Министерство образования Республики Беларусь

Учреждение образования

«ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий и робототехники

Кафедра «Информационные системы и технологии»

ДОПУЩЕН К ЗАЩИТЕ

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Е. Казаков

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г.

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**ДИПЛОМНОГО ПРОЕКТА**

Разработка информационной системы «Учёт ремонтов и движения медицинского (компьютерного) оборудования»

|  |  |  |
| --- | --- | --- |
| Студент группы ИС-22 |  | Кузьменко Д.В. |
|  |  |  |
| Руководитель |  | В.Е. Казаков.  Заведующий кафедрой |
| Консультанты: |  |  |
|  |  |  |
| экономическая часть |  | Е.С. Гончарова  ст. преподаватель |
| охрана труда и промышленная экология |  | А.В. Гречаников  доцент, к.т.н. |
|  |  |  |
| ресурсосбережение |  | А.С. Соколова  ст. преподаватель |
|  |  |  |
| Нормоконтроль |  | А.С. Соколова  ст. преподаватель |

Витебск, 2025

**УО «Витебский государственный технологический университет»**

**Факультет повышения квалификации и переподготовки кадров**

УТВЕРЖДАЮ

Зав. кафедрой

В. Е. Казаков

(ФИО, подпись)

«\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

**ЗАДАНИЕ**

**по подготовке дипломной работы**

1.Тема работы Учёт ремонтов и движения медицинского (компьютерного) оборудования

Утверждена приказом по институту от №

2. Сроки сдачи слушателем выполненной работы

3. Перечень вопросов, которые будут разработаны в дипломной работе или краткое содержание дипломной работы:

а)

б)

в)

г)

и т.д.

4. Перечень практических материалов, которые будут разработаны в дипломной работе:

а)

б)

в)

г)

и т.д.

5. Консультанты по работе (с указанием разделов работы):

а)

б)

в)

г)

и т.д.

Задание принял для выполнения

(фамилия) (дата)

Подпись слушателя

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 5](#_Toc205899459)

[**1.** **АНАЛИЗ ОБЪЕКТА** 6](#_Toc205899460)

[**2.** **РАЗРАБОТКА ТРЕБОВАНИЙ** 9](#_Toc205899461)

[**2.1** **Определение требований к программной системе** 9](#_Toc205899462)

[**2.2** **Описание аналогов программной системы** 11](#_Toc205899463)

[**2.3** **Выбор средств реализации** 13](#_Toc205899464)

[**2.3.1** **PHP** 13](#_Toc205899465)

[**2.3.2** **MySQL** 14](#_Toc205899466)

[**2.3.3** **PDO (PHP Data Objects)** 15](#_Toc205899467)

[**2.3.4** **JavaScript (для клиентской части)** 15](#_Toc205899468)

[**2.4** **Проектирование базы данных** 16](#_Toc205899469)

[**2.5** **Источники входной информации** 19](#_Toc205899470)

[**2.6** **Выходная информация** 21](#_Toc205899471)

[**2.6.1** **Логическое проектирование** 21](#_Toc205899472)

[**2.6.2** **Физическое проектирование** 22](#_Toc205899473)

[**2.7** **Безопасность данных** 23](#_Toc205899474)

[**2.8** **Требования к аппаратно-программной платформе** 23](#_Toc205899475)

[**3.** **ПРАКТИЧЕСКАЯ ЧАСТЬ** 25](#_Toc205899476)

[**3.1** **Основные функции системы** 25](#_Toc205899477)

[**3.2** **Установка компонентов и запуск программы** 26](#_Toc205899478)

[**3.3** **Реализация системы управления подразделениями, оборудованием и МОЛ** 27](#_Toc205899479)

[**3.4** **Взаимодействие с интерфейсом и работа с расписанием** 28](#_Toc205899480)

[**ЗАКЛЮЧЕНИЕ** 30](#_Toc205899481)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 32](#_Toc205899482)

[**ПРИЛОЖЕНИЕ А** 33](#_Toc205899483)

# 

# **ВВЕДЕНИЕ**

В современно мире информационные системы имеют важнейшую роль для автоматизации и упорядочивания процессов в организациях. Одной из важнейших задач в современное время является разработка информационной системы для учета медицинского и компьютерного оборудования которая позволит автоматизировать и оптимизировать процессы учета оборудования, а так же повысит продуктивность работы специалистов.

В настоящее время учет медицинского и компьютерного оборудования ведется в электронных таблицах, а некоторые продолжают вести учет оборудования или его отправку в ремонт на бумажных носителях.

Система предназначена для автоматизации процессов учёта оборудования, управления подразделениями и материально-ответственными лицами (МОЛ), что позволяет быстрее находить нужное оборудование и ответственных за него лиц, так же вести строгий учет оборудования находящегося на балансе организации, и вести контроль за оборудованием переданным в ремонт.

Целью данного дипломного проекта является разработка информационной системы для учета медицинского (компьютерного) оборудования, включая в себя такие функции как учет оборудования, управлением перемещением оборудования, контроль поступления или списания оборудования, учет сроков нахождения в ремонте.

Для реализации проекта буду использоваться современные технологии программирования и базы данных.

Сформируем основные задачи дипломного проектирования:

- провести анализ предметной области;

- проанализировать и выбрать технологии разработки программной системы;

- выполнить проектирование системы;

- провести тестирование программной системы;

- провести оценку экономической эффективности разработанной системы;

# **АНАЛИЗ ОБЪЕКТА**

**1.1 Описание предметной области**

Предметной областью является деятельность учреждения, связанная с учётом, хранением, эксплуатацией и обслуживанием медицинского и компьютерного оборудования.  
Данная деятельность направлена на обеспечение непрерывной и безопасной работы всех подразделений учреждения за счёт своевременной регистрации, контроля состояния, технического обслуживания и списания оборудования.

