Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПО ПРОГРАММЕ БАКАЛАВРИАТА

09.03.04 Программная инженерия

(код, наименование ОПОП ВО: направление подготовки, направленность (профиль))

«Разработка программно-информационных систем»

Программное обеспечение для системы контроля и управления доступом на круизном судне

(название темы)

Дипломный проект

(вид ВКР: дипломная работа или дипломный проект)

Автор ВКР _____ Д.Д.Дараган

(подпись, дата) (инициалы, фамилия)

Группа ПО-02б

Руководитель ВКР _____ Е. И. Аникина

(подпись, дата) (инициалы, фамилия)

Нормоконтроль _____ А. А. Чаплыгин

(подпись, дата) (инициалы, фамилия)

ВКР допущена к защите:

Заведующий кафедрой _____ А. В. Малышев

(подпись, дата) (инициалы, фамилия)

Курск 2024 г.

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

УТВЕРЖДАЮ:

Заведующий кафедрой

_____

(подпись, инициалы, фамилия)

«____» _____ 20____ г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ ПО ПРОГРАММЕ БАКАЛАВРИАТА

Студента Дарагана Д.Д., шифр 20-06-0258, группа ПО-02б

1. Тема «Программное обеспечение для системы контроля и управления доступом на круизном судне» утверждена приказом ректора ЮЗГУ от «15» апреля 2024 г. № 1616-с.

2. Срок предоставления работы к защите «11» июня 2024 г.

3. Исходные данные для создания программной системы:

3.1. Перечень решаемых задач:

1) исследовать и проанализировать предметную область

2) спроектировать базу данных для будущей системы

3) сконструировать базу данных в соответствии с бизнес-правилами системы

4) заполнить базу данными, отражающими отношения внутри базы данных

5) сконструировать программную систему контроля и управления доступом на круизном судне

6) протестировать программную систему

3.2. Входные данные и требуемые результаты для программы:

1) Входными данными для программной системы являются: личные данные пользователей, отдыхающих на круизном судне; данные о наложенных штрафах и ограничениях; конфигурации дверей, к которым запрашивается доступ; данные о ситуации на судне.

2) Выходными данными для программной системы являются: данные каждой таблицы базы в формате набора полей, содержимое которых можно просматривать, изменять и удалять; данные о доступах каждого пассажира к каждой двери на круизном лайнере с учётом текущей ситуации на судне. ИТ-ресурсов, воронка заявок.

4. Содержание работы (по разделам):

4.1. Введение

4.1. Анализ предметной области

4.2. Техническое задание: основание для разработки, назначение разработки, требования к интерфейсу, требования к программной системе, построение ER-модели, нефункциональные требования к программной системе, требования к оформлению документации.

4.3. Технический проект: общая характеристика организации решения задачи, общие сведения о программной системе, основание выбора технологии проектирования, диаграмма размещения, описание архитектуры программной системы, проектирование пользовательского интерфейса программной системы.

4.4. Рабочий проект: спецификация компонентов и классов программной системы, тестирование программной системы, сборка компонентов программной системы.

4.5. Заключение

4.6. Список использованных источников

5. Перечень графического материала: ...

Руководитель ВКР		Е. И. Аникина
	(подпись, дата)	(инициалы, фамилия)
Задание принял к исполнению		Д.Д.Дараган
	(подпись, дата)	(инициалы, фамилия)

РЕФЕРАТ

Объем работы равен 69 страницам. Работа содержит 9 иллюстраций, 1 таблицу, 20 библиографических источников и 0 листов графического материала. Количество приложений – 2. Графический материал представлен в приложении А. Фрагменты исходного кода представлены в приложении Б.

Перечень ключевых слов: коммерческий сайт, Система, CMS, Битрикс, Joomla, аддитивные технологии, 3D-принтеры, услуги, сервисы, информатизация, автоматизация, информационные технологии, веб-форма, Apache, классы, база данных, средства защиты информации, подсистема, компонент, модуль, сущность, информационный блок, метод, контент-редактор, администратор, пользователь, web-сайт.

Объектом разработки является web-сайт компании, занимающейся производством 3D-принтеров, выпуском оборудования для создания порошков, разработкой программного обеспечения и организацией центров аддитивного производства.

Целью выпускной квалификационной работы является привлечение клиентов, увеличение заказов, информирование о продукции и услугах путем создания сайта компании.

В процессе создания сайта были выделены основные сущности путем создания информационных блоков, использованы классы и методы модулей, обеспечивающие работу с сущностями предметной области, а также корректную работу web-сайта, разработаны разделы, содержащие информацию о компании, ее деятельности, производимой продукции и услугах, разработан сервис по заказу 3D-деталей.

При разработке сайта использовалась система управления контентом «1С-Битрикс: Управление сайтом».

Разработанный сайт был успешно внедрен в компании.

ABSTRACT

The volume of work is 69 pages. The work contains 9 illustrations, 1 table, 20 bibliographic sources and 0 sheets of graphic material. The number of applications is 2. The graphic material is presented in annex A. The layout of the site, including the connection of components, is presented in annex B.

List of keywords: commercial website, System, CMS, Bitrix, Joomla, additive technologies, 3D printers, services, services, informatization, automation, information technology, web form, Apache, classes, database, component, module, entity, information block, method, content editor, administrator, user, web site.

The object of the research is the analysis of information technologies for the development of a production company's website.

The object of the development is the website of a company engaged in the production of 3D printers, the production of equipment for the creation of powders, software development and the organization of additive manufacturing centers.

The purpose of the final qualifying work is to attract customers, increase orders, inform about products and services by creating a company website.

In the process of creating the site, the main entities were identified by creating information blocks, classes and methods of modules were used to ensure work with the entities of the subject area, as well as the correct operation of the website, sections containing information about the company, its activities, products and services were developed, a service for ordering 3D parts was developed.

When developing the site, the content management system «1C – Bitrix: Site Management» was used.

The developed website was successfully implemented in the company.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД – база данных.

ИС – информационная система.

ИТ – информационные технологии.

UI - User Interface, пользовательский интерфейс.

ООП - объектно-ориентированное программирование.

