчивость: PostgreSQL обладает высокими возможностями резервного копирования и репликации.
* Широкий набор инструментов: PostgreSQL предлагает различные инструменты для управления базой данных и её интеграции с другими системами.

Недостатки PostgreSQL:

* Сложность использования: PostgreSQL может быть сложнее в установке и настройке, чем MySQL.
* Снижение производительности: PostgreSQL может быть менее эффективной в обработке больших объёмов данных.

SQLite – встраиваемая СУБД, отличающаяся компактностью и простотой использования. Основные характеристики SQLite:

* Компактность: SQLite хранит данные в одном файле, что облегчает копирование и переносимость.
* Легкость использования: SQLite не требует отдельного сервера, что упрощает установку и настройку.
* Быстрота работы: SQLite оптимизирована для высокой скорости выполнения запросов.

Недостатки SQLite:

* Ограниченные возможности: SQLite имеет ограниченное количество типов данных и не поддерживает сложные функции, как в других СУБД.
* Отсутствие отказоустойчивости: SQLite не предоставляет механизмы репликации и резервного копирования, что может снизить надёжность.

Сравнение СУБД приведено в таблице 3.

Таблица 2 – Сравнение СУБД

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | MySQL | PostgreSQL | SQLite |
| Поддержка различных типов данных | + | + | - |
| Популярность | + | - | + |
| Отказоустойчивость | + | + | - |
| Не требует удалённого сервера | - | - | + |
| Простота использования | + | - | + |
| Высокая производительность | + | - | + |

Исходя из этого сравнения, для реализации программного продукта была выбрана СУБД MySQL. MySQL обеспечивает высокую производительность, отказоустойчивость и совместимость с различными серверными языками. Несмотря на то, что для работы MySQL требуется удалённый сервер, его популярность и простота использования делают его предпочтительным выбором для реализации базы данных информационной системы.

Для разработки были рассмотрены несколько IDE: Visual Studio, Visual Studio Code, PyCharm и Atom.

Visual Studio Code – лёгкая и гибкая IDE, разработанная Microsoft. Она обладает следующими характеристиками:

* Поддержка множества языков программирования: Visual Studio Code поддерживает различные языки, включая HTML, CSS, JavaScript, PHP, Python и другие.
* Интеграция с системами контроля версий: Visual Studio Code имеет встроенную поддержку Git и позволяет легко управлять репозиториями.
* Возможность расширения функционала: Visual Studio Code предлагает широкий набор расширений, которые позволяют добавлять функции и настраивать IDE под конкретные потребности.
* Мультиплатформенность: Visual Studio Code работает на различных платформах, включая Windows, macOS и Linux.
* Лёгкий синтаксис и интуитивный интерфейс: IDE имеет простой и понятный интерфейс, что делает её удобной для разработчиков.

Недостатки Visual Studio Code:

* Не такой функциональный, как другие IDE: Visual Studio Code может быть менее функциональным по сравнению с более тяжелыми IDE, такими как Visual Studio.
* Нуждается в настройке: для некоторых задач требуется установка дополнительных расширений.

PyCharm – мощная IDE, специально предназначенная для разработки на Python. Ключевые особенности PyCharm:

* Полнофункциональная IDE для Python: PyCharm включает все необходимые инструменты для разработки на Python.
* Интеграция с системами контроля версий: PyCharm поддерживает Git и другие системы контроля версий.
* Поддержка различных платформ: PyCharm доступен для Windows, macOS и Linux.
* Возможность расширения функционала: PyCharm предлагает плагины и расширения для добавления новых функций.

Недостатки PyCharm:

* Требовательность к ресурсам: PyCharm может быть ресурсоемким и неудовлетворительно работать на медленных на слабых компьютерах.
* Ограничения в бесплатной версии: некоторые функции доступны только в платной версии PyCharm.

Visual Studio – универсальная IDE, разработанная Microsoft, которая поддерживает множество языков программирования. Основные характеристики Visual Studio:

* Поддержка множества языков: Visual Studio работает с различными языками, включая C#, C++, JavaScript, Python и другие.
* Интеграция с другими продуктами Microsoft: Visual Studio тесно интегрируется с продуктами Microsoft, такими как Azure и Visual Studio Team Services.

Недостатки Visual Studio:

* Требовательность к ресурсам: Visual Studio может быть ресурсоемким и неудовлетворительно работать на медленных на слабых компьютерах.
* Сложность использования: из-за большого функционала Visual Studio может быть сложной в использовании для новичков.
* Отсутствие поддержки Linux: Visual Studio работает только на Windows.

Таблица 3 – Сравнение IDE

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | VS Code | PyCharm | Visual Studio |
| Поддержка множества языков | + | - | + |
| Интеграция с Git | + | + | + |
| Высокая производительность | + | - | - |
| Возможность расширения функционала | + | + | - |
| Мультиплатформенность | + | + | - |
| Лёгкость использования | + | - | - |
| Лёгкость и гибкость | + | - | - |

Таким образом, после рассмотрения вариантов средств разработок, было принято решение использовать Visual Studio Code.

1. Тем не менее, Visual Studio Code обладает преимуществами:
2. Поддержка всех современных языков для web-разработки, таких как: JavaScript, HTML/CSS, TypeScript;
3. Встроенная система контроля версий Git;
4. Поддержка популярных веб-фреймворков, таких как Django;
5. Поддержка большинства ОС: Windows, Linux и MacOS;
6. Встроенные инструменты для работы с базами данных (Доступ к Oracle, SQL Server, PostgreSQL, MySQL и другим базам данных);
7. После рассмотрения всех инструментов разработки для проекта были выбраны HTML5, CSS3, JS, SQLite и Python-фреймворк Django, а инструменты для проектирования и разработки информационной системы – Draw.io, MySQL Workbench и Visual Studio Code.

3 Техническое задание

В начале разработки создавалось техническое задание, в котором указывались основные требования.

Для создания технического задания использовался стандарт ГОСТ 34.602-2020.

Согласно ГОСТ 34.602-2020 техническое задание должно включать следующие разделы:

1. Введение.
2. Основания для разработки.
3. Назначение системы.
4. Требования к системе.
5. Требования к техническому обеспечению.
6. Требования к программному обеспечению.
7. Организационно-технические требования.

Техническое задание на разработку программного продукта представлено в Приложении А.

4 Проектирование ИС

4.1 Структурная схема ИС

Одним из важнейших этапов разработки информационной системы является проектирование диаграмм, которые помогают лучше понять структуру системы и её работоспособность в целом.

Диаграмма вариантов использования отражает отношения между актёрами и прецедентами, являясь составной частью модели прецедентов.

Актёры:

* Администратор: Пользователь с расширенными правами, который может управлять контентом (добавлять, изменять, удалять оборудование и пользователей).
* Пользователь: Пользователь может просматривать информацию об оборудовании, историю, добавлять оборудование.

На рисунке 1 представлена диаграмма прецедентов Use Case, являющаяся визуальным отображением функциональных требований информационной системы. Диаграмма описывает взаимодействие различных пользователей (актеров) с системой, а также реакцию системы на их действия.

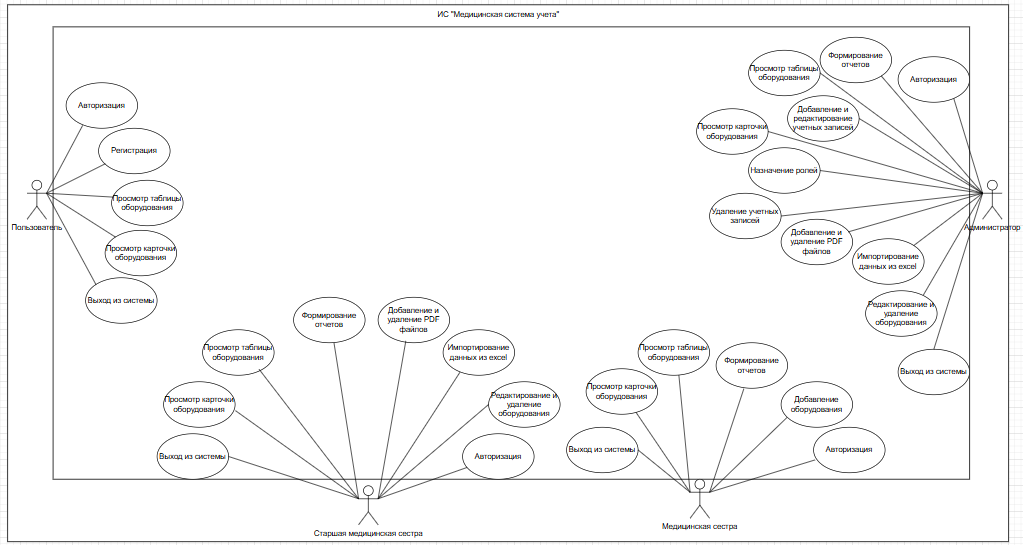


Рисунок 1 - Диаграмма прецедентов Uses CASE

Диаграмма деятельности (рисунок 2) описывает процесс перехода потока управления между действиями в рамках системы. В ней представлена последовательность операций, выполняемых пользователями и системой, а также условия, определяющие возможные переходы между действиями. Данная диаграмма способствует анализу реакции системы на взаимодействие с пользователями и определению последовательности шагов, необходимых для достижения заданных результатов.

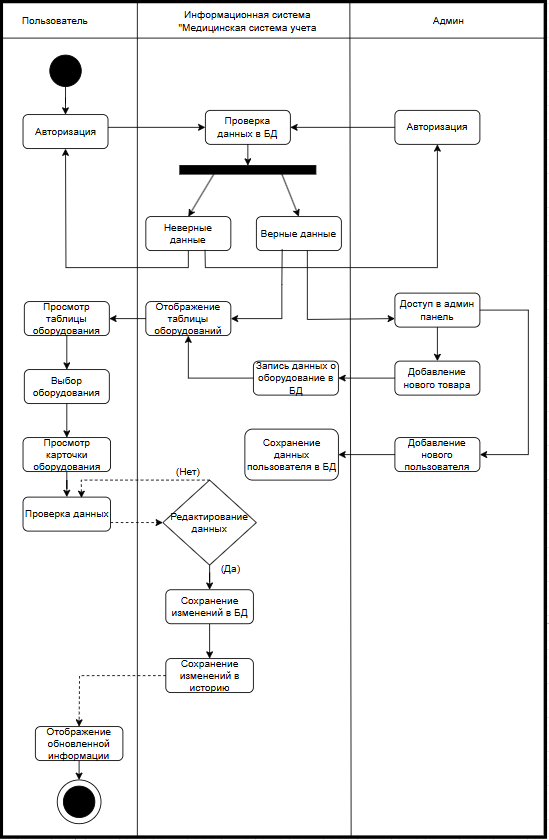


Рисунок 2 – Диаграмма деятельности

Контекстная диаграмма IDEF0 (A0) представляет собой высокоуровневое описание функциональных возможностей информационной системы.