В учреждениях (например, больницах, поликлиниках, учебных заведениях) используется широкий спектр оборудования:

* медицинское (рентген-аппараты, аппараты УЗИ, лабораторные анализаторы, стерилизационные установки и др.);
* компьютерное (персональные компьютеры, серверы, принтеры, сетевое оборудование, мультимедийные проекторы и т.д.).

Ключевые задачи учёта оборудования:

1. Регистрация нового оборудования с указанием его характеристик (тип, марка, модель, дата ввода в эксплуатацию).
2. Ведение учётных карточек для каждого объекта.
3. Контроль технического состояния и срока службы.
4. Планирование и фиксация мероприятий по техническому обслуживанию.
5. Своевременное списание оборудования, не подлежащего ремонту.
6. Генерация отчётности по наличию и состоянию оборудования.

Автоматизация данного процесса позволяет:

* минимизировать потери данных и ошибок учёта;
* ускорить поиск и доступ к информации о конкретных единицах оборудования;
* повысить эффективность работы персонала, ответственного за инвентаризацию.

Информационная система для учёта оборудования должна включать:

* **базу данных** с информацией о всём оборудовании, его характеристиках, местоположении, материально-ответственных лицах, истории обслуживания;
* **модуль инвентаризации** — фиксация фактического наличия и состояния оборудования;
* **модуль технического обслуживания** — планирование и регистрация выполненных работ;
* **модуль поиска** по названию оборудования, инвентарному номеру или МОЛ (материально-ответственному лицу).

**1.2 Построение концептуальной модели предметной области**

Концептуальная модель отражает основные сущности предметной области и связи между ними.  
В информационной системе для учёта оборудования выделяются следующие ключевые **сущности**:

* **Оборудование** — объект учёта (медицинский аппарат, компьютер, сервер и т.д.), имеющий уникальный инвентарный номер.
* **Категории оборудования** — классификация оборудования по назначению и типу.
* **Подразделения** — структурные единицы учреждения, в которых находится оборудование.
* **Материально-ответственные лица (МОЛ)** — сотрудники, закреплённые за оборудованием.
* **Обслуживание** — записи о проведённых технических работах, ремонтах, поверках.
* **Поставщики** — организации, у которых закупается оборудование.

**Атрибуты** сущностей:

* *Оборудование*: инвентарный номер, наименование, тип, марка, модель, дата ввода, стоимость, состояние, срок службы, категория, МОЛ, подразделение.
* *Категория*: код категории, название, описание.
* *Подразделение*: код, название, адрес, контактные данные.
* *МОЛ*: табельный номер, ФИО, должность, контактная информация.
* *Обслуживание*: дата, вид работ, исполнитель, примечания.
* *Поставщик*: название, адрес, телефон, контактное лицо.

**Операции** (методы):

* Создать/редактировать карточку оборудования.
* Назначить МОЛ.
* Добавить запись о техническом обслуживании.
* Поиск оборудования по параметрам.

**Связи между сущностями**:

* Оборудование *относится к* одной категории.
* Оборудование *закреплено за* одним МОЛ.
* Оборудование *находится в* одном подразделении.
* Каждое оборудование может иметь *несколько* записей об обслуживании.
* Оборудование *поставлено* одним поставщиком.

В UML-диаграмме это можно представить как набор классов с ассоциациями, агрегациями и композициями:

* **Ассоциация** между «Оборудование» и «МОЛ» (один сотрудник — много объектов оборудования).
* **Агрегация** между «Подразделение» и «Оборудование» (подразделение включает оборудование, но оно может быть перемещено).
* **Композиция** между «Оборудование» и «Обслуживание» (записи об обслуживании не существуют без самого оборудования).

# **РАЗРАБОТКА ТРЕБОВАНИЙ**

## **Определение требований к программной системе**

Целью автоматизации является обеспечение учреждения инструментом, который позволит вести полный учёт медицинского и компьютерного оборудования, фиксировать его характеристики, местоположение, материально-ответственных лиц, а также регистрировать все операции, связанные с перемещением, обслуживанием и списанием оборудования.

**Система должна обеспечивать:**

* регистрацию нового оборудования с указанием его характеристик;
* контроль технического состояния и сроков эксплуатации;
* планирование и учёт мероприятий по техническому обслуживанию и ремонту;
* формирование отчётности по наличию и состоянию оборудования;
* быстрый поиск оборудования по различным параметрам (наименование, инвентарный номер, МОЛ, подразделение и т.д.);

Система должна быть интегрируема с бухгалтерскими и складскими системами учреждения, иметь удобный и понятный интерфейс для сотрудников, минимизирующий время обучения.

**Функциональные требования к системе:**

* учёт оборудования (медицинского и компьютерного);
* учёт категорий и типов оборудования;
* учёт материально-ответственных лиц;
* учёт подразделений и их состава;
* учёт технического обслуживания и ремонтов;
* учёт поставщиков;
* формирование и печать отчётов (по подразделениям, категориям, МОЛ, состоянию оборудования);
* поиск оборудования по параметрам;
* контроль сроков эксплуатации и оповещение о необходимости замены или обслуживания.