ПО – программное обеспечение.

РП – рабочий проект.

СУБД – система управления базами данных.

ТЗ – техническое задание.

ТП – технический проект.

CRUD - Create, Read, Update, Delete, основные операции для работы с данными.

ER (Entity-Relationship model) – модель данных, позволяющая описывать концептуальные схемы предметной области. ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных.

СКУД – система контроля и управления доступом

УД – учётные данные

ЧС – чрезвычайная ситуация

PkID(Primary key Identifier, первичный ключ) – в реляционной модели данных один из потенциальных ключей отношения, выбранный в качестве основного ключа (или ключа по умолчанию).

UID(Unique identifier, уникальный идентификатор) – идентификатор, который используют для однозначного определения объекта, однако признак уникальности не заложен в ID.

FKID(Foreign key identifier, внешний ключ) – идентификатор, который применяется для принудительного установления связи между данными в двух таблицах с целью контроля данных, которые могут храниться в таблице внешнего ключа.

TreeView – элемент графического интерфейса для иерархического отображения информации. Представляет собой совокупность связанных отношениями структуры пиктограмм в иерархическом древе.

ВВЕДЕНИЕ

История контроля доступа берёт своё начало со времён до Нашей Эры. На протяжении всей истории человечества мы наблюдаем за тем, как в нем используются элементы безопасности. Механические деревянные замки были обнаружены на территории современного Ирака еще в 4000 году до н. э., а гробница фараона Тутанхамона была заперта с помощью веревочного узла. Исторически контроль доступа включал в себя нечто большее, чем просто замки. У стражников королевства были сложные смены, которые позволяли им находиться в определенных местах только в определенное время, чтобы обеспечить круглосуточную охрану. В то время такие устаревшие технологии, как рвы, разводные мосты и сторожевые башни были самыми современными инженерными решениями, используемыми для обеспечения защиты периметра и контролируемого доступа. В современном Мире одним из самых эффективных наследников способов контроля является СКУД.

Система контроля и управления доступом (СКУД) является программно-аппаратным комплексом, в основе которого лежит принцип автоматического определения и реализации прав доступа на охраняемом объекте.

В наше время контроль доступа является сложным процессом, применяющимся для защиты мест, имущества, данных или граждан от личных до огромных корпоративных масштабов. Современные достижения в области технологий позволили создать более надежные и эффективные способы управления и защиты, и современные решения по контролю доступа выходят далеко за рамки стандартных ключей и карт доступа.

Удаленное управление остается важнейшей задачей с начала 2020 года, позволяя предприятиям обеспечивать безопасность своих зданий, даже когда там никого нет, и прокладывая путь к продуктивным гибридным моделям работы. Удаленный доступ и управление безопасностью позволяют как корпоративным, так и небольшим организациям оставаться гибкими, позволяя командам выполнять повседневные задачи без необходимости физического

присутствия на объекте. Такие функции, как удаленное отпирание дверей, полезны для того, чтобы впустить в здание поставщиков, подрядчиков и сотрудников, которые забыли или потеряли свои учетные данные. Благодаря доступу к отчетам о деятельности в режиме реального времени удаленное управление также позволяет организациям гибко перестраиваться на ходу.

Цель настоящей работы – проектирование базы данных круизного лайнера и разработка приложения для СКУД карт доступа пассажиров. Для достижения поставленной цели необходимо решить *следующие задачи:*

- исследовать предметную область;
- спроектировать базу данных;
- создать базу данных;
- заполнить базу данных информацией.
- разработать интерфейс;
- реализовать функции приложения;

Структура и объем работы. Отчет состоит из введения, 4 разделов основной части, заключения, списка использованных источников, 2 приложений. Текст выпускной квалификационной работы равен 69 страницам.

Во введении сформулирована цель работы, поставлены задачи разработки, описана структура работы, приведено краткое содержание каждого из разделов.

В первом разделе рассмотрены назначение и важность роли СКУД в наши дни, способы управления доступом, элементы современных СКУД, проанализирована выборка заинтересованных лиц и перспектив развития подобных систем в будущем. Также, в этом разделе представляется сценарий проекта, на котором базируются бизнес-правила будущего программного продукта.

Во втором разделе сформулированы основные требования к разработке приложения. В нем содержатся цели и задачи проекта, а также требования пользователя к интерфейсу. В разделе описано полное строение ER-модели базы данных. Кроме того, данный раздел охватывает различные функцио-

нальные аспекты, обсуждает архитектуру системы, приводит примеры вариантов использования.

В третьем разделе объясняется выбор используемых технологий проектирования, таких как tkinter, customtkinter, SQLite и asyncio. Также в этом разделе осуществляется проектирование пользовательского интерфейса и предоставление описания классов системы.

В четвертом разделе данной работы представлен подробный список классов и их методов, которые были использованы в процессе разработки сайта. Кроме того, в этом разделе проводится детальное тестирование разработанного приложения, с целью проверки его функциональности и стабильности.

В заключении преподносятся основные выводы и результаты, полученные в ходе разработки проекта.

В приложении А представлен графический материал. В приложении Б представлены фрагменты исходного кода.

1 Анализ предметной области

1.1 Характеристика СКУД и её назначение

Сегодня под термином СКУД (Система контроля и управления доступом) понимается вид физической безопасности, который управляет точками входа в ваш бизнес или внутренние помещения здания. Простой пример СКУД – ворота, физически не допускающие неавторизованных пользователей и позволяющие войти только авторизованным пользователям.

Минимальная функциональность СКУД – проверка наличия кода в базе данных и, при положительном результате, выполнение разблокировки преграждающего устройства. Однако в настоящее время существует некий «джентльменский набор» выполняемых действий, которые должны быть реализованы в любой уважающей себя СКУД – это контроль кого, куда и когда пускать.

В некоторых случаях к этому набору добавляются функции запрета повторного прохода (запрет на повторный вход, если не было выхода) и запрет прохода во внутренние помещения без пересечения проходной (внешнего периметра).

В большинстве случаев профессиональные СКУД обеспечивают и иные режимы проходов, например, проход по двум ключам, шлюз, контрольный обход территории, дисциплина проходов во внутренние зоны.