На Рисунке 3 – контекстная диаграмма IDEF0 показаны ключевые процессы. Диаграмма также иллюстрирует, как взаимодействуют основные компоненты системы: пользователи вводят данные через интерфейс, которые затем обрабатываются системой для обновления базы данных.

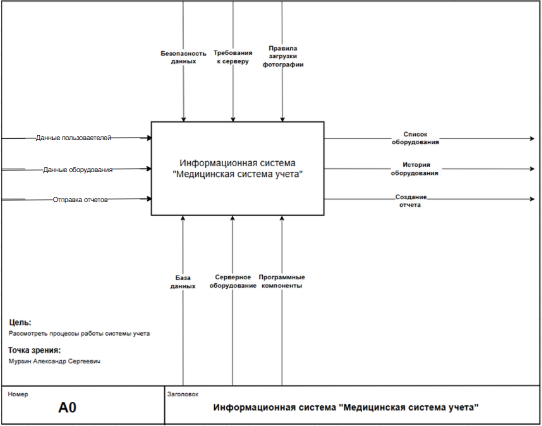


Рисунок 3 – Контекстная диаграмма

Диаграмма декомпозиции A1, представленная на рисунке 4, раскрывает и детализирует ключевые процессы информационной системы. На уровне действий, необходимых для их выполнения.



Рисунок 4 – Диаграмма декомпозиции А1

Диаграмма потоков данных (DFD) представляет собой графическое представление потоков данных в системе, отражающее как данные передаются, обрабатываются и хранятся. DFD диаграмма позволяет моделировать взаимодействие между различными компонентами системы, включая пользователей, базы данных и веб-серверы.

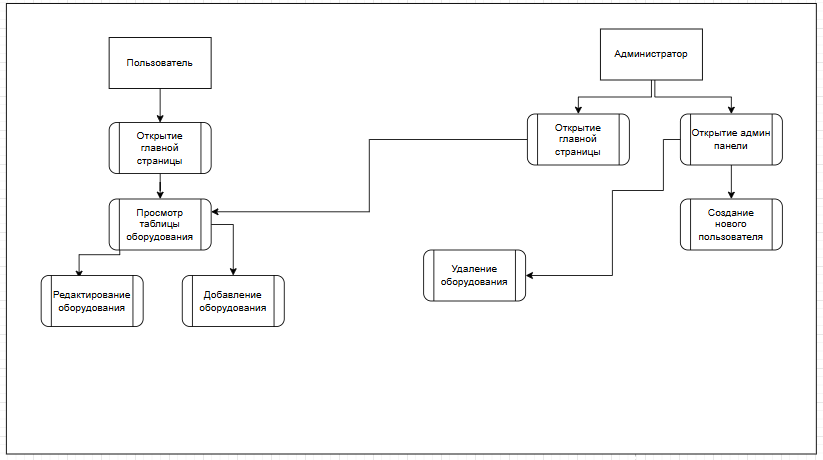


Рисунок 5 – Диаграмма потоков данных

4.2 Проектирование базы данных

Проектирование базы данных начинается с концептуального проектирование базы данных.

Концептуальное проектирование – построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Далее происходит преобразование концептуальной модели в логическую модель, по формальным правилам.

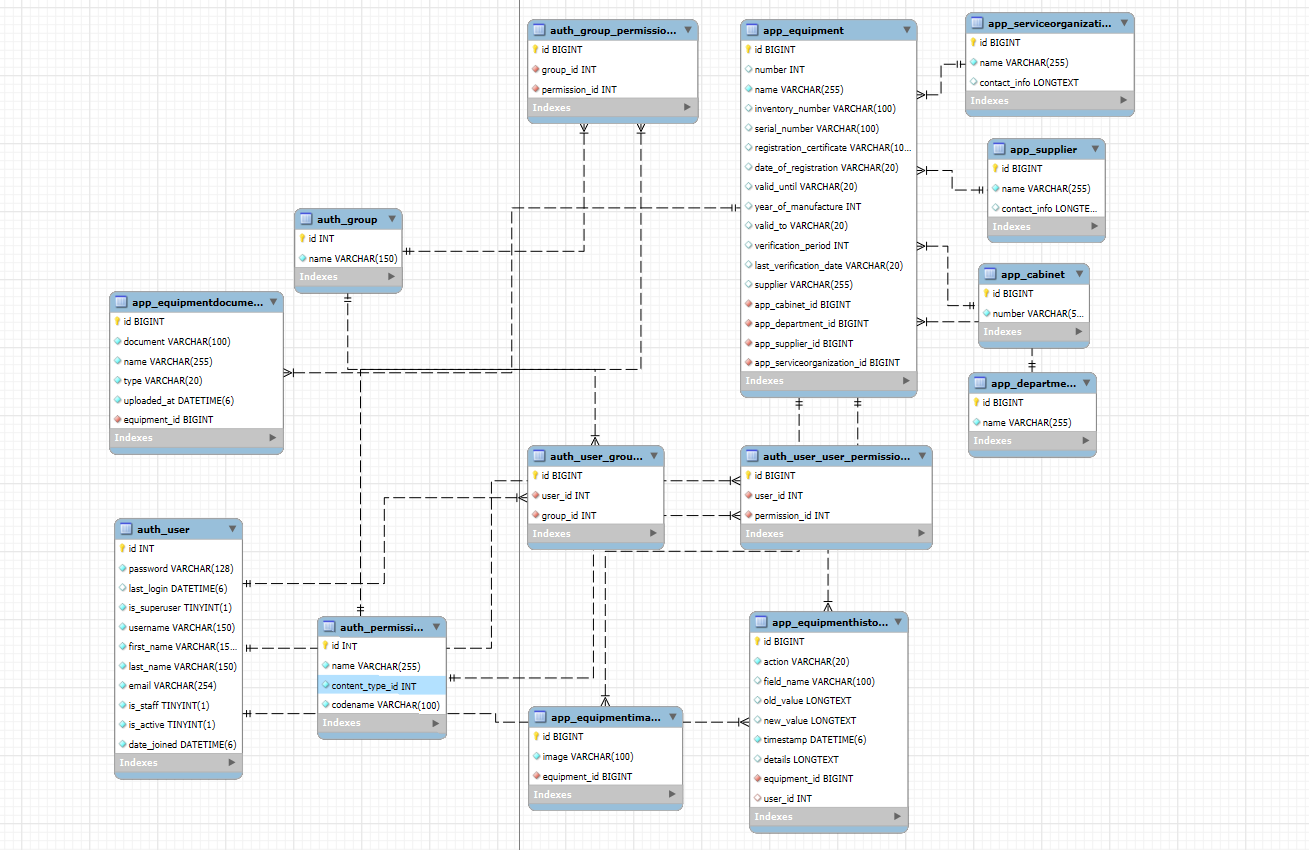
****

Рисунок 5 – ER-модель базы данных

Данная ER-модель содержит 14 таблицы для полного функционирования и качественной сортировки информации. Перечень таблиц представлен в таблице 4. В таблицах с 5 по 12 представлены поля, тип данных поля и описание.

Таблица 4 – Таблицы ER-модели

|  |  |
| --- | --- |
| Таблица | Описание |
| auth\_group\_permissions | Связующая таблица, которая определяет разрешения |
| auth\_group | Таблица групп пользователей |
| app\_equipment | Таблица, содержащая информацию об оборудовании |
| auth\_user | Таблица пользователей ИС |
| auth\_user\_user\_permission | Таблица индивидуальных разрешений |
| app\_equipmentimage | Таблица, хранящая изображения оборудования |
| auth\_permission | Таблица разрешений |
| auth\_user\_groups | Связующая таблица, которая определяет, к каким группам принадлежат пользователи |
| app\_equipmentdocument | Таблица для хранения документов |
| app\_cabinet | Таблица кабинетов |
| app\_department | Таблица отделений |
| app\_supplier | Таблица поставщиков |
| app\_equipmenthistory | Таблица истории изменений оборудования |
| app\_serviceorganization | Таблица обслуживающих организаций |

Таблица 5 – Таблица Аuth\_group\_permissions

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | Уникальный идентификатор пользователя (PK) |
| group\_id | BigInt (20) | Идентификатор группы (FK) |
| permission\_id | BigInt (20) | Идентификатор разрешения (FK) |

Таблица 6 – Таблица Аuth\_group

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | Int (20) | Уникальный идентификатор мастера (PK) |
| name | Varchar (150) | Название группы |

Таблица 7 – Таблица Аpp\_equipment

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| number | BigInt (20) | Уникальный идентификатор (PK) |
| name | Varchar (255) | Название оборудования |
| inventory\_number | Varchar (255) | Инвентарный номер |
| serial\_number | Varchar (255) | Серийный номер |
| registration\_certificate | Varchar (255) | Номер регистрационного сертификата |
| date\_of\_registration | Timestamp | Дата регистрации |
| valid\_until | Date | Дата окончания действия |
| year\_of\_manufacture | Integer | Год производства |
| valid\_to | Date | Дата, до которой оборудование считается действительным |
| verification\_period | Integer | Период проверки в днях |
| last\_verification\_date | Date | Дата последней проверки |
| instruction\_pdf | VARCHAR(100) | Путь к файлу инструкции |
| registration\_certificate\_pdf | VARCHAR(100) | Путь к файлу скана РУ |
| created\_at | DATETIME(6) | Дата и время создания |
| updated\_at | DATETIME(6) | Дата и время обновления |
| app\_cabinet\_id | BIGINT | Внешний ключ на таблицу кабинетов |
| app\_department\_id | BIGINT | Внешний ключ на таблицу отделений |
| app\_supplier\_id | BIGINT | Внешний ключ на таблицу поставщиков |
| app\_serviceorganization\_id | BIGINT | Внешний ключ на таблицу обслуживающих организаций |