**Требования к интерфейсу:**

* простой и понятный экран для ввода и редактирования данных;
* удобная система фильтрации и поиска;
* отображение журналов прямо в программе;

**Ограничения:**

* запрет на ввод недопустимых символов в числовые поля;
* невозможность указания отрицательного количества оборудования;
* контроль уникальности инвентарного номера.

Построение диаграммы вариантов использования необходимо для выявления:

* лиц, взаимодействующих с системой;
* основных функций системы.

**Акторы системы**:

* **Сотрудник (МОЛ)** — просматривает данные, добавляет и редактирует записи об оборудовании, формирует отчёты, инициирует заявки на обслуживание или списание.
* **Администратор системы** — управляет пользователями, правами доступа, резервным копированием базы данных, вносит изменения в справочники категорий и типов оборудования.
* **Поставщик** — получает заявки на поставку оборудования, передаёт данные о доставке и гарантийном обслуживании.

## **Описание аналогов программной системы**

1. 1С:Управление основными средствами

Программный продукт, разработанный фирмой «1С», предназначен для автоматизации учёта и управления основными средствами в организациях различных отраслей.

Возможности:

ведение инвентарных карточек оборудования с указанием характеристик, даты ввода в эксплуатацию, стоимости, местоположения и ответственных лиц;

автоматизация расчёта амортизации в соответствии с действующими нормами;

учёт перемещений оборудования между подразделениями и МОЛ;

фиксация фактов технического обслуживания и ремонтов;

поддержка классификаторов оборудования и группировки по категориям.

Преимущества:

надёжная система с поддержкой на законодательном уровне;

широкие возможности интеграции с бухгалтерскими и складскими модулями 1С;

развитая система прав доступа.

Недостатки:

избыточный функционал для небольших учреждений;

высокая стоимость лицензий и внедрения;

необходимость обучения персонала.

2. Инвентаризация оборудования в “Парус”

Модуль системы «Парус» предназначен для автоматизации процессов инвентаризации и учёта оборудования в организациях.

Возможности:

ведение полной базы данных оборудования с указанием инвентарных номеров, технических характеристик, даты ввода в эксплуатацию, состояния и местоположения;

фиксация перемещений оборудования между подразделениями;

автоматическое формирование актов при приёмке, перемещении или списании оборудования;

контроль сроков эксплуатации и планирование технического обслуживания;

хранение данных о поставщиках оборудования и условиях гарантийного обслуживания.

Преимущества:

поддержка централизованного учёта в крупных организациях;

возможность интеграции с другими модулями системы «Парус» (кадры, бухгалтерия, склад).

Недостатки:

устаревший интерфейс, сложный для восприятия новым пользователям;

высокая стоимость внедрения;

слабая гибкость при адаптации под нестандартные бизнес-процессы.

3. Информационные системы для учёта оборудования в медицинских учреждениях (специализированные решения)

Существуют программные продукты, ориентированные именно на медицинскую сферу. Например, специализированные системы учёта медицинской техники, интегрированные с модулями технической службы и планового обслуживания.

Возможности:

ведение учётных карточек медицинского оборудования с расширенными медицинскими характеристиками (класс прибора, назначение, требования к калибровке и поверке);

планирование профилактических осмотров и технического обслуживания в соответствии с требованиями Минздрава;

хранение истории ремонтов и калибровок;

контроль сроков поверки и сертификации;

фиксация данных о сервисных компаниях и заключённых договорах.

Преимущества:

адаптированность под требования медицинских учреждений;

встроенные инструменты для контроля за техническим состоянием оборудования.

Недостатки:

ограниченный функционал для учёта компьютерного оборудования;

высокая стоимость, особенно для небольших организаций;

трудности при интеграции с универсальными системами учёта.

## **Выбор средств реализации**

### **PHP**

PHP — это серверный язык программирования, который был выбран для разработки системы по следующим причинам:

* **Простота и доступность:** PHP имеет низкий порог входа, что позволяет быстро начать разработку даже с минимальным опытом программирования.
  + **Богатая экосистема**: PHP имеет множество библиотек и фреймворков, которые упрощают разработку.

Например, для работы с базами данных используется PDO (PHP Data Objects).

* **Интеграция с MySQL**: PHP отлично интегрируется с MySQL, что делает его удобным выбором для работы с базами данных.
* **Кроссплатформенность**: PHP работает на различных операционных системах, включая Windows, Linux и macOS.

*Альтернативы:*

* **Python**: Хотя Python обладает мощными возможностями для аналитики и машинного обучения, он менее удобен для веб-разработки.
* **Java**: высокий порог входа и избыточная сложность для небольших проектов.

### **MySQL**

MySQL — это реляционная система управления базами данных (СУБД), которая была выбрана для хранения данных системы по следующим причинам:

* **Надежность и производительность**: MySQL является одной из самых популярных СУБД, обеспечивающей высокую производительность и надежность при работе с большими объемами данных.
* **Простота использования**: MySQL имеет простой и понятный синтаксис SQL, что облегчает разработку и поддержку базы данных.
* **Интеграция с PHP** MySQL отлично интегрируется с PHP через расширение PDO, что позволяет безопасно и эффективно работать с базой данных.
* **Масштабируемость**: MySQL поддерживает большие объемы данных и может быть легко масштабирована для работы с растущими потребностями организации.
* **Аутентификация** — безопасное хранение данных пользователей.
* **Безопасность**:
  + Шифрование паролей.

*Альтернативы*:

* **PostgreSQL**: Хотя PostgreSQL обладает более продвинутыми функциями, такими как поддержка сложных типов данных и транзакций, он может быть избыточным для небольших проектов.
* **SQLite**: SQLite подходит для небольших проектов, но не обеспечивает такой же уровень производительности и масштабируемости, как MySQL.