Три лидера отрасли, каждый из которых разработал инновационные системы контроля доступа – Keyscan, HID и RBH Access Technologies разрабатывают передовые системы безопасности, которые являются универсальными, масштабируемыми и гибкими, оставаясь при этом высокофункциональными.

1.2 Ключевые элементы системы контроля доступа и принцип их работы

1.2.1 Учётные данные

Учетные данные контроля доступа используют RFID (радиочастотной идентификации) для передачи сигналов на панель контроля доступа. Каждая метка имеет уникальный зашифрованный идентификационный номер. Владелец системы может выдать всем сотрудникам метки одного типа, но при этом настроить одну метку на разрешение входа, а другую - на запрет входа в определенные зоны здания.

УД бывают следующих видов:

- Приложения для смартфонов
- Пароль или пин-код
- Биометрия
- Брелок или карта доступа

1.2.1.1 Приложения для смартфонов

Для контроля доступа с помощью мобильных устройств используются смартфоны, планшеты и носимые электронные устройства, которые служат удостоверением личности пользователя для входа в офис или другие деловые помещения. Поскольку большинство работодателей поощряют тенденцию Bring Your Own Device (BYOD), контроль доступа с помощью приложений стал одним из самых эффективных инструментов для обеспечения дополнительного уровня безопасности в любой организации. В настоящее время электронные устройства позволяют осуществлять биометрическую аутентификацию без необходимости инвестировать в дорогостоящие биометрические считыватели.

1.2.1.2 Пароль или ПИН-код

Преимущество парольных систем по сравнению с системами дискреционного контроля доступа на основе матрицы доступа заключается в том,

что в них нет объекта, связанного с монитором безопасности, который хранит информацию о контроле доступа к конкретным объектам. Кроме того, парольные системы обеспечивают безопасность даже в том случае, если посторонние лица имеют неограниченный или технически возможный доступ к носителям, на которых записаны и хранятся зашифрованные объекты. Эти преимущества парольных систем управления доступом делают их чрезвычайно распространенными в документальных информационных системах.

1.2.1.3 Биометрия

Объекты со строгими требованиями к соблюдению норм и правил, такие как больницы и производственные предприятия, требуют особенно надежных систем контроля доступа и безопасности. Биометрические технологии используют измерения тела и физические характеристики для поиска уникальных идентификационных признаков человека (обычно отпечатков пальцев, сканирования сетчатки глаза или лица), чтобы сделать идентификацию положительной.

1.2.1.4 Брелок или карта доступа

Системы контроля доступа на основе карт играют важную роль в защите вашей собственности. Системы доступа без ключа - это эффективный и доступный способ обеспечить безопасность людей, помещений и данных. Карточные системы подходят для объектов любого типа - охраняемого склада, общежития или коммерческого здания.

Базовая современная СКУД состоит из:

- Считывателя
- Контроллера
- Замка

1.2.2 Считыватель

Считыватель меток находится на одной или обеих сторонах двери - на одной стороне двери, если система контролирует только вход, или на обеих

сторонах, если система контроля доступа контролирует вход и выход. Считыватель содержит антенну, которая подключается к панели контроля доступа и получает от нее питание. Когда человек входит в здание со своей меткой контроля доступа, на антенну считывателя поступает его зашифрованный идентификационный номер.

1.2.3 Контроллер

Контроллер - это ядро системы. В нем хранится информация об авторизации, которая настраивается администратором системы. Контроллер получает зашифрованный номер метки от считывателя, расшифровывает его, затем сравнивает ID-номер с ID-номерами, которые были загружены в систему. Если номера совпадают, и пользователь имеет право доступа к двери, дверь разблокируется.

1.2.4 Замок

Панель управления доступом, или контроллер, управляет электрическим замком двери. Если пользователю разрешено войти в здание, дверь автоматически разблокируется и может быть открыта. Общая схема устройства системы изображена на рисунке 1.1.

1.3 Анализ категорий пользователей СКУД

Система контроля и управления доступом - неотъемлемая часть безопасности. Она обеспечивает дополнительный уровень защиты, позволяя контролировать и отслеживать тех, кто имеет доступ к объекту защиты.

СКУД применима в отраслях - в здравоохранении, корпоративном, образовательном и государственном секторах. Сегодня они применяются на различных объектах, таких как стационарные, долговременные и жилые учреждения, больницы, муниципалитеты, кондоминиумы, распределительные комплексы, малые и крупные предприятия, а также колледжи и университеты.

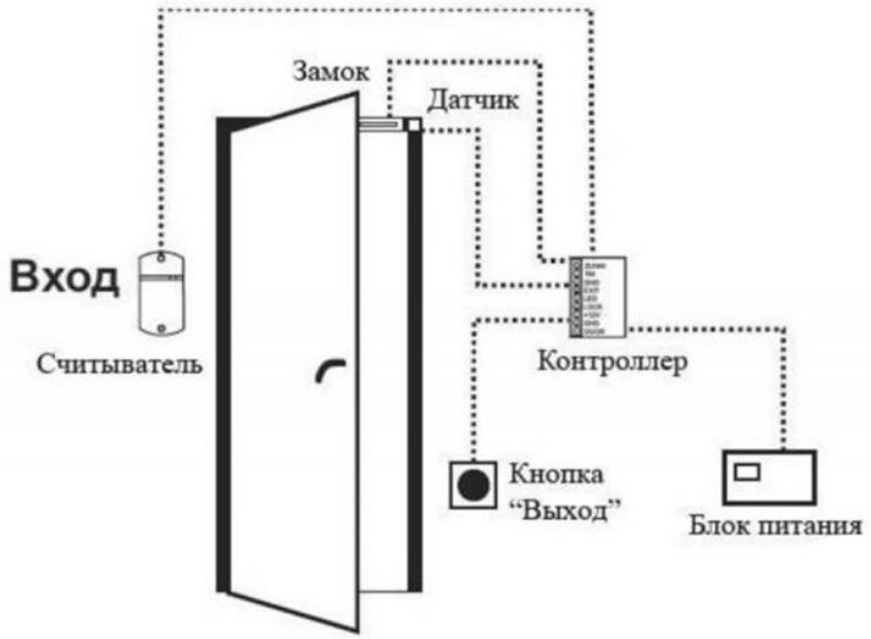

Рисунок 1.1 – Схема устройства современной СКУД

Среди аудитории пользователей программного обеспечения СКУД – системные администраторы и специалисты по информационной безопасности, нанятые организацией с целью контроля правил системы, устранения неполадок, проведения аудитов безопасности, поддержания работоспособности, внесения изменений.