Таблица 8 – Таблица auth\_user

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | Уникальный идентификатор услуги (PK) |
| password | Varchar (255) | Пароль пользователя |
| last\_login | Timestamp | Дата и время последнего входа |
| is\_superuser | Boolean | Признак суперпользователя |
| username | Varchar (255) | Имя пользователя |
| first\_name | Varchar (255) | Имя |
| email | Varchar (255) | Электронная почта |
| is\_staff | Boolean | Админ состав |
| is\_active | Boolean | Активность пользователя |
| date\_joined | Timestamp | Дата регистрации пользователя |

Таблица 9 – Таблица auth\_permission

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | Уникальный идентификатор услуги (PK) |
| name | Varchar (255) | Название разрешения |
| content\_type\_id | BigInt (20) | Идентификатор типа контента |
| codename | Varchar (255) | Кодовое имя разрешения |

Таблица 10 – Таблица аpp\_equipmentimage

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | Уникальный идентификатор услуги (PK) |
| image | Varchar (255) | Путь к изображению |
| equipment\_id | BigInt (20) | Идентификатор оборудования (FK) |

Таблица 11– Таблица app\_equipmentdocument

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | Уникальный идентификатор услуги (PK) |
| document | VARCHAR(100) | Путь к файлу документа |
| name | VARCHAR(255) | Название документа |
| uploaded\_at | DATETIME(6) | Дата и время загрузки документа |
| equipment\_id | BIGINT | Внешний ключ, связывающий документ с оборудованием |
| type | VARCHAR(20) | Тип документа |

Таблица 12 – Таблица auth\_user\_groups

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | Уникальный идентификатор услуги (PK) |
| user\_id | BigInt (20) | Идентификатор пользователя (FK) |
| group\_id | BigInt (20) | Идентификатор группы (FK) |

Таблица 13 – Таблица Auth\_user\_user\_permissions

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | Уникальный идентификатор услуги (PK) |
| user\_id | BigInt (20) | Идентификатор пользователя (FK) |
| permission\_id | BigInt (20) | Идентификатор разрешения (FK) |

Таблица 14 – Таблица app\_serviceorganization

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | ID обслуживающей организации (PK) |
| name | VARCHAR(255) | Название обслуживающей организации |
| contact\_info | VARCHAR(255) | Контактная информация |

Таблица 15 – Таблица app\_supplier

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | ID поставщика (PK) |
| name | VARCHAR(255) | Название поставщика |
| contact\_info | VARCHAR(255) | Контактная информация |

Таблица 16 – Таблица app\_cabinet

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | ID кабинета (PK) |
| number | VARCHAR(55) | Номер кабинета |

Таблица 17 – Таблица app\_department

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | ID отделения (PK) |
| name | VARCHAR(155) | Название отделения |

Таблица 18 – Таблица app\_equipmenthistory

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | BigInt (20) | ID истории (PK) |
| action | VARCHAR(20) | Действие |
| field\_name | VARCHAR(100) | Название изменённого поля |
| old\_value | LONGTEXT | Старое значение |
| new\_value | LONGTEXT | Новое значение |
| timestamp | DATETIME(6) | Дата и время изменения |
| details | LONGTEXT | Дополнительные детали |
| equipment\_id | BIGINT | id оборудования |
| user\_id | INT | id пользователя |

4.3 Проектирование интерфейса

Для разработки прототипа пользовательского интерфейса был выбран инструмент draw.io – это инструмент для создания блок-схем, диаграмм и прочего.

Были спроектированы три прототипа будущих страниц информационной системы: окно авторизации (рисунок 6), окно главной страницы (рисунок 7), а также страница карточки оборудования (рисунок 8). Создание прототипов будущих страниц помогает и ускоряет процесс разработки программного продукта.

На рисунке 6 представлен прототип окна авторизации.

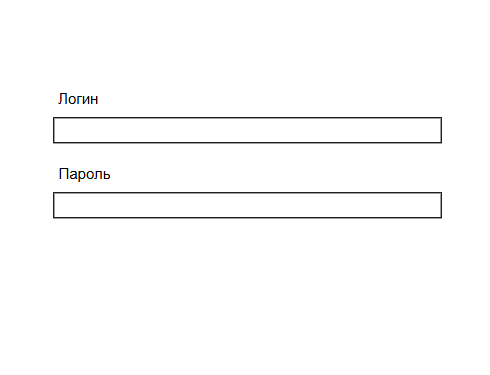


Рисунок 6 – Прототип окна авторизации

На рисунке 7 представлен прототип окна авторизации.

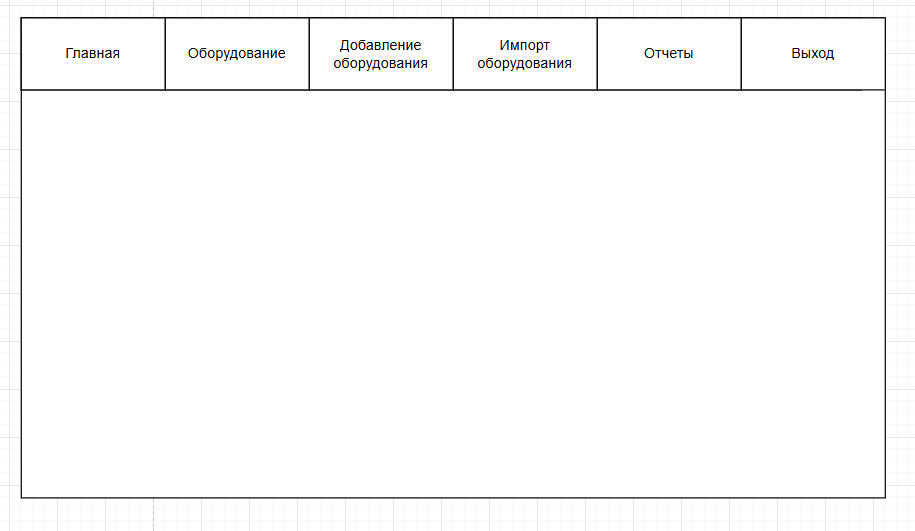
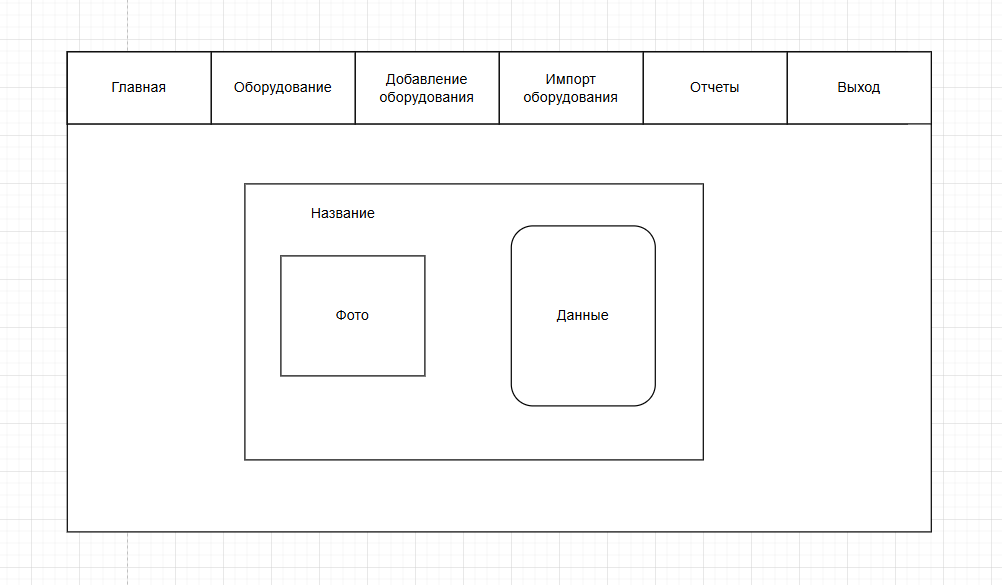


Рисунок 7 – Прототип окна главной страницы

На рисунке 8 представлен прототип страницы главная



На рисунке 8 представлен прототип карточки оборудования

**5 Разработка ИС**

**5.1 Разработка интерфейса ИС**

В ИС «Медицинская система учета» реализуется следующий функционал:

* Авторизация, регистрация пользователей в ИС.
* Просмотр карточки оборудования.
* Просмотр истории оборудования.
* Добавление и удаление оборудования из ИС.
* Редактирование оборудования.
* Осуществление поиска и фильтрации.
* Формирование отчётов.
* Интеграция со сторонними ИС.
* Импорт оборудования из excel.
* Загрузка и управление документаций оборудования.

Программный продукт прост в освоении из-за интуитивно понятному интерфейсу. Для разработки интерфейса информационной системы использовался такой фреймворк, как Django. Разработка удобного пользовательского интерфейса – это один из важнейших этапов в процессе создания информационной системы.

Все поля ввода и формы в информационной системе обладают проверкой вводимых данных, таким образом в случае некорректности введенных данных пользователь не получит сообщение об ошибке, а будет уведомлен о возникшем несоответствии.

В разработке информационной системы будет использоваться формат -.html. Файл с таким форматом содержат в себе HTML – разметку с встроенными шаблонными тегами и фильтрами Django.

Разработка информационной системы началась с создания основного шаблона, код которого используется для создания всех следующих страниц (рисунки с 9 по 10). В состав входит верхняя навигационная панель, основные стили для элементов на странице.