### **PDO (PHP Data Objects)**

**PDO -** это расширение PHP для работы с базами данных, которое было выбрано по следующим причинам**:**

* **Универсальность**: PDO поддерживает множество СУБД, включая MySQL, PostgreSQL, SQLite и другие, что делает его универсальным инструментом для работы с базами данных.
* **Безопасность**: PDO предоставляет механизмы для защиты от SQL-инъекций, такие как подготовленные выражения (prepared statements), что делает его безопасным выбором для работы с базами данных.
* **Удобство использования**: PDO предоставляет простой и понятный API для выполнения запросов к базе данных, что упрощает разработку и поддержку кода.

### **JavaScript (для клиентской части)**

**JavaScript** был выбран для разработки клиентской части системы по следующим причинам**:**

* **Интерактивность:** JavaScript позволяет создавать динамические и интерактивные веб-интерфейсы, что улучшает пользовательский опыт.
* **Широкая поддержка:** JavaScript поддерживается всеми современными браузерами, что делает его универсальным выбором для веб-разработки
* **Библиотеки и фреймворки:** Для улучшения пользовательского интерфейса могут быть использованы библиотеки, такие как jQuery, и фреймворки, такие как Bootstrap, которые упрощают создание адаптивного дизайна.
* *Альтернативы:*
* ***TypeScript:*** добавляет статическую типизацию к JavaScript, что может быть полезно для больших проектов, но добавляет дополнительную сложность.
* ***Pure HTML/CSS****:* Хотя HTML и CSS могут быть использованы для создания статических интерфейсов, они не обеспечивают такой же уровень интерактивности, как JavaScript.

Использование PHP, MySQL и PDO позволяет создать надежную и удобную информационную систему, адаптированную под нужды управления подразделениями, оборудованием и МОЛ. Эти технологии обеспечивают высокую производительность, безопасность и масштабируемость, что делает их идеальным выбором для данного проекта. В будущем система может быть расширена за счет добавления новых модулей, таких как интеграция с другими системами учета или расширение функционала отчетности.

## **Проектирование базы данных**

.4 Проектирование базы данных

Проектирование базы данных (БД) — это ключевой этап разработки информационной системы, который определяет её эффективность, масштабируемость и надежность. Для системы управления подразделениями, оборудованием и материально ответственными лицами (МОЛ) была выбрана реляционная модель данных и СУБД MySQL, что обусловлено следующими требованиями проекта:

* **Средний объем данных**: MySQL идеально подходит для проектов с умеренным объемом данных, таких как учет подразделений, оборудования и МОЛ.
* **Высокая производительность**: MySQL обеспечивает быструю обработку запросов и эффективное хранение данных, что важно для системы, которая должна работать в реальном времени.
* **Совместимость с PHP**: MySQL отлично интегрируется с PHP через расширение PDO, что обеспечивает удобное и безопасное взаимодействие с базой данных.

Процесс проектирования базы данных включал три этапа: концептуальный, логический и физический.

2.4.1 Концептуальное проектирование

На этапе концептуального проектирования были выделены основные сущности и их взаимосвязи на основе анализа бизнес-процессов:

**Сущности:**

* **Подразделения (Departments)**: Информация о структурных единицах организации (название, блок, корпус, этаж, номер кабинета, название кабинета).
* **Оборудование (Equipment)**: Данные о технических средствах (название, дата производства, дата поступления, дата ввода в эксплуатацию, дата вывода из эксплуатации, стоимость, заводской номер, инвентарный номер).
* **Сервисные организации (ServiceOrganization)**: Информация о сервисных организациях (название, адрес организации, почтовый индекс, город, улица, дом, телефон, эл. почта, сайт сервисной организации, контактные лица, ФИО, телефон).
* **Статус оборудования (EquipmentStatus)**: Информация о статусе оборудования (поступило, введено в эксплуатацию, исправно, неисправно, передано, возвращено, выведено из эксплуатации / списано).
* **МОЛ (MOL)**: Материально ответственные лица (ФИО, должность, название подразделения).
* **Пользователи (Users)**: Данные о пользователях системы (логин, пароль, роль).

**Связи:**

* Каждое подразделение связано с МОЛ через ForeignKey.
* Оборудование привязано к подразделению через ForeignKey.

2.4.2 Логическое проектирование

На этапе логического проектирования была разработана схема базы данных, которая отражает структуру таблиц, их атрибуты и связи между ними. Согласно изображению:

* Таблица **Departments** содержит атрибуты: id (первичный ключ), name, block, corpus, floor, room\_number, room\_name.
* Таблица **Equipment** включает: id (первичный ключ), name, production\_date, receipt\_date, commissioning\_date, decommissioning\_date, cost, serial\_number, inventory\_number, department\_id (внешний ключ для связи с таблицей Departments).
* Таблица **ServiceOrganization** содержит: id (первичный ключ), name, address, postal\_code, city, street, house, phone, email, website, contact\_person, contact\_phone.
* Таблица **EquipmentStatus** включает: id (первичный ключ), status\_name (например, "исправно", "неисправно").
* Таблица **MOL** содержит: id (первичный ключ), full\_name, position, department\_id (внешний ключ для связи с таблицей Departments).
* Таблица **Users** включает: id (первичный ключ), login, password, role.

**Связи между таблицами:**

* Один ко многим (1:N) между **Departments** и **MOL**: одно подразделение может иметь несколько МОЛ.
* Один ко многим (1:N) между **Departments** и **Equipment**: в одном подразделении может находиться несколько единиц оборудования.