В нашем случае, на круизном лайнере, аудиторией приложения является человек из экипажа, занимающийся внесением данных пассажиров в базу, наложением штрафов, созданием правил для входа в те или иные сектора, манипуляцией дверьми во время ЧС и т.д.

1.4 Перспективы развития

В перспективе развития СКУД на круизном лайнере ключевым является масштабирование её возможностей. Для самой системы это может быть увеличение быстродействия, расширение набора правил и ограничений, добавление умных систем обнаружения (камер) для фиксации нарушений и автоматического наложения штрафов, внедрение технической поддержки в реальном времени для нерядовых случаев.

В перспективе развития ПО для СКУД, ключевым является наиболее точное отражение функций системы для воздействия на них со стороны специалиста.

В будущем может быть рассмотрена интеграция back-end в виде набора новых бизнес-правил, дополнительного уровня защиты базы данных от атак злоумышленников, таких как SQL-инъекции, а front-end – как более широкую реализацию библиотек "tkinter "и "customtkinter ", что обеспечит более гибкое управление ресурсами и улучшит масштабируемость приложения.

Системы безопасности изменялись и дополнялись в течение многих лет, от веревок и деревянных замков до облачных систем и биометрии. Сейчас аутентификация работает повсеместно - от больших и надежных объектов до телефонов в карманах людей. При взгляде на будущие тенденции в области контроля доступа мы можем сделать вывод, что процессы инновации, интеграции и адаптивность необходимы, но не в ущерб безопасности. Будущее контроля доступа, в большей степени для предприятий, охватывает не только задачи защиты данных и ограничения доступа, а также позволяет делать процесс быстрее и с большей надежностью.

1.5 Особенности СКУД на пассажирском судне

Цель работы - специализированная программная и информационная поддержка СКУД круизного судна.

Актуальность данной работы заключается в необходимости установки порядка и правил на объекте с большим количеством людей, минимизации влияния ручного управления на систему, а также сокращении расходов на персонал, занимающий потенциальные места для продажи.

СКУД на пассажирском или круизном судне работает нестатично – решение о пропуске и отказе базируется на заданных заранее правилах доступа, соотвественно которым это решение может меняться на протяжении поездки. Основные правила выглядят следующим образом:

1. Каждый человек на борту имеет постоянный доступ к жизненно-необходимым дверям:(коридорные и лестничные, туалет, собственный но-

мер, столовая, выход на площадку, медпункт, детская комната. Однако, если помещение переполнено, действует режим ЧС(для площадок с открытым небом), пассажир пытается попасть в не соответствующий его полу туалет и т.д. – в этом случае доступ не предоставляется.

2. Предоставление доступа к некоторым комнатам действует по иерархической системе. Так, обладатели тарифа "Средний "к бару, аквапарку, кинотеатру, кальянной, бильярдной, банкетному залу, тиру. Однако, предоставлению доступа могут помешать иные ограничения, наложенные на комнату, в которую ведёт дверь. Это медицинские ограничения (аквапарк), ограничения по времени (банкетный зал), судимости (тир) и штрафы, накладываемые после проишествий.

3. Персонал имеет доступ ко всем дверям, кроме личных номеров пассажиров.

4. Пассажир может менять тариф в процессе путешествия. Это происходит в случае доплаты за услуги в процессе поедки (для повышения) и, в случае персонала, при отстранении от полномочий – тариф меняется на "Эконом ".

5. Штраф пассажира может быть снят, но только, если:

- Штраф относится к снимаемым

- Прошли 1 сутки с момента наложения и пассажир заплатил выкуп за снятие штрафа или обжаловал его.

6. Каждый человек на борту должен иметь свой пропуск, их количество для каждого не должно превышать 1.

7. Несовершеннолетние также имеют свой пропуск. Однако, для разблокировки пропуска за несовершеннолетним должен быть закреплён сопровождающий.

8. Ограничения пассажира могут быть как постоянными, так и наоборот. К ограничениям, что не изменяются и не оспариваются во время поездки относятся, например, Судимости и ограничения по здоровью.

9. У каждого пассажира должна быть своя комната. Пассажир может открыть только ту жилую комнату, что закреплена за ним.

10. В случае возникновения ЧС, СКУД начинает работать в другом режиме. Например, в случае шторма все двери наружу закрываются, двери лестниц и коридоров всегда открыты для исключения ситуации давки.

2 Техническое задание

2.1 Основание для разработки

Полное наименование системы: «Программное обеспечение для системы контроля и управления доступом на круизном судне». Основанием для разработки программы является приказ ректора ЮЗГУ от «15» апреля 2024 г. №1779-с «Об утверждении тем выпускных квалификационных работ».

2.2 Цель и назначение разработки

Целью данной работы является проектирование базы данных круизного лайнера и разработка приложения для карт доступа пассажиров.

В настоящее время перед СКУД стоят задачи: четкое и быстрое регулирование правил доступа во время поездки и обеспечение безопасной эвакуации пассажиров в случае ЧС. Для выполнения этих задач требуется систематизация данных.

Основными задачами при проектировании и разработке базы данных и приложения являются:

- исследование предметной области;
- проектирование базы данных;
- создание базы данных;
- заполнение базы данных информацией;
- разработка интерфейса;
- реализация приложения.