Рисунок 9 – Навигационная панель (часть 1)

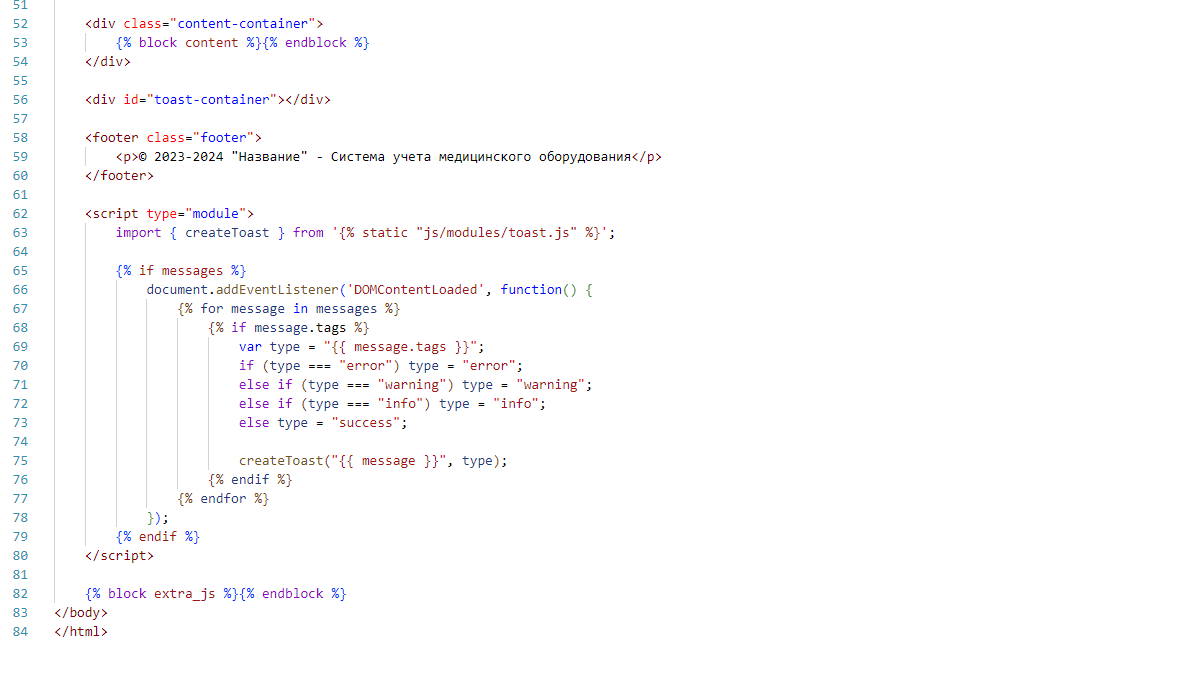


Рисунок 10 – Навигационная панель (часть 2)

Для отображения навигационной панели используется файл base.html который обеспечивает общую навигационную всем страницам.

На рисунке 11–13 представлена главная страница информационной системы «Медицинская система учета» home.html.



Рисунок 11 – Главная страница (часть 1)

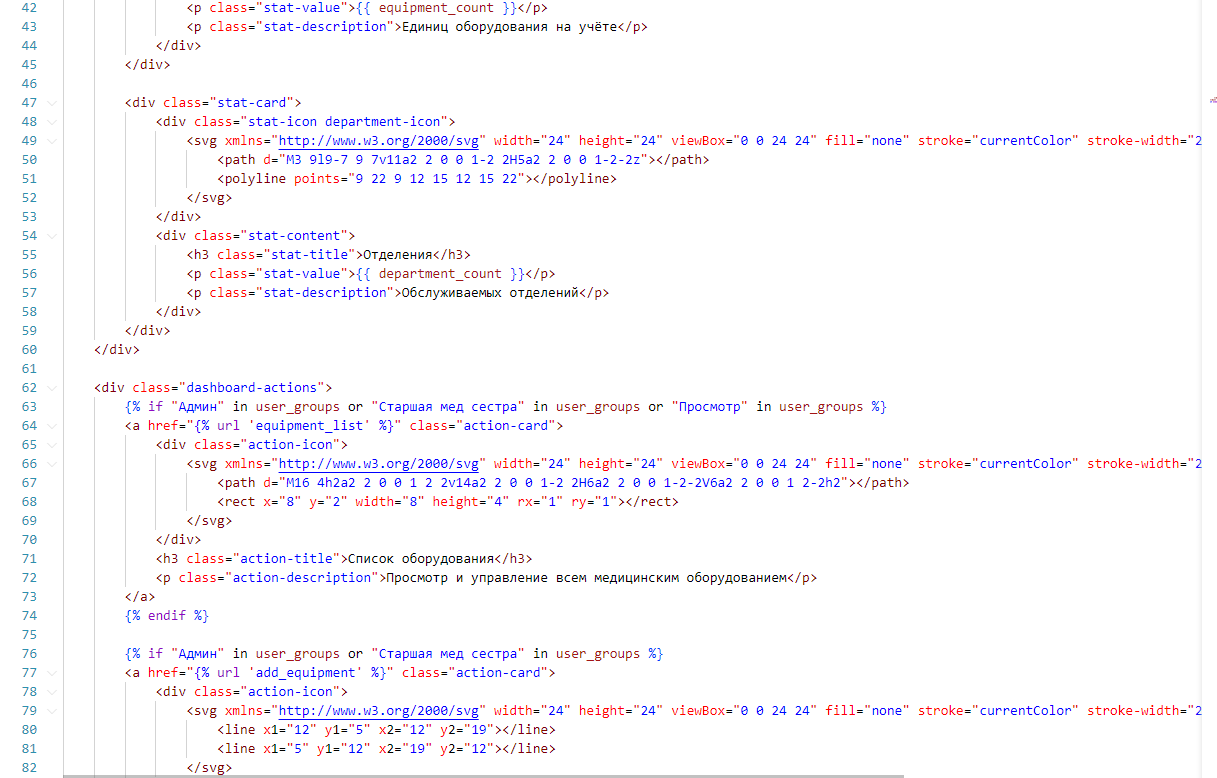


Рисунок 12 – Главная страница (часть 2)

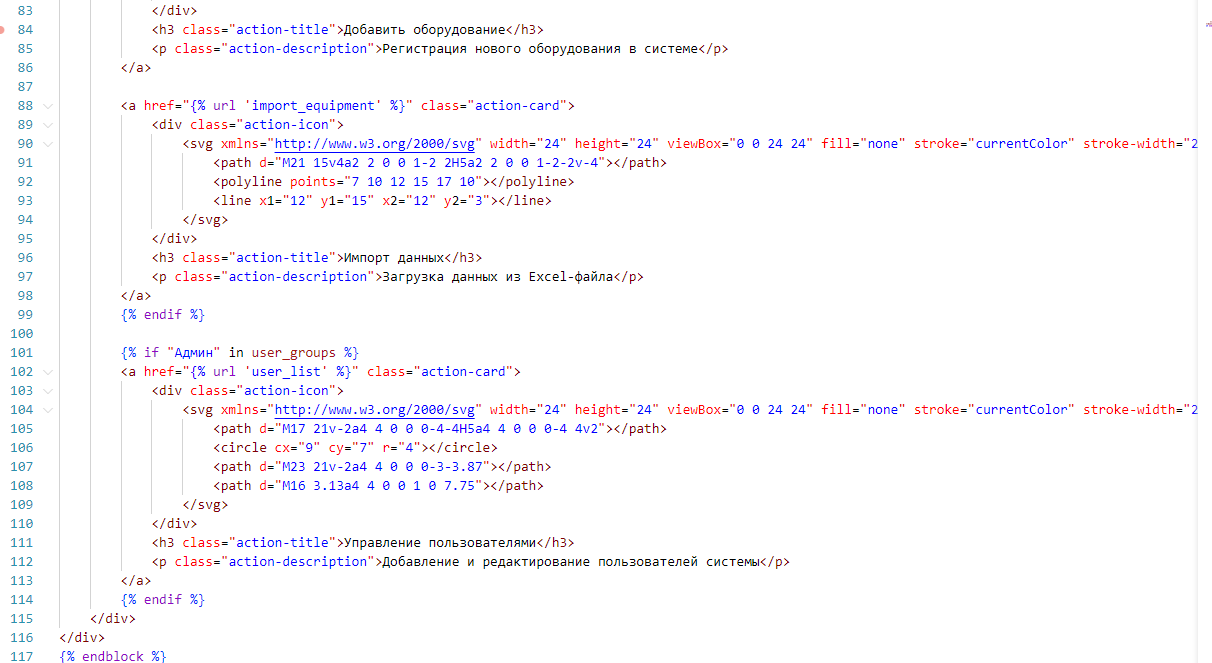


Рисунок 13 – Главная страница (часть 3)

**5.2 Разработка базы данных ИС**

В качестве СУБД для реализации ИС «Медицинская система учета» была выбрана MySQL, исходя из описания в разделе «Анализ инструментов, используемых в разработки программного продукта».

База данных содержит 9 таблиц (ER-модель на рисунке 5). Среди этих таблиц располагаются: таблица с пользователями, таблица разрешений, таблица групп пользователей, таблица информации об оборудовании, таблица изображений, таблица журнала администрирования, таблица определения групп пользователей, таблица индивидуальных разрешений.

Разработка базы данных началась с создания таблицы с оборудованием. Таблица с данными представлена на рисунке 17.

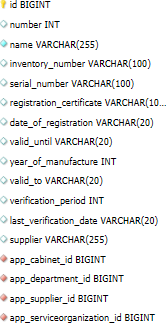


Рисунок 17 – Таблица оборудования

Далее была создана таблица с изображениями оборудования, которая связана с таблицей оборудования через внешний ключ. Таблица представлена на рисунке 18.



Рисунок 18 – Таблица изображений

После была создана таблица с документами оборудования, которая также связана с таблицей оборудования. Представлена на рисунке 19.

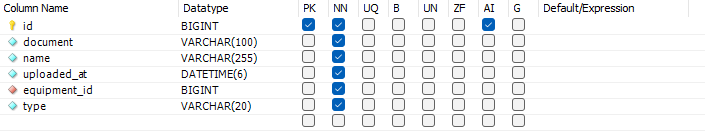


Рисунок 19 – Таблица документов оборудований

Затем была создана таблица с пользователями. Таблица представлена на рисунке 20.

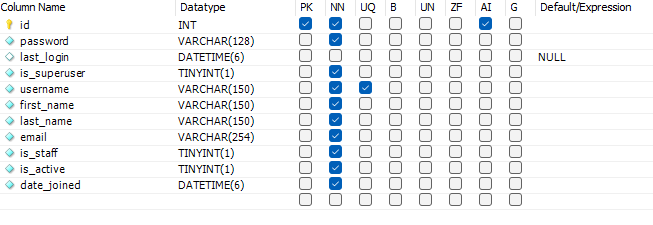


Рисунок 20 – Таблица пользователей

**5.3 Разработка серверной части**

Серверная часть проекта построенная на Django, обеспечивает стабильную работу информационной системе. Основная функциональность включает работу с оборудованием, отделами, кабинетами, сервисными организациями и поставщиками через административную панель.