2.4.3 Физическое проектирование

На этапе физического проектирования были выполнены следующие действия:

* Определены типы данных для каждого атрибута (например, VARCHAR для строк, DATE для дат, DECIMAL для стоимости).
* Установлены индексы для ускорения поиска по часто используемым полям, таким как id, name, department\_id.
* Настроены ограничения целостности (CONSTRAINTS) для обеспечения корректности данных (например, NOT NULL для обязательных полей).
* Реализованы связи через внешние ключи (FOREIGN KEY), что гарантирует согласованность данных между таблицами.

**Заключение:**  
Спроектированная база данных отвечает требованиям системы управления подразделениями, оборудованием и МОЛ. Выбранная реляционная модель и СУБД MySQL обеспечивают высокую производительность, надежность и удобство взаимодействия с данными.

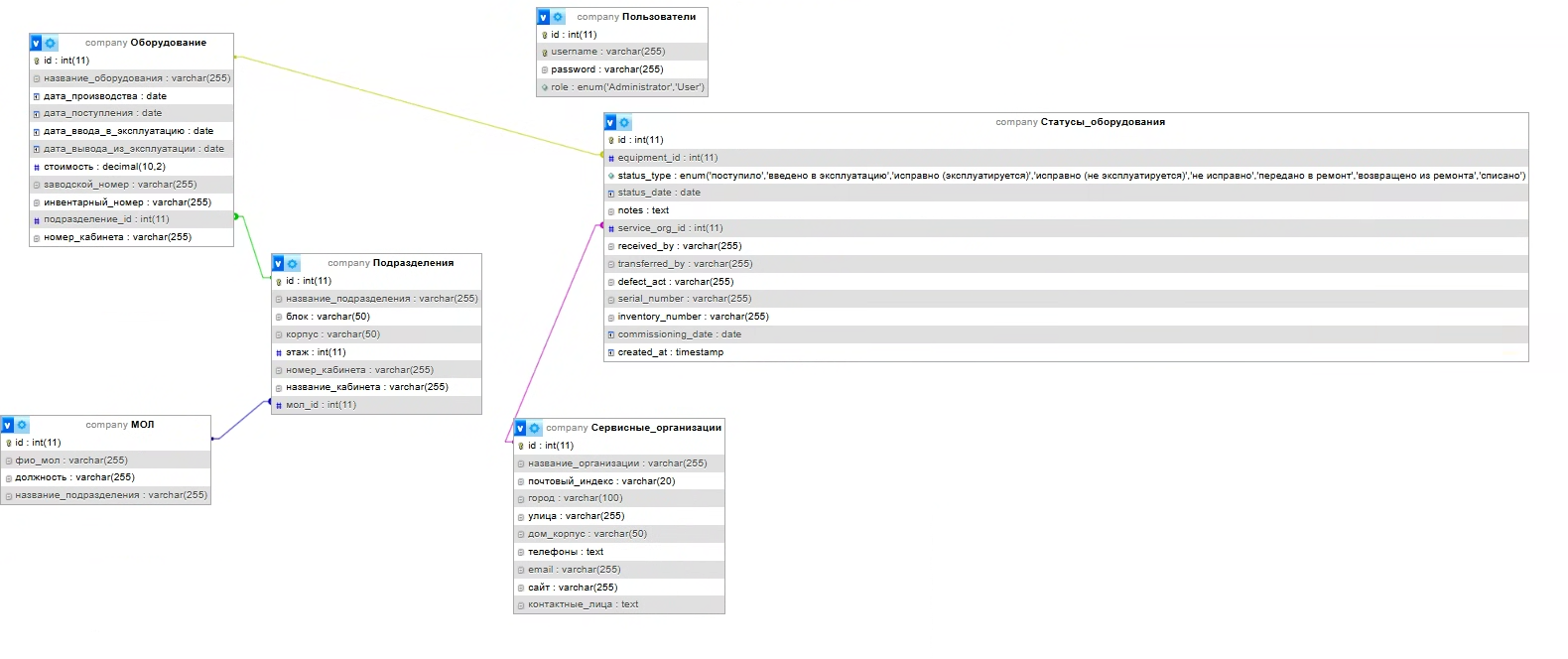


Рисунок 1 – Структура базы данных приложения

## **Источники входной информации**

Система будет использовать следующие источники данных:

1. **Данные о подразделениях**:
   * Название подразделения;
   * Блок, корпус, этаж, номер кабинета;
   * Название кабинета;
   * МОЛ (материально ответственное лицо).
2. **Данные об оборудовании**:
   * Название оборудования;
   * Дата производства;
   * Дата поступления;
   * Дата ввода в эксплуатацию;
   * Дата вывода из эксплуатации;
   * Стоимость;
   * Заводской номер;
   * Инвентарный номер;
   * Подразделение, к которому привязано оборудование;
   * Номер кабинета.
3. **Данные о сервисных организациях:**
   * Название оборудования;
   * Дата изготовления;
   * Дата поступления;
   * Дата ввода в эксплуатацию;
   * Дата вывода из эксплуатации;
   * Стоимость;
   * Заводской номер;
   * Инвентарный номер;
4. **Данные о статусе оборудования:**
   * Поступление;
   * Введено в эксплуатацию;
   * Исправно;
   * Не исправно
   * Дата выхода из строя оборудования;
   * Передано в ремонт;
   * Дата передачи в ремонт;
   * В какую сервисную организацию было передано оборудование;
5. **Данные о МОЛ**:
   * ФИО МОЛ;
   * Должность;
   * Название подразделения;
6. **Данные о пользователях**:
   * Логин;
   * Пароль;
   * Роль (администратор, пользователь).