2.3 Требования к интерфейсу

Приложение должно включать в себя:

- навигацию по таблицам данных (Главное окно с возможностью выбрать таблицу по названию);
- страницу для работы с данными о пассажирах
- страницу для работы с данными о детях
- страницу для работы с данными о штрафах

- страницу для работы с данными о дверях
- страницу для работы с данными о комнатах
- страницу для отображения данных об имеющихся доступах всех пассажиров ко всем дверям с возможностью быстрой проверки наличия доступа у пассажира к двери в базе данных.
- отображение текущей таблицы данных в реальном времени(виджет TreeView для каждой таблицы данных);
- возможность так или иначе воздействовать на внесённые данные базы внутри приложения(напр. ”Изменить элемент ”Удалить элемент ”Добавить элемент”и т.д.);
- понятную навигацию и легкий поиск среди элементов конкретной таблицы (Возможность перелистывать элемент в текущем окне отображения, поиск по полю ID);
- отображение локального времени во избежание ошибок;

2.4 Требования к программной системе приложения «СКУД на круизном лайнере»

2.4.1 Требования к данным программной системы

Система должна уметь эффективно обрабатывать данные пассажиров, включая личную информацию, данные о тарифе, информацию о штрафах и спутниках. Необходимо обеспечить конфиденциальность и безопасность этих данных.

Для работы СКУД в системе должна быть представлена информация о следуюших обьектах:

- ДОСТУПЫ;
- ДОСТУПЫ-ЧС;
- ПАССАЖИРЫ;
- ДВЕРИ;
- КОМНАТЫ;
- ШТРАФЫ;

- ДЕТИ;

2.5 Построение ER-модели данных

Наборы объектов и связей в базе данных.

В приведенной на рисунке 2.1 ER-диаграмме представлены сущности и атрибуты, которые будут использоваться в программной системе

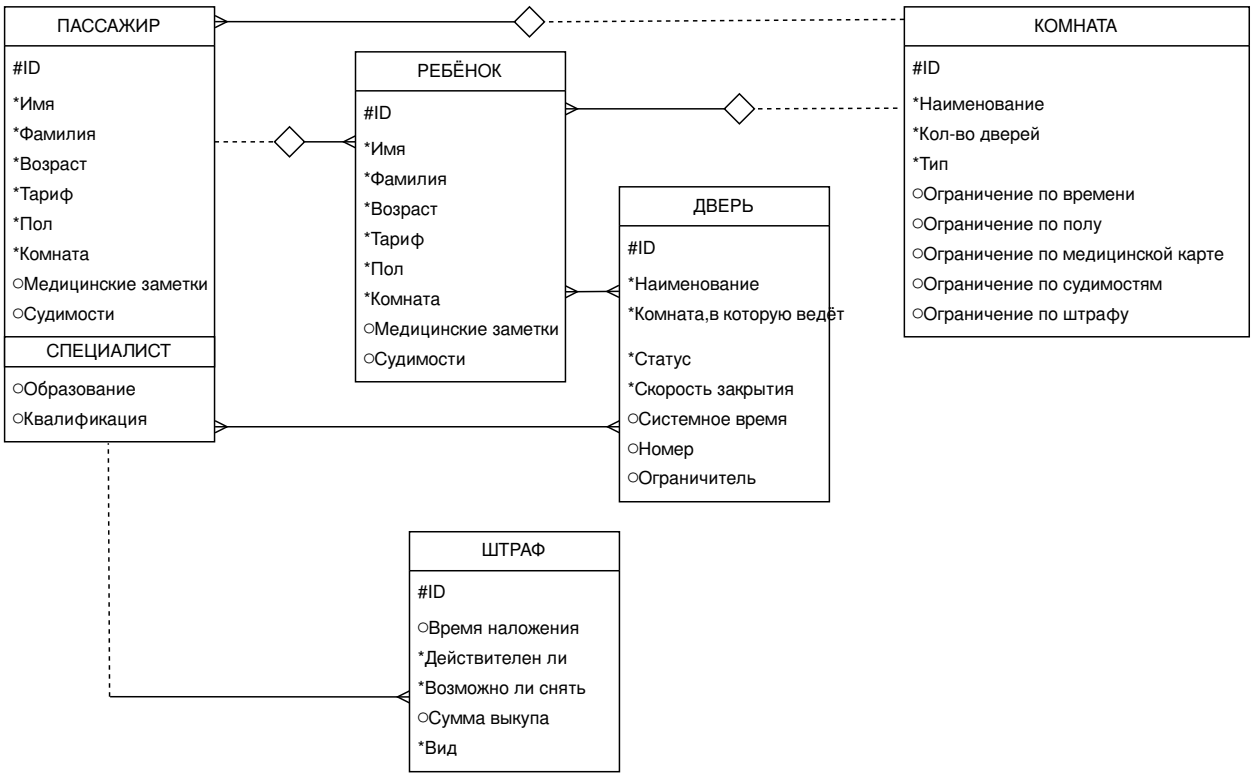

Рисунок 2.1 – ER-диаграмма

2.6 Функциональные требования к программной системе

На основании анализа предметной области, разрабатываемая программная система «СКУД на круизном лайнере» должна включать в себя следующие ключевые функциональные возможности:

- Открыть/закрыть таблицу
- Редактировать таблицу
- Перейти к следующему/предыдущему элементу
- Добавить новый элемент – реализация опции добавления нового элемента с последующей актуализацией этих данных в БД

- Редактировать выбранный элемент – реализация опции редактирования элемента с последующей актуализацией этих данных в БД

- Найти элемент по идентификатору

- Удалить элемент – реализация опции редактирования элемента с последующей актуализацией этих данных в БД

- Запросить проверку доступа – реализация опции запроса проверки пассажира на наличие доступа к двери

2.7 Моделирование вариантов использования

На рисунке 2.2 представлена диаграмма прецедентов, которая служит важным средством для систематизации функциональных требований и возможностей системы, выявляя ключевые сценарии использования. Диаграмма прецедентов изображает, как пользователи могут взаимодействовать с системой, а также помогает выявить набор основных функциональных возможностей, доступных пользователям.

При построении диаграммы вариантов использования применяется унифицированный язык визуального моделирования UML.

Каждый прецедент на диаграмме отражает отдельный путь взаимодействия в системе и представляет потенциальные действия, которые могут быть предприняты пользователями в различных ролях. Пользователем или действующим лицом является сущность, взаимодействующая с системой извне (например, человек, техническое устройство).