На рисунке 21 представлен отрывок кода страницы управления медицинским оборудованием в административной панели.

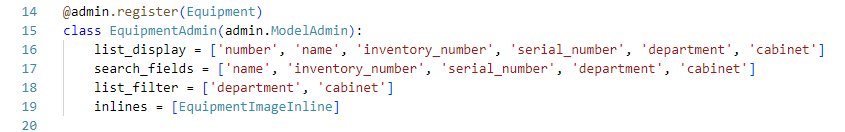
****

Рисунок 21 – Управления медициским оборудованием

Данный код обеспечивает удобный интерфейс для администратора по управлению медицинским оборудованием, что способствует эффективному ведению учета и поддержанию актуальности данных в системе.

На рисунке 22 и 23 изображены отрывки кода управления отделами и кабинетами в информационной системе.

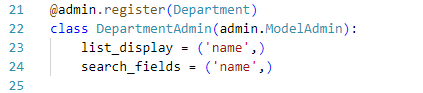


Рисунок 22 – Код страницы отчета

На данном отрывке кода реализована форма управления отделами в административной панели:

* Отображение данных

В таблице отображается название отдела

* Поиск

Реализован поиск по названию отдела

* Добавление / редактирование
* Возможность добавления новых отделов и редактирования существующих

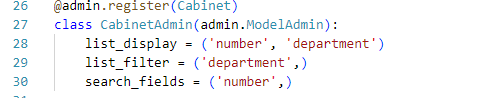


Рисунок 23 – Код страницы отчета

На рисунке 23 отображается вывод таблицы кабинетов в административной панели:

Отображение данных:

Для каждого кабинета отображается его номер и принадлежность к отделу

* Фильтрация:

Возможность фильтрации кабинетов по отделам

* Поиск:

Реализован поиск по номеру кабинета

Данный код страницы подробно показан в приложении Б.

5.4 Тестирование информационной системы

Во время разработки необходимо провести тестирование для проверки работоспособности информационной системы «Медицинская система учета». Тестирование – процесс проверки информационной системы на основе набора тестов, путем сопоставления полученных результатов выполнения программы с прогнозируемыми выходными данными набора тестов. Основной целью тестирования является обнаружение дефектов в информационной системе для дальнейшего проведения отладки. Для тестирования используется:

* Модульное тестирование (Unit Testing):

Проводится тестирование отдельных компонентов системы в изоляции. Отладка – процесс поиска и устранения ошибок в коде или в функциональности оборудования. Для тестирования был создан сценарий тестирования регистрации пользователя для информационной системы «Сувенирные мечи».

Таблица 14 – Сценарий 1: Тестирование регистрации

|  |  |
| --- | --- |
| Поле | Описание |
| Дата теста | 18.05.2025 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Успешная регистрация нового пользователя |
| Этапы теста | Система обрабатывает запрос на регистрацию с корректными данными пользователя. |
| Тестовые данные | username: 'test\_reg\_user' 'first\_name': 'Test Reg Nik','email': 'test\_register@example.com','password': 'password123', 'password2': 'password123' |
| Ожидаемый результат | Система возвращает индикатор успешной регистрации. |
| Фактический результат | Пользователь успешно зарегистрировался |

В таблице 14 представлен тестовый набор сценария регистрации пользователя в информационной системе «Медицинская система учета».

Таблица15 – Сценарий 2: Тестирование авторизации

|  |  |
| --- | --- |
| Поле | Описание |
| Дата теста | 18.05.2025 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Успешная авторизация существующего пользователя |
| Этапы теста | Система обрабатывает запрос на авторизацию с корректными email и паролем. |
| Тестовые данные | username: 'test\_auth\_user', password: 'password123', |
| Ожидаемый результат | Система возвращает индикатор успешной авторизации и данные пользователя (например, user\_id). |
| Фактический результат | Пользователь успешно авторизовался |

В таблице 15 представлен тестовый набор сценария авторизации пользователя в информационной системе



Рисунок 24 – Код проверки регистрации Django TestCase (часть 1)

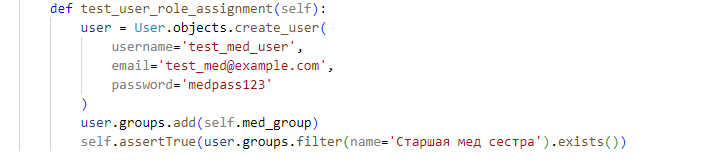


Рисунок 25 – Код проверки регистрации Django TestCase (часть 2)

Для того чтобы начать проверку, необходимо открыть терминал и ввести команду python manage.py test app.tests.RegistrationTest. На рисунке 26 изображён результат проверки кода регистрации.

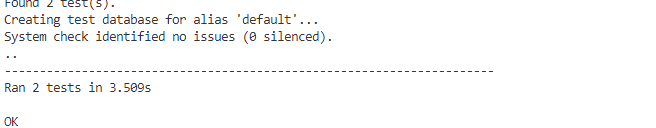


Рисунок 26 – Результат проверки регистрации Django TestCase

В результате проверки кода регистрации, проверка прошла успешно и у условного пользователя удалось пройти регистрацию.

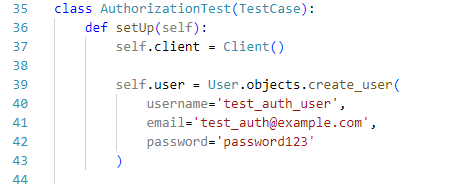


Рисунок 27 – Код проверки авторизации Django TestCase(часть 1)

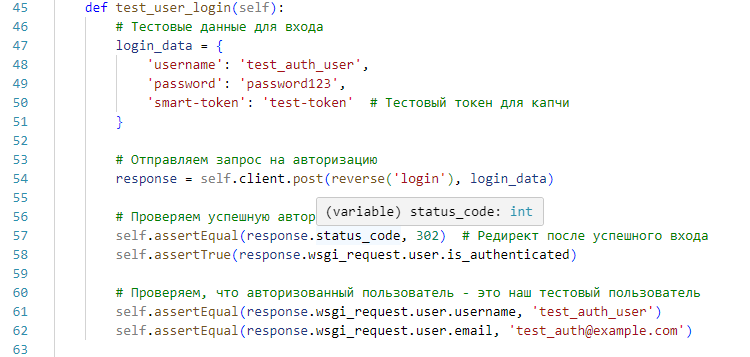


Рисунок 28 – Код проверки авторизации TestCase (часть 2)

Для того чтобы начать проверку, необходимо открыть терминал и ввести команду python manage.py test app.tests.AuthorizationTest. На рисунке 29 изображён результат проверки кода регистрации.

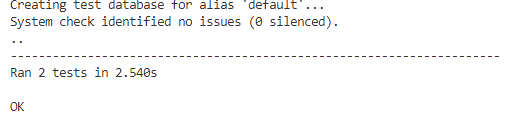


Рисунок 29 – Результат проверки авторизации TestCase

В результате проверки кода авторизации, проверка прошла успешно и у условного пользователя удалось войти в систему под своими данными.

В ходе тестирования были проверены основные пользовательские сценарии взаимодействия с регистрацией и авторизацией:

* Функция регистрации успешно отрабатывает – Система имитирует успешное создание нового пользователя.
* Функция авторизации успешно отрабатывает – Система имитирует успешный вход пользователя с использованием его данных.

Все тесты прошли успешно (используя мокированные сервисы), что подтверждает:

* корректное взаимодействие тестового кода с логикой сервисов (в имитированной среде),
* следование заданным правилам и сценариям работы функций регистрации и авторизации,
* соответствие результатов имитируемых операций ожидаемым в тестовых сценариях.

6 Документирование информационной системы

6.1 Руководство наблюдателя информационной системы

При входе информационную систему «Медицинская система учета», наблюдатель сразу попадает на страницу авторизации, представленную на рисунке 30.