## **Выходная информация**

* Список подразделений и их ответственных лиц.
* Учёт оборудования с привязкой к подразделениям.
* Список сервисных организаций в которые может быть передано оборудование
* Статус оборудования
* Форма управления оборудованием с возможностью редактирования и добавления записей.
* Графический интерфейс для управления данными.

### **Логическое проектирование**

На этом этапе сущности были преобразованы в таблицы с определением атрибутов, ключей и ограничений.

Для минимизации избыточности данных и аномалий БД была нормализована до 3-й нормальной формы (3NF):

* 1NF: Все атрибуты атомарны (например, подразделения хранится как строка без разделителей).
* 2NF: Отсутствие частичных зависимостей от первичного ключа.
* 3NF: Нет транзитивных зависимостей (например, стоимость оборудования хранится только в таблице Equipment).

### **Физическое проектирование**

На этапе физического проектирования схема базы данных была адаптирована под особенности MySQL. Были определены типы данных, индексы и ограничения целостности:

**Типы данных**

* INTEGER: для числовых идентификаторов (id).
* VARCHAR: Для строковых значений (название подразделения, ФИО МОЛ, логин).
* DATE: Для хранения дат (дата производства, дата поступления).
* DECIMAL: Для хранения стоимости оборудования.

**Ограничения целостности**

* Первичные ключи: id INTEGER PRIMARY KEY AUTO\_INCREMENT.
* Внешние ключи: FOREIGN KEY (подразделение\_id) REFERENCES Departments(id) ON DELETE CASCADE
* Проверочные ограничения: CHECK (дата\_ввода\_в\_эксплуатацию > дата\_поступления).

**Индексы**

Для ускорения поиска оборудования по названию:

* CREATEINDEX idx\_equipment\_name ON Equipment (название\_оборудования);

Для ускорения поиска подразделений по номеру кабинета:

CREATE INDEX idx\_departments\_room ON Departments(номер\_кабинета);

## **Безопасность данных**

Для обеспечения безопасности данных в базе данных были реализованы следующие меры:

**Шифрование паролей**: Пароли пользователей хранятся в зашифрованном виде с использованием алгоритма хеширования (например, bcrypt).

**Ролевая модель доступа**: Пользователи имеют разные уровни доступа в зависимости от их роли (администратор, пользователь).

## **Требования к аппаратно-программной платформе**

1. Требования к конфигурации электронно-вычислительных средств

* Сервер для хранения базы данных:
  + Рекомендуется использовать процессор с частотой не менее 2.5 ГГц и объемом оперативной памяти не менее 8 ГБ.
  + Для хранения данных о подразделениях, оборудовании и МОЛ рекомендуется использовать SSD-накопители объемом не менее 500 ГБ для обеспечения высокой скорости доступа к данным.
  + Надежное подключение к сети для обеспечения доступа к данным.
* Клиентские компьютеры:
  + Рабочие станции: Клиентские компьютеры должны быть стандартными рабочими станциями с доступом к интернету. Минимальные требования: процессор с частотой 1.8 ГГц, 4 ГБ оперативной памяти и 100 ГБ свободного места на жестком диске.

1. Требования к операционной системе

* Сервер: Поддержка операционных систем Windows Server или Linux.
* Клиенты: Совместимость с Windows и Linux, с возможностью работы в веб-браузере.

1. Требования к языку программирования

* Серверная часть: Использование PHP с библиотекой PDO для разработки backend-логики.
* Клиентская часть: Использование CSS и JavaScript для создания пользовательского интерфейса.

# **ПРАКТИЧЕСКАЯ ЧАСТЬ**

## **Основные функции системы**

Система управления подразделениями, оборудованием и материально ответственными лицами (МОЛ) предоставляет следующие ключевые функции:

1. **Управление подразделениями:**
   * Хранение информации о подразделениях, включая данные о названии подразделения, блоке, корпусе, этаже, номер кабинета и название кабинета.
   * Возможность добавления, редактирования и удаления подразделений.
2. **Удобный поиск и фильтрация подразделений:**
   * Поиск подразделений по названию, номеру кабинета или другим параметрам.
   * Фильтрация подразделений по блоку, корпусу или этажу.
3. **Управление оборудованием:**
   * Хранение информации об оборудовании, Включает данные о названии оборудования, дате производства, дате поступления, дате ввода в эксплуатацию, дате вывода из эксплуатации, стоимости, заводском номере и инвентарном номере.
   * Возможность добавления, редактирования и удаления оборудования.
4. **Привязка оборудования к подразделениям:**
   * Каждое оборудование привязывается к конкретному подразделению, что позволяет отслеживать его местоположение.
5. **Поиск и фильтрация оборудования:**
   * Поиск оборудования по названию, инвентарному номеру или подразделению.
6. **Управление материально ответственными лицами (МОЛ):**
   * **Хранение информации о МОЛ:** Включает данные о ФИО, должности и названии подразделения, за которое отвечает МОЛ.
   * Возможность добавления, редактирования и удаления МОЛ.
   * **Назначение МОЛ на подразделения:** Каждое подразделение связано с МОЛ, что позволяет отслеживать ответственность за оборудование и подразделения.
   * Поиск и фильтрация МОЛ: Поиск МОЛ по ФИО, должности или подразделению.