2.7.1 Вариант использования «Открыть/закрыть таблицу»

Пользователь выбирает интересующую его таблицу из списка предложенных и открывает окно, репрезентирующее требуемый от приложения интерфейс.

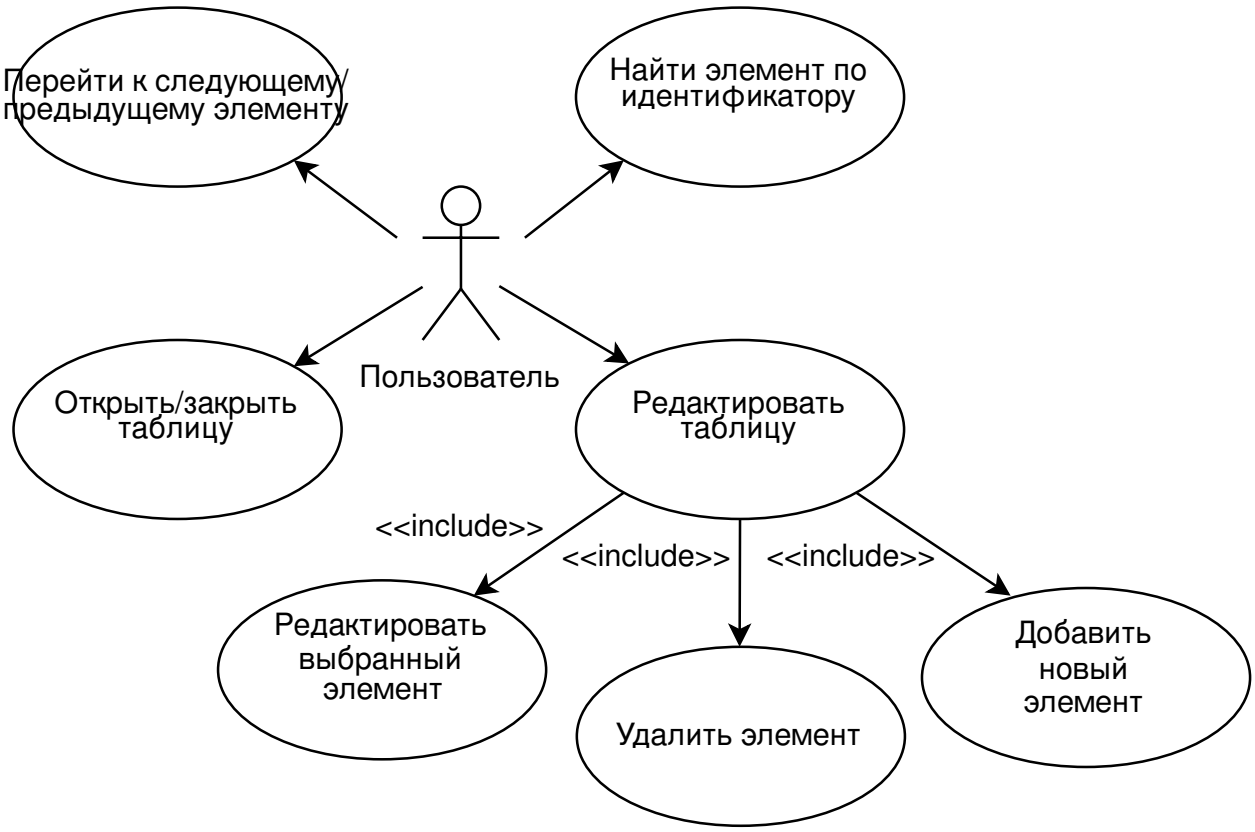

Рисунок 2.2 – Диаграмма прецедентов

2.7.2 Вариант использования «Редактировать таблицу»

Пользователь решает внести какие-либо изменения в таблице. Он может сделать это, добавляя новые элементы, удаляя или редактируя существующие.

2.7.3 Вариант использования «Перейти к следующему/предыдущему элементу»

Пользователь листает элементы в таблице посредством перехода к предыдущему/следующему за текущим.

2.7.4 Вариант использования «Найти элемент по идентификатору»

Пользователь решает не листать элементы по одиночке т.к. не хочет тратить на это время. Он помнит идентификатор искомого элемента, вводит его в предназначенное текстовое поле и переходит к искомому элементу.

2.7.5 Вариант использования «Добавить новый элемент»

Пользователь заполняет поля данных будущего объекта и добавляет его в таблицу по клику.

2.7.6 Вариант использования «Редактировать выбранный элемент»

Пользователь редактирует необходимые ему поля у текущего элемента и вносит эти изменения в таблицу по клику.

2.7.7 Вариант использования «Удалить элемент»

Пользователю больше не нужен выбранный элемент и он стирает его в таблице по клику.

2.8 Нефункциональные требования к программной системе

2.8.1 Требования к надежности

Для обеспечения надёжной работы базы данных, она должна быть приведена в нормализированный по трём формам вид: -Первая нормальная форма (1NF)

Все атрибуты объекта ПАССАЖИР имеют только одно значение.

Все атрибуты объекта РЕБЁНОК имеют только одно значение.

Все атрибуты объекта ДВЕРЬ имеют только одно значение.

Все атрибуты объекта КОМНАТА имеют только одно значение.

Все атрибуты объекта ШТРАФ имеют только одно значение.

Все атрибуты объекта пересечения ДОСТУПЫ имеют только одно значение.

Все атрибуты объекта пересечения ДОСТУПЫ ЧС имеют только одно значение.

-Вторая нормальная форма (2NF)

Атрибуты объекта ПАССАЖИР зависят от полного UID (id пассажира).

Атрибуты объекта РЕБЁНОК зависят от полного UID (id ребёнка).

Атрибуты объекта ДВЕРЬ зависят от полного UID (id двери).

Атрибуты объекта КОМНАТА зависят от полного UID (id Комнаты).

Атрибуты объекта ШТРАФ зависят от полного UID (id штрафа).

Атрибуты объекта пересечения ДОСТУПЫ зависят от двойного UID.

Атрибуты объекта пересечения ДОСТУПЫ ЧС зависят от двойного UID.