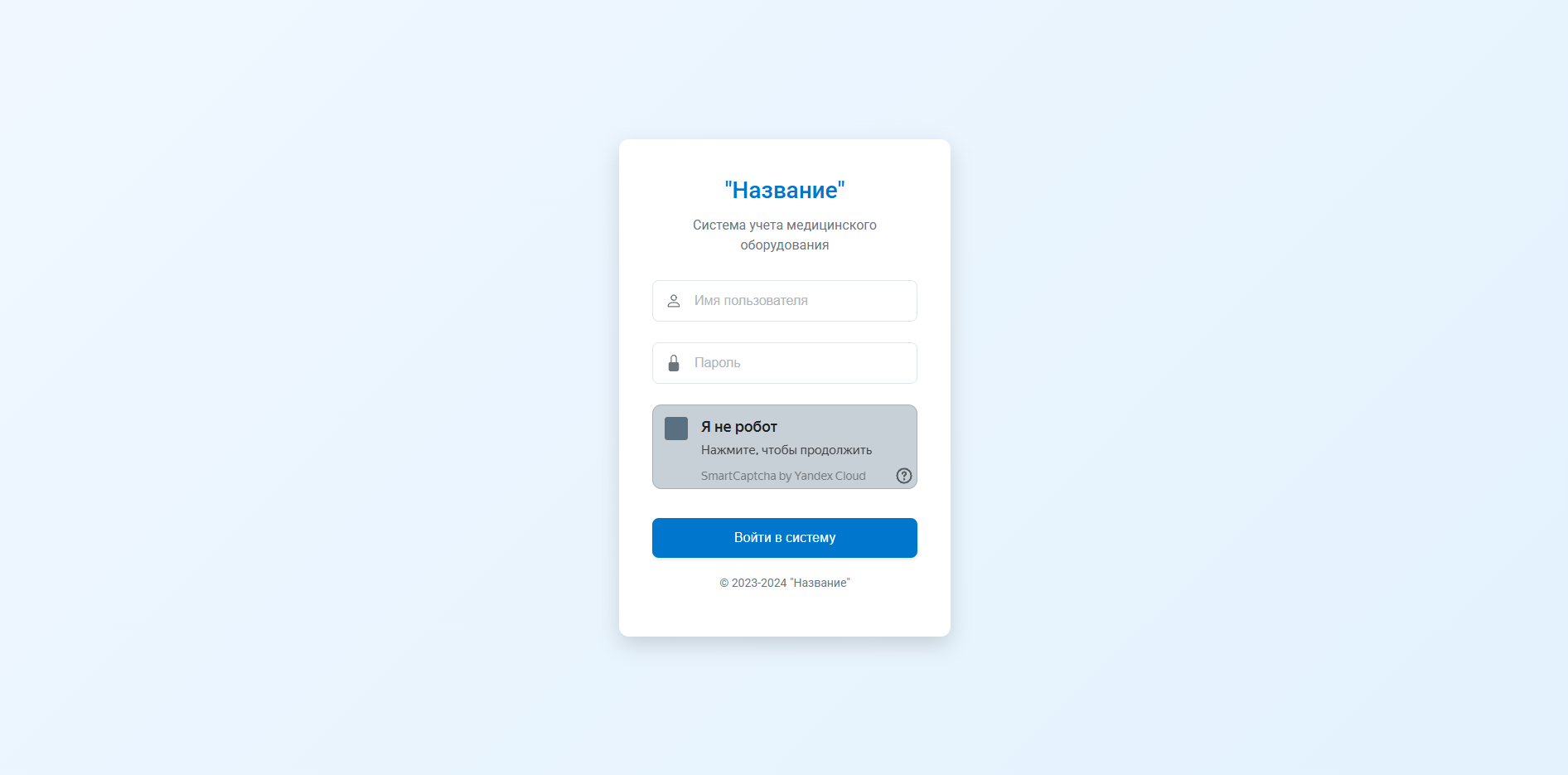


Рисунок 30 – Страница авторизации

На данной странице происходит авторизация уже существующих пользователей информационной системы.

После успешной авторизации наблюдатель переход на главную страницу информационной системы, представленную на рисунке 31. Где он может сразу увидеть сколько единиц оборудования находиться в данный момент на учете в организации. А так же сколько обслуживаемых отделений в данный момент находится в организации. Исходя из главной страницы наблюдатель уже может понять сколько оборудования обслуживается.

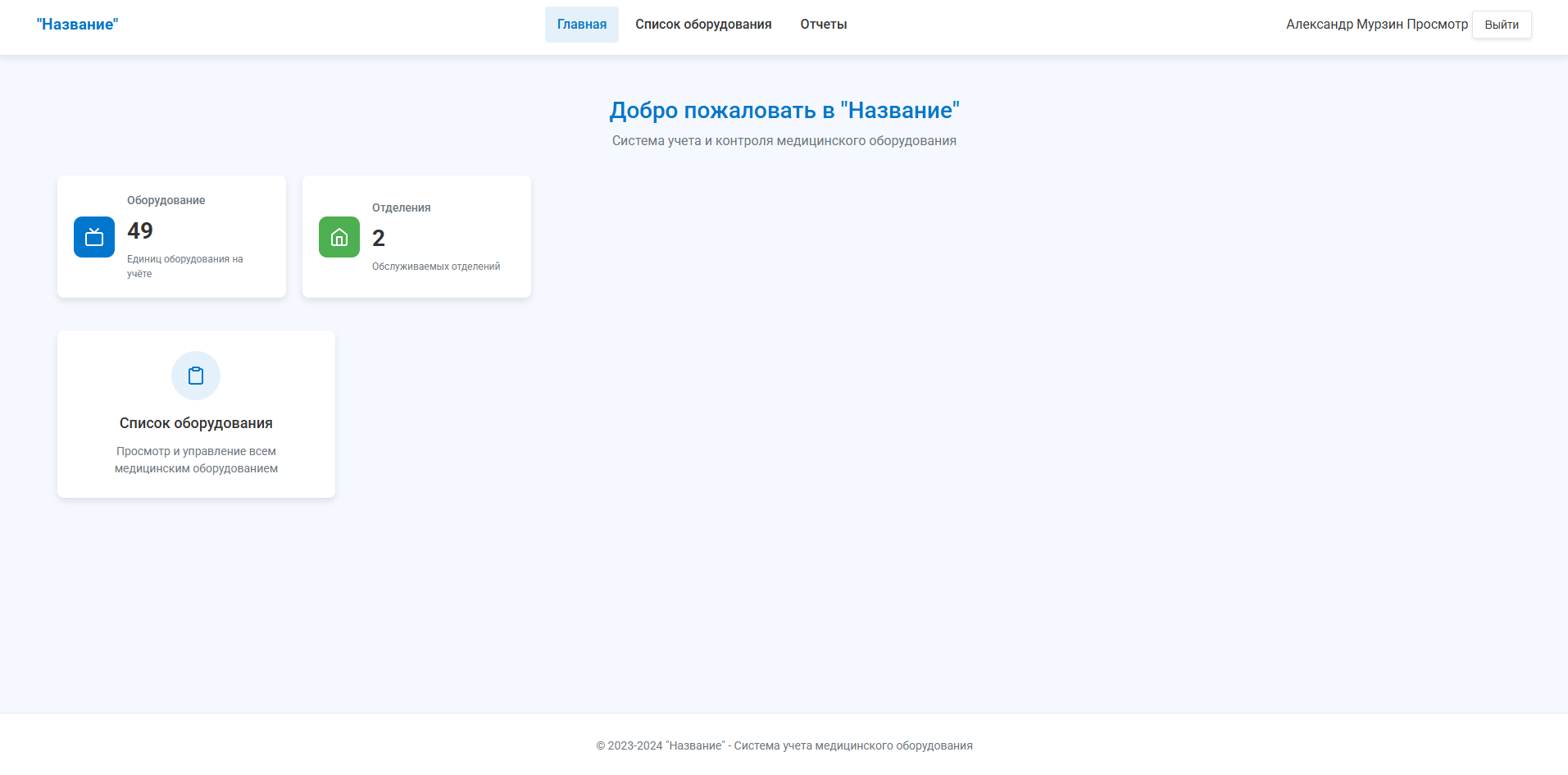


Рисунок 31 – Главная страница

После главной страницы наблюдатель может перейти на страницу списка оборудований, представленную на рисунке 32

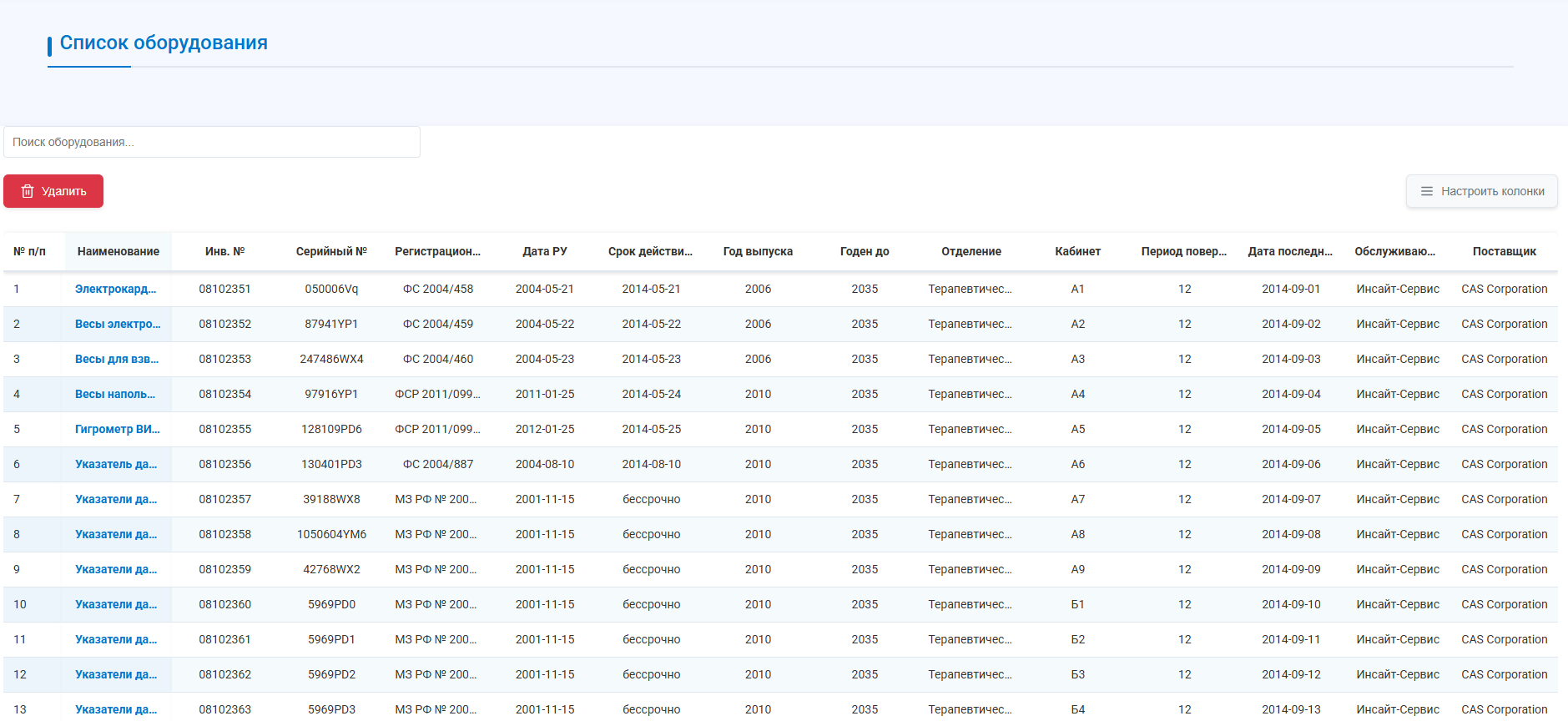


Рисунок 32 – Страница списка оборудования

На данной странице наблюдатель может посмортеть почти все данные об оборудованиях, а так же изменить показ колокон, нажав на настройку колонок которая представленна на рисунке 33.

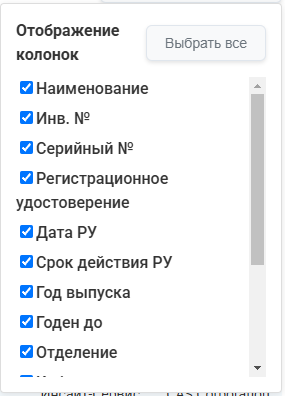


Рисунок 33 – Отображение колонок

После чего наблюдатель может найти нужное ему оборудование и открыть карточку с ним что бы увидеть более детальную информацию об оборудования, а так же увидеть его фотографию и дополнительный PDF документы к ниму (рисунок 34).