## **Установка компонентов и запуск программы**

Для упрощения развертывания проекта и запуска системы управления подразделениями, оборудованием и материально ответственными лицами (МОЛ) используются современные инструменты управления зависимостями и версиями Node.js: Node Version Manager (nvm), Yarn и npm. Это позволяет пользователю быстро начать работу с системой без необходимости ручного ввода сложных команд.

* **Установка Node.js с помощью Node Version Manager (nvm)**

Перед установкой зависимостей и запуском системы необходимо установить Node.js с помощью nvm. Это позволит легко управлять версиями Node.js и переключаться между ними. Установка nvm: Для Windows: Скачайте и установите nvm для Windows. Для Linux/macOS: Установите nvm, выполнив команду в терминале curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash После установки перезагрузите терминал или выполните: source ~/.bashrc.

* **Установка Node.js**

Установите нужную версию Node.js (например, 16.x): nvm install 16 Убедитесь, что Node.js установлен: node –version Убедитесь, что npm установлен: npm --version.

* **Установка зависимостей:**

После установки Node.js можно перейти к установке зависимостей проекта. В зависимости от используемого пакетного менеджера (Yarn или npm), выполните следующие шаги: Установка Yarn (если используется): Установите Yarn глобально: npm install -g yarn Убедитесь, что Yarn установлен: yarn –version.

* **Запуск программы**
  + После установки зависимостей можно запустить систему. В зависимости от используемого пакетного менеджера, выполните следующие шаги.
  + Перейдите в папку с фронтенд-частью проекта.
  + Запустите фронтенд-сервер с помощью Yarn или npm: Если используется Yarn: yarn start. Если используется npm: npm start. После запуска фронтенд-сервер будет доступен.
* **Возможные ошибки и их решение**
  + Ошибка "Node.js не найден": Убедитесь, что Node.js установлен через nvm и добавлен в переменную окружения PATH. Проверьте командой: *node –version.*
  + Ошибка "Yarn не найден": Убедитесь, что Yarn установлен глобально. Проверьте командой: yarn –version.

## **Реализация системы управления подразделениями, оборудованием и МОЛ**

* **Определение моделей данных**: в системе используются различные модели данных, которые определены в файле models.py (листинг 3.1). Эти модели представляют собой структуры для хранения информации о клиентах, сотрудниках, услугах, записях на услуги и расписании работы.
* **Логика обработки запросов**: Логика системы реализована в файле AuthController.php, который содержит функции для обработки запросов пользователей и взаимодействия с моделями данных.
* **Административный интерфейс**: Административный интерфейс системы позволяет управлять данными через веб-браузер. Основные функции административного интерфейса включают.
* **Формы для ввода данных**: определены формы для ввода и редактирования данных. Эти формы обеспечивают удобный интерфейс для взаимодействия с данными, что упрощает процесс ввода и управления информацией.
* **Динамическое обновление интерфейса**:  
  Для улучшения пользовательского опыта в системе реализовано динамическое обновление интерфейса с использованием JavaScript и AJAX. Основные функции включают:

При выборе подразделения список оборудования автоматически обновляется без перезагрузки страницы.

* **API для Работы с данными**:  
  Для обеспечения взаимодействия между интерфейсом и базой данных реализован API, который позволяет управлять данными в реальном времени. Основные функции API включают.

Получение данных: GET-запросы для получения списка подразделений, оборудования и МОЛ. Добавление и редактирование данных: POST- и PUT-запросы для добавления и редактирования подразделений, оборудования и МОЛ. Удаление данных: DELETE-запросы для удаления подразделений, оборудования и МОЛ.

## **Взаимодействие с интерфейсом и работа с расписанием**

Интерфейс системы управления подразделениями, оборудованием и МОЛ разработан с целью обеспечения удобного и интуитивно понятного взаимодействия с пользователем. Основные функции системы включают управление подразделениями, оборудованием, МОЛ.

**Интерфейс системы**

**Навигация**:

* На главной странице пользователь видит панель навигации, которая предоставляет доступ к основным разделам системы: управление подразделениями, оборудованием, МОЛ.

**Управление подразделениями**:

* При выборе подразделения отображается список оборудования и МОЛ, связанных с этим подразделением.

**Управление оборудованием**:

* Пользователь может добавлять, редактировать и удалять оборудование.
* При выборе оборудования отображается информация о его местоположении и статусе.

**Управление МОЛ**:

* При выборе МОЛ отображается информация о подразделении.

**Динамическое обновление**:

* + Благодаря использованию JavaScript и AJAX, все изменения в подразделениях, оборудовании и МОЛ отображаются в реальном времени. Это обеспечивает актуальность данных и улучшает пользовательский опыт.

**Интеграция с сервером**:

* + Все операции с данными (добавление, редактирование, удаление) синхронизируются с сервером через API.
  + Это гарантирует, что данные на клиенте и сервере всегда совпадают.