-Третья нормальная форма (3NF) При проверке всех объектов не было выявлено независимых от UID атрибутов или атрибутов, зависимых от других атрибутов.

На основе ER-модели данных построена реляционная модель данных, которой должна соответствовать БД. Она показанная на рисунке 2.3

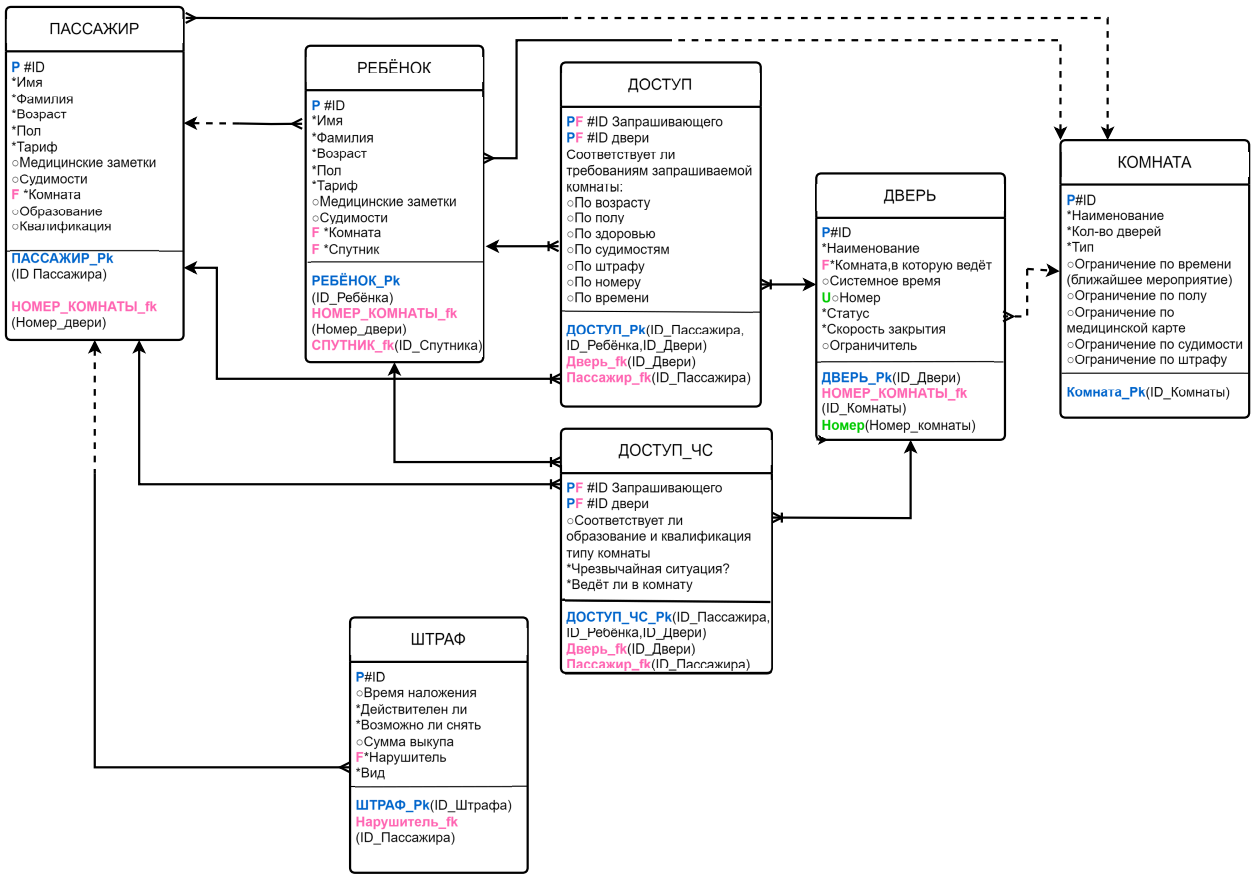

Рисунок 2.3 – Реляционная модель данных

2.8.2 Требования к целостности

1. Поддержка современных операционных систем, обеспечивающих стабильную и безопасную работу на различных платформах;

2. Реализация защиты от атак типа SQL-инъекций, предотвращающей нежелательное вмешательство в работу приложения;

3. Регулярное обновление компонентов системы для предотвращения уязвимостей, связанных с устаревшим программным обеспечением.

2.8.3 Требования к программному обеспечению

Для реализации программной системы необходимо подготовить следующие компоненты:

Интерпретатор языка программирования Python;

СУБД SQLite;

Для работы и приложения требуется ОС Windows 7 или более поздняя версия/Ubuntu 16.0 или более поздняя версия с установленным интерпретатором языка python и установленной СУБД SQLite.

2.8.4 Требования к аппаратному обеспечению

Системе необходим центральный процессор с количеством ядер от 2 и выше с частотой ядра от 1.4 ГГц. Объем оперативной памяти – 4 Гб. Свободное место на диске: 13МБ

2.9 Требования к оформлению документации

Требования к стадиям разработки программ и программной документации для вычислительных машин, комплексов и систем независимо от их назначения и области применения, этапам и содержанию работ устанавливаются ГОСТ 19.102–77. Программная документация должна включать в себя:

1. Анализ предметной области;

2. Техническое задание;

3. Технический проект;

4. Рабочий проект.

3 Технический проект

3.1 Общая характеристика организации решения задачи

Необходимо спроектировать и разработать приложение, которое обеспечит функционирование СКУД на круизном лайнере AIDABlu.

Приложение представляет собой панель управления для работы с данными в базе данных системы. Панель содержит текстовую и графическую информацию (TreeView).

Приложение является десктопным, т.е. располагается и запускается внутри лишь одной операционной системы. Каждое окно приложения (за исключением главного) – это панель управления для конкретной таблицы базы данных. Приложение разработано на языке Python v3.10. Управление данными БД реализовано с помощью библиотеки sqlite3 и SQL – языка структурированных запросов к базе данных.

3.2 Общие сведения о программно-информационной системе

Полное наименование системы: Программное обеспечение для системы контроля и управления доступом на круизном судне.