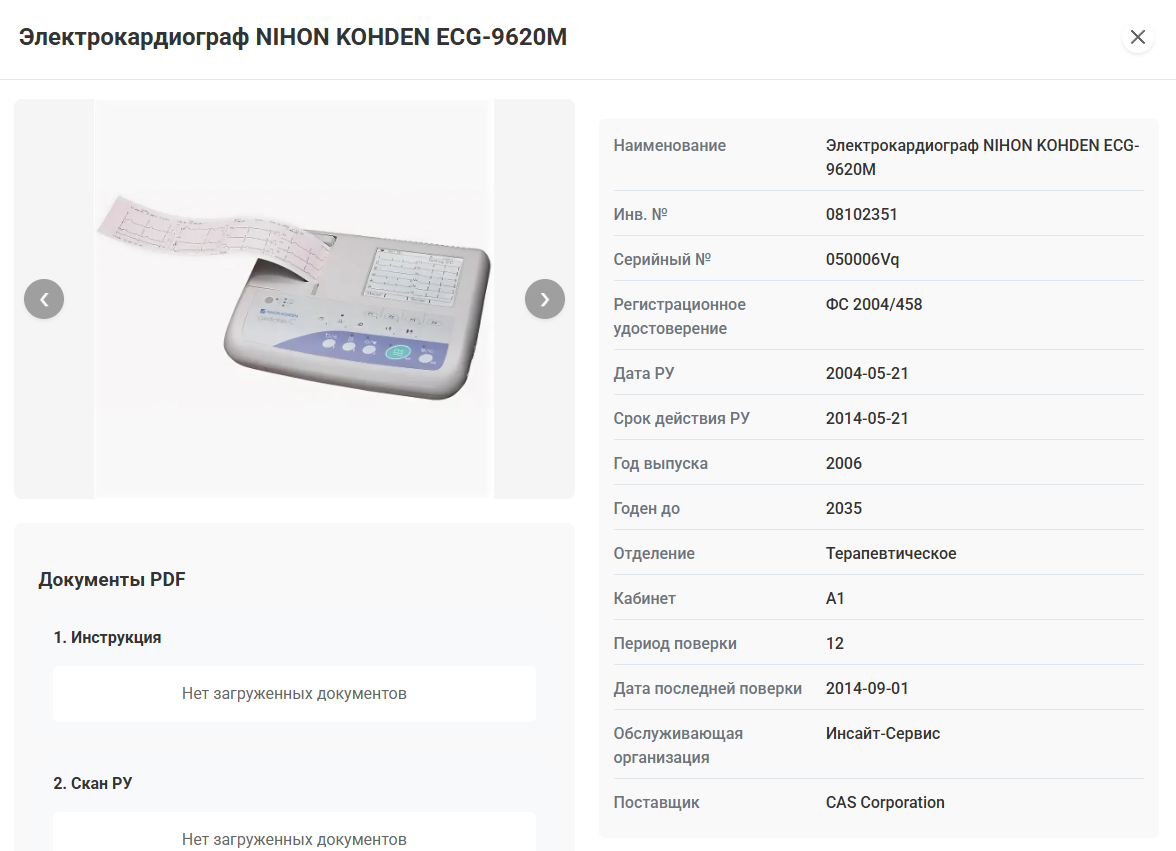


Рисунок 34 – Страница с подробной информацией об оборудовании

7 Стоимость разработки и внедрения программного продукта

Разработка любого программного продукта (далее – ПП) должна быть экономически обоснованной. При этом обоснование целесообразности разработки ПП должно учитывать не только затраты на саму разработку и сопутствующие статьи расходов. При создании программного продукта очень важно оценить его стоимость. Время, затраченное на разработку отдельных компонентов, отражает трудоёмкость и длительность разработки программного обеспечения.

7.1 Расчёт трудоёмкости разработки программного обеспечения

Затраты времени могут включать:

затраты труда на анализ предметной области проекта – . ;

затраты труда на анализ инструментов – ;

затраты труда на проектирование проекта и описание этапов проектировании – ;

затраты труда на создание технического задания на разработку проекта – ;

затраты труда на разработку программного продукта – ;

затраты труда на тестирование программного продукта – ;

затраты труда на подготовку и оформление документов – .

Суммарные затраты труда рассчитываются как сумма составных затрат труда по формуле (1):

, (1)

Расчет суммарных затрат времени представлен в таблице 20.

Таблица 16 – Суммарные затраты времени

|  |  |  |
| --- | --- | --- |
| Вид работ | Трудоемкость в часах | |
| Всего | В том числе машинное время |
| Анализ предметной области проекта и описание структуры | 20 | 15 |
| Анализ инструментов | 20 | 20 |

Продолжение таблицы 16

|  |  |  |
| --- | --- | --- |
| Проектирование проекта и описание этапов проектирования | 40 | 40 |
| Создание технического задания на разработку проекта | 20 | 15 |
| Разработка программного продукта | 100 | 100 |
| Тестирование программного продукта | 40 | 40 |
| Подготовка документации | 20 | 10 |
| Итого: | 260 | 240 |

7.2 Расчёт затрат на разработку программного обеспечения

Затраты на разработку программного обеспечения включают в себя:

* затраты на зарплату разработчика ();
* затраты на амортизацию оборудования ();
* затраты на эксплуатацию оборудования (электроэнергия) ();
* затраты на материалы, из расходованные при проведении разработки (бумага, картридж) ();
* прочие затраты (использование платного ПО).

Затраты на разработку рассчитываются путем суммы всех затрат по формуле 2.

, (2)

7.2.1 Затраты на оплату труда

Затраты на оплату (ЗОТ) труда разработчика ПО включают затраты на оплату труда и отчисления от фонда заработной платы.

Затраты на заработную плату определяются произведением часовой тарифной ставки разработчика и трудоемкости разработки программного продукта по следующей формуле (3).

, (3)

где ∑t – суммарные затраты труда, вычисляемые по формуле (1), час.;

– часовая тарифная ставка, руб.

В расчете зарплаты учитываем модальную заработную плату по Иркутску – 80000 руб. Ставка за час – 484 руб. (при условии среднего количества рабочих часов при 40–часовой недели –165 ч.).

,

Далее необходимо рассчитать затраты на оплату труда.

Итоговый расчет заработной платы для разработчика ПО рассчитывается по формуле (4).

, (4)

Далее необходимо рассчитать заработную плату с учетом налога (ЕСН=13%).

7.2.2 Затраты на амортизацию оборудования

Амортизация – это процесс периодического переноса начальной стоимости основного средства или нематериального актива на производственные, коммерческие или общехозяйственные расходы – в зависимости от того, как этот актив используется.

Расчет амортизации производится при помощи данных формул (5–8).

, (5)

где: – годовая норма амортизации;

– срок полезного использования в соответствии с классификатором.

, (6)

где: – ежегодная сумма амортизации;

– начальная стоимость оборудования.

, (7)

где: – ежемесячная сумма амортизации.

Далее необходимо сумму ежемесячной амортизации умножить на количество месяцев эксплуатации оборудования.

, (8)

При стоимости оборудования 70 000 руб. и сроке службы 3 года.

Поскольку общее время эксплуатации оборудования в рамках проекта составляет 2 месяца. (240/165 = 1,5мес)

7.2.3 Расчёт затрат на электроэнергию

Для расчета затрат на электроэнергию первоочередное необходимо рассчитать расход электроэнергии оборудования по следующей формуле (9).

, (9)

где: P – электрическая мощность в киловаттах;

t – время в часах.

Расчет стоимости электроэнергии по следующей формуле (10):

, (10)

где Ц – стоимость киловатт–час.

Мощность ноутбука: 0,05 кВт. Согласно нормам, рабочий день длится 8 часов. Предполагаемое время выполнения работы – 30 рабочих дня, что составляет 240 рабочих часов.

При тарифе 1.58 руб./кВт⋅ч.

7.2.4 Затраты на материалы

Затраты на материалы складывается из суммы стоимости материалов, которые сводятся в таблице 17.

Таблица 17 – Затраты на материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу (руб.) | Кол–во (шт.) | Всего (руб.) |
| Бумага (500 листов) | 460 | 1 | 460 |
| Переплет диплома | 500 | 1 | 500 |
| **Итого** |  |  | **960** |

Общие расходы представлены в таблице 18.

Таблица 18 – Общие расходы

|  |  |  |  |
| --- | --- | --- | --- |
| Статьи затрат | Индекс | Сумма, руб. | Удельный вес затрат, % |
| Заработная плата |  | 109 480,8 | 91 |
| Амортизация оборудования |  | 3 884 | 5 |
| Затраты на электроэнергию |  | 18,96 | 0,5 |
| Затраты на материалы |  | 960 | 3,5 |
| Итого: |  | 114 343,76 | 100 |

7.2.5 Расчёт цены

Устанавливая цену, нужно исходить из необходимости компенсации 11 затрат на ее производство, уплаты налогов и получение прибыли для дальнейшего развития предприятия. Состав расчетной цены на разработку определяется по формуле (11).

, (11)

где – затраты на разработку;

– прибыль от реализации.

При планируемой прибыли 12%.

, (11)

где – затраты на разработку;

– прибыль от реализации.

При планируемой прибыли 12%.

При НДС 20%.

Итоговая цена.

После проведения всех расчетов, итоговой стоимостью разработки и проектирования программного продукта будет –153 678,01 рубля.