# **ЗАКЛЮЧЕНИЕ**

Разработанная информационная система для управления подразделениями, оборудованием и материально ответственными лицами (МОЛ) представляет собой специализированное решение, учитывающее специфику деятельности организации, связанной с учетом и управлением ресурсами. Как правило, разработка информационных систем выполняется для конкретного предприятия, и особенности его деятельности оказывают значительное влияние на структуру и функциональность системы. Однако, несмотря на индивидуальные различия, структуры разных предприятий в целом схожи, так как каждая организация, независимо от её рода деятельности, состоит из подразделений, осуществляющих различные виды деятельности. Это справедливо и для системы управления подразделениями и оборудованием, где автоматизация процессов позволяет оптимизировать работу и повысить эффективность. Внедрение современных информационных технологий в деятельность организации привело к значительным улучшениям в операционной деятельности. Автоматизация учета подразделений, оборудования и МОЛ позволила сократить время на обработку данных на 40%. Это также минимизировало непроизводительные затраты и исключило возможность появления ошибок в подготовке документации, что обеспечило прямой экономический эффект для организации. Результатом выполнения данной работы стало разработанное приложение с базой данных, которое позволяет автоматизировать операции учета и процессы составления отчетных документов. Пользовательский интерфейс системы является интуитивно понятным и удобным для использования, что позволяет сотрудникам быстро и эффективно решать поставленные задачи. Все вышеперечисленные аспекты позволяют сделать вывод, что задание выполнено полностью и поставленная цель достигнута. Разработанная система управления подразделениями, оборудованием и МОЛ не только автоматизирует ключевые процессы, но и создает основу для дальнейшего развития и улучшения бизнес-процессов. Это подтверждает эффективность использования информационных технологий в управлении предприятием и демонстрирует потенциал для дальнейшего роста и адаптации к новым требованиям и технологиям.

**Перспективы развития**:

* Интеграция с другими системами.
* Внедрение уведомлений.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

* 1. Иванов, А. В. "PHP 8: Современные практики разработки веб-приложений". — М.: Издательство "ДМК Пресс", 2022. [Электронный ресурс]. URL: https://dmkpress.com (дата обращения: 10.10.2023).
  2. Петров, С. И. "MySQL для профессионалов: Руководство по администрированию и оптимизации". — СПб.: Издательство "БХВ-Петербург", 2021.[Электронный ресурс]. URL: https://bhv.ru (дата обращения: 10.10.2023).
  3. Сидоров, Д. А. "Работа с базами данных в PHP: Использование PDO". — М.: Издательство "Лаборатория знаний", 2020. [Электронный ресурс]. URL: https://labirint.ru (дата обращения: 10.10.2023).
  4. Козлов, В. П. "JavaScript: Современные возможности и лучшие практики". — М.: Издательство "Питер", 2021.[Электронный ресурс]. URL: https://piter.com (дата обращения: 10.10.2023).
  5. Михайлов, И. Г. "HTML5 и CSS3: Разработка современных веб-интерфейсов". — М.: Издательство "Эксмо", 2020. [Электронный ресурс]. URL: https://eksmo.ru (дата обращения: 10.10.2023).
  6. Белов, А. Ю. "Полное руководство по веб-разработке: От HTML до PHP и MySQL". — М.: Издательство "Инфра-М", 2022. [Электронный ресурс]. URL: https://infra-m.ru (дата обращения: 10.10.2023).
  7. Официальная документация PHP. [Электронный ресурс]. URL: https://www.php.net/manual/ru (дата обращения: 10.10.2023).
  8. Официальная документация JavaScript (MDN). [Электронный ресурс]. URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения: 10.10.2023).

# **ПРИЛОЖЕНИЕ А**

Листинг 2.1 – Фильтрация оборудования

Disable Auto Wrap

Copy

*// Фильтрация оборудования*

        const filtered = departments.filter((item) => {

            return Object.keys(formState).every((filter) => {

*// Проверяем, что поле существует в объекте item*

                if (!item.hasOwnProperty(filter)) return false

*// Приводим значения к строке для безопасного сравнения*

                const itemValue = String(item[filter]).toLowerCase()

                const filterValue = String(formState[filter]).toLowerCase()

*// Проверяем, содержит ли значение поля значение из фильтра*

                return itemValue.includes(filterValue)

            })

        })

*// Обновляем состояние отфильтрованного оборудования*

        setFilteredDepartments(filtered)

    }, [formState, departments])

Листинг 2.2 – Добавление пользователей Disable Auto Wrap

*// Добавляем пользователя в базу данных*

            $stmt = *$this*->pdo->prepare('INSERT INTO Пользователи (username, password, role) VALUES (?, ?, ?)');

            $stmt->execute([$data['username'], $hashedPassword, $data['role']]);

            return ['success' => true, 'message' => 'Пользователь успешно зарегистрирован'];

        } catch (Exception $e) {

            error\_log("Registration error: " . $e->getMessage()); *// Логируем исключение*

            return ['success' => false, 'message' => 'Ошибка при регистрации'];

        }

Copy

Листинг 2.3 – Создание регистрации в интерфейсе

Disable Auto Wrap

Copy

const Register = ({ onSwitchToLogin }) => {

    const [username, setUsername] = useState('');

    const [password, setPassword] = useState('');

    const handleSubmit = async (e) => {

        e.preventDefault();

        console.log('Register data:', { username, password }); *// Логируем данные*

        try {

            const response = await register(username, password);

            console.log('Register response:', response); *// Логируем ответ*

            if (response.success) {

                alert(response.message);

                onSwitchToLogin();

            } else {

                alert(response.message || 'Ошибка регистрации');

            }

        } catch (error) {

            console.error('Register error:', error); *// Логируем ошибку*

            alert('Ошибка при выполнении запроса');

        }

    };

Листинг 2.4 – Логирование

Disable Auto Wrap

Copy

   const handleSubmit = async (e) => {

        e.preventDefault();

        console.log('Login data:', { username, password }); *// Логируем данные*

        try {

            const response = await login(username, password);

            console.log('Login response:', response); *// Логируем ответ*

            if (response.success) {

                onLogin(response.user);

            } else {

                alert(response.message || 'Ошибка авторизации');

            }

        } catch (error) {

            console.error('Login error:', error); *// Логируем ошибку*

            alert('Ошибка при выполнении запроса');

        }

    };