Заключение

В ходе выполнения дипломного проекта была успешно разработана информационная система «Медицинская система учета». Система предназначена для организации работы медицинского учреждения по учету и управлению медицинским оборудованием, предоставляя функционал как для медицинского персонала, так и для административного состава. Были реализованы основные сущности базы данных, соответствующие бизнес-логике проекта, включая информацию о пользователях, медицинском оборудовании, документации, а также необходимых связующих таблиц для обеспечения целостности данных. Между сущностями установлены необходимые связи.В рамках проекта был разработан и реализован следующий функционал:

* Авторизация пользователей в системе с использованием капчи для безопасности
* Управление пользователями с различными уровнями доступа
* Управление медицинским оборудованием
* Загрузка и управление изображениями оборудования
* Загрузка и управление документацией по оборудованию
* Импорт и экспорт данных об оборудовании в форматах Excel и PDF
* Генерация отчетов по оборудованию с возможностью фильтрации
* Поиск и фильтрация оборудования по различным параметрам

Функциональное тестирование ключевых пользовательских сценариев, таких как авторизация пользователя, управление оборудованием и генерация отчетов, подтвердило корректность работы реализованной бизнес-логики и готовность системы к эксплуатации. В процессе выбора инструментальных средств для разработки были проанализированы различные технологии и языки программирования. На основе проведенного анализа для реализации информационной системы был выбран стек технологий, включающий:

* Python и Django в качестве основного фреймворка
* JavaScript для интерактивных элементов интерфейса
* HTML и CSS для создания пользовательского интерфейса
* SQLite для хранения данных
* ReportLab для генерации PDF-отчетов
* OpenPyXL для работы с Excel-файлами

Разработанный программный продукт обладает интуитивно понятным интерфейсом и реализует полный набор функций, необходимый для эффективного управления медицинским оборудованием. Основной технической сложностью в ходе разработки стало обеспечение безопасности системы и эффективное управление файлами (изображениями и документами). Эта проблема была успешно решена путем реализации системы прав доступа и оптимизации работы с файлами. Все поставленные цели и задачи дипломного проекта по созданию информационной системы «Медицинская система учета» были успешно выполнены, что подтверждает достижение требуемого уровня функциональности и готовности системы к внедрению в медицинское учреждение.

Список использованных источников

Приложение А – Техническое задание

**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

**Техническое задание**

**ИНФОРМАЦИОННАЯ СИСТЕМА «МЕДИЦИНСКАЯ СИСТЕМА УЧЕТА»**

|  |  |  |
| --- | --- | --- |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | (А.И. Тирский) |
|  | (подпись, дата) |  |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | (А.С. Мурзин) |
|  | (подпись, дата) |  |

Иркутск 2025

**1 Введение**

**1.1** **Общие сведения**

Документ представляет собой техническое задание на создание информационной системы «Медицинская система учета оборудования», предназначенной для автоматизации и управления процессами учета медицинского оборудования в медицинских учреждениях. Система должна обеспечивать комплексное решение для учета оборудования, управления обслуживанием.

**1.2** **Цели и задачи**

Цель создания информационной системы «Медицинская система учета» автоматизация учета оборудования и улучшение обслуживания. Задачи системы включают:

* Учет и управление оборудованием.
* Автоматизация контроля обслуживания.
* Ведение документации.
* Управление доступом и безопасность.
* Формирование отчетов.
* Улучшение обслуживания.
* Много доступность.

**2 Основания для разработки**

**2.1** **Нормативные документы**

Документ основывается на следующих нормативных документах:

* ГОСТ 34.602-2020 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы».
* ГОСТ Р 56477-2015 «Проектирование и внедрение информационных систем. Общие требования».

**2.2** **Проектные документы**

Проектные документы включают:

* Пояснительную записку.
* Руководство пользователя.

**3 Назначение системы**

**3.1 Общее описание**

Назначение информационной системы «Медицинская система учета оборудования» заключается в создании единой платформы для автоматизации и оптимизации процессов учета медицинского оборудования. Основная задача системы – предоставить медицинскому персоналу удобный и быстрый доступ к информации об оборудовании, упростить процесс учета и обслуживания, а также улучшить контроль за состоянием медицинской техники.

**3.2** **Преимущества и новизна**

Информационная система будет предоставлять:

* Автоматизация процессов – сокращение времени на учет оборудования, формирование отчетов и контроль сроков поверки за счет интеграции всех процессов в единую систему.
* Удобство для персонала – возможность быстрого поиска оборудования, отслеживания его состояния и местоположения.
* Аналитика и контроль – встроенные инструменты для анализа состояния оборудования, прогнозирования сроков поверки и оптимизации обслуживания

**4 Требования к системе**

**4.1** **Функциональные требования**

Раздел «Администратор»

* Добавление, редактирование и удаление учетных записей.
* Назначение ролей.
* Добавление, редактирование и удаление оборудования.
* Импортирование данных об оборудовании.
* Формирование отчетов об оборудовании.
* Добавление и удаление PDF файлов для оборудования.
* Просмотр истории оборудования.

Раздел «Старшая медицинская сестра»

* Добавление и редактирование оборудования
* Импортирование данных об оборудовании.
* Формирование отчетов об оборудовании.
* Добавление и удаление PDF файлов для оборудования.

Раздел «Медицинская сестра»

* Добавление и редактирования оборудования
* Формирование отчетов об оборудовании
* Просмотр таблицы оборудования.
* Просмотр карточки оборудования.

Раздел «Пользователь»

* Формирование отчетов об оборудовании.
* Просмотр таблицы оборудования.
* Просмотр карточки оборудования.

**4.2** **Технические требования**

1. Производительность:
   1. Обработка до 1000 задач одновременно.
   2. Время отклика системы не более 15 секунд при загрузке данных.
2. Надежность:
   1. Доступность системы не менее 99,5% в год.
   2. Регулярное резервное копирование данных.
3. Безопасность:
   1. Аутентификация пользователей.

**4.3. Эксплуатационные требования**

1. Удобство использования:
   1. Дружественный и интуитивный интерфейс как для сотрудников, так и для администраторов.

**5 Требования к техническому обеспечению**

**5.1 Оборудование**

* Сервер: Серверная платформа с процессором не менее 4 ядер, 8 ГБ ОЗУ, SSD объемом 500 МБ.
* Клиентские рабочие станции: ПК с ОС Windows 7, 4 ГБ ОЗУ, 500 МБ свободного места на диске.

**5.2 Сетевые требования**

* Сеть: Доступ в Интернет со скоростью не менее 10 Мбит/с.
* Сетевые протоколы: Поддержка TCP/IP, HTTPS.

**6 Требования к программному обеспечению**

**6.1 Программные компоненты**

* Операционная система: Серверная версия Windows Server.
* Базы данных: MySQL.
* Программное обеспечение: Веб–сервер Nginx, интерпретатор PHP.

**6.2 Интерфейсы**

* Интерфейс пользователя: Веб-интерфейс с поддержкой браузеров Chrome, Firefox, Edge.

**7 Организационно–технические требования**

**7.1 Этапы разработки**

В таблице 1 представлены сроки и этапы разработки информационной системы.

Таблица 1 – Сроки и этапы разработки дипломного проекта

|  |  |  |
| --- | --- | --- |
| № | Наименование этапов дипломного проекта | Сроки |
| 1 | Предпроектное исследование предметной области | 22.02.25 |
| 2 | Разработка технического задания | 28.02.25 |
| 3 | Провести проектирование программного продукта. | 15.03.25 |
| 4 | Разработка программного продукта | 30.04.25 |
| 5 | Тестирование и отладка программного продукта | 06.05.25 |
| 6 | Составление программной документации | 11.05.25 |
| 7 | Рассчитать стоимость разработки и внедрения программного продукта | 16.05.25 |
| 8 | Защита дипломного проекта | 19.05.25 |

Приложение Б – Листинг таблицы

@login\_required

def equipment\_list(request):

if not check\_view\_permission(request.user):

messages.error(request, 'У вас нет прав для просмотра оборудования')

return redirect('home')

if request.method == 'POST' and (check\_admin\_permission(request.user) or check\_senior\_nurse\_permission(request.user)):

equipment\_ids = request.POST.getlist('equipment\_ids')

if equipment\_ids:

try:

with transaction.atomic():

Equipment.objects.filter(id\_\_in=equipment\_ids).delete()

for index, equipment in enumerate(Equipment.objects.all().order\_by('number'), 1):

equipment.number = index

equipment.save()

messages.success(request, "Оборудование успешно удалено и перенумеровано.")

except Exception as e:

messages.error(request, f"Ошибка при удалении и перенумеровании оборудования: {e}")

else:

messages.warning(request, "Не выбрано оборудование для удаления.")

return redirect('equipment\_list')

page\_size = int(request.GET.get('page\_size', 100))

page = int(request.GET.get('page', 1))

start = (page - 1) \* page\_size

end = start + page\_size

search\_query = request.GET.get('q', '')

equipment\_query = Equipment.objects.all().order\_by('number')

if search\_query:

equipment\_query = equipment\_query.filter(

Q(name\_\_icontains=search\_query) |

Q(inventory\_number\_\_icontains=search\_query) |

Q(serial\_number\_\_icontains=search\_query) |

Q(registration\_certificate\_\_icontains=search\_query) |

Q(department\_\_icontains=search\_query) |

Q(cabinet\_\_icontains=search\_query) |

Q(service\_organization\_\_icontains=search\_query) |

Q(supplier\_\_icontains=search\_query)

)

total\_count = equipment\_query.count()

equipment\_list = equipment\_query[start:end]

if request.headers.get('X-Requested-With') == 'XMLHttpRequest':

equipment\_data = []

for equipment in equipment\_list:

has\_images = EquipmentImage.objects.filter(equipment=equipment).exists()

equipment\_data.append({

'id': equipment.id,

'number': equipment.number,

'name': equipment.name,

'inventory\_number': equipment.inventory\_number or '',

'serial\_number': equipment.serial\_number or '',

'registration\_certificate': equipment.registration\_certificate or '',

'date\_of\_registration': equipment.date\_of\_registration or '',

'valid\_until': equipment.valid\_until or '',

'year\_of\_manufacture': equipment.year\_of\_manufacture or '',

'valid\_to': equipment.valid\_to or '',

'department': equipment.department or '',

'cabinet': equipment.cabinet or '',

'verification\_period': equipment.verification\_period or '',

'last\_verification\_date': equipment.last\_verification\_date or '',

'service\_organization': equipment.service\_organization or '',

'supplier': equipment.supplier or '',

'has\_images': has\_images

})

return JsonResponse({

'equipment': equipment\_data,

'total': total\_count,

'page': page,

'page\_size': page\_size,

'pages': (total\_count + page\_size - 1) // page\_size

})

context = {

'equipment\_list': equipment\_list,

'user\_groups': request.user.groups.values\_list('name', flat=True),

'total\_count': total\_count,

'page': page,

'page\_size': page\_size,

'pages': (total\_count + page\_size - 1) // page\_size,

'search\_query': search\_query

}

return render(request, 'equipment\_list.html', context)