Краткое обозначение системы: "СКУД на круизном лайнере ".

Описание системы: "СКУД на круизном лайнере "предназначена для корпораций, организующих круизные путешествия, предоставляя им платформу для удобного контроля и управления данными всей бизнес-системы круизного лайнера на время поездки. Система создана для обеспечения комфортной и безопасной поездки каждого пассажира и функционирования в форс-мажорных ситуациях.

Условия эксплуатации: "СКУД на круизном лайнере "предназначена для использования как в нормальных, так и в чрезвычайных условиях работы.

Архитектура системы: Программное обеспечение основано на десктопной архитектуре, используя современные технологии разработки, вклю-

чая TKinter и CustomTkinter для Front-End части и sqlite3 для реализации запросов к БД в Back-End. Система использует базу данных SQLite.

Технологии и инструменты: В разработке использовались tkinter, customtkinter, sqlite3, asyncio, time

3.3 Обоснование выбора технологии проектирования

3.3.1 Описание используемых технологий и языков программирования

В процессе разработки приложения используются программные средства и языки программирования. Каждое программное средство и каждый язык программирования применяется для круга задач, при решении которых они необходимы.

3.3.2 tkinter

Выбор tkinter для Front-End обосновывается его простотой и кроссплатформенной поддержкой, а также минимальными требованиями к интерфейсу в пользу быстродействия.

Пакет tkinter («интерфейс Tk») - это интерфейс Python для создания GUI. Tkinter доступен на большинстве платформ Unix, включая macOS, а также на системах Windows.

Tkinter входит в состав большинства инсталляций Python, что делает его легкодоступным для разработчиков, которые хотят создавать приложения с графическим интерфейсом, не требуя дополнительных инсталляций или библиотек.

3.3.3 customtkinter

Выбор customtkinter для Front-End обосновывается его расширенным функционалом по сравнению с tkinter. Customtkinter предлагает большой список параметров для настройки виджетов и является полностью совместимым с элементами tkinter.

CustomTkinter - это библиотека пользовательского интерфейса для настольных компьютеров на основе Tkinter, которая обеспечивает современный вид и полностью настраиваемые виджеты.

3.3.4 SQLite

Выбор SQLite в качестве системы управления базами данных также обосновывается его удобством как на этапе проектирования, так и на этапе реализации.

SQLite - это библиотека на языке C, которая предоставляет легкую дисковую базу данных, не требующую отдельного серверного процесса и позволяющую обращаться к базе данных с помощью нестандартного варианта языка запросов SQL. Некоторые приложения могут использовать SQLite для внутреннего хранения данных. Также можно создать прототип приложения с использованием SQLite, а затем перенести код на более крупную базу данных, такую как PostgreSQL или Oracle.

3.3.5 asyncio и time

Модули asyncio и time были использованы для актуализации работы системы в реальном времени и многопоточном режиме.

3.3.6 Язык структурированных запросов к базе данных SQL

SQL - это стандартизированный язык программирования, который используется для управления реляционными базами данных и выполнения различных операций над данными в них.

SQL используется для следующего:

- изменение структуры таблиц данных и индексов базы данных;
- добавление, обновление и удаление строк данных;
- извлечение подмножеств информации из реляционных систем управления базами данных (РСУБД).

3.3.7 Язык программирования Python

3.3.7.1 Достоинства языка Python

Python - очень продуктивный язык. Благодаря простоте Python разработчики могут сосредоточиться на решении проблемы. Написание кода экономит время и освобождает его для более ёмкой работы с другими составляющими проекта.

Python поставляется под лицензией OSI с открытым исходным кодом. Это делает его свободным для использования и распространения. Можно загружать исходный код, изменять его и даже распространять свою версию Python. Это полезно для организаций, которые хотят изменить некоторые специфические функции и использовать свою версию для разработки.

Стандартная библиотека Python огромна, в ней можно найти практически все функции, необходимые для решения любой задачи. Таким образом, не придется зависеть от внешних библиотек.

Во многих языках, таких как C/C++, для запуска программы на разных платформах необходимо изменять код. С Python дело обстоит иначе. Вы пишете один раз и запускаете программу в любом месте.

3.3.7.2 Недостатки языка Python

Язык программирования Python использует большой объем памяти. Это может быть недостатком при создании приложений, когда предпочтение отдаётся оптимизации памяти.

Python используется для программирования на стороне сервера. Пользователь не видит Python на стороне клиента или в мобильных приложениях.

Python - динамически типизированный язык, поэтому тип данных переменной может измениться в любой момент. Переменная, содержащая целое число, в будущем может стать строкой, что может привести к ошибкам времени выполнения.

3.4 Проектирование пользовательского интерфейса

На основании требований к пользовательскому интерфейсу, представленных в пункте 2.3 технического задания, был разработан графический интерфейс десктопного приложения с применением python tkinter, customtkinter и SQLite. Этот процесс подчеркивает важность интуитивно понятного и эффективного взаимодействия с пользователем. Разработанный интерфейс ориентирован на обеспечение легкости в использовании и интуитивного понимания функционала приложения, предоставляя пользователю простое и эффективное взаимодействие с приложением.

1. **Навигация по таблицам с данными:** Реализация функции навигации на основе полей psr_btn, psr_ua_btn, drs_btn, rms_btn, pns_btn, acs_btn;

2. **Панель управления для каждой из таблиц с данными:** диверсификация интерфейсов происходит на основе метода show_table класса Controller;

3. **Отображение текущей таблицы с данными в реальном времени:** актуальность этого графического элемента (дерева) поддерживается за счёт методов TreeRefresh и TreeCreate класса Controller;

4. **Воздействие на внесённые данные базы внутри приложения:** Добавление, изменение или удаление элементов реализованы в методах add_element, update_element и delete_element класса Tables соответственно, внутри которых формирование строк запросов к БД реализовано с помощью класса Utils.

5. **Навигация и лёгкий поиск среди элементов среди данных:** Пользователь может переходить по элементам последовательно (метод MoveTo класса Controller), или же использовать отдельное текстовое поле для метода search_element класса Tables, перейдя сразу к искомому элементу таблицы.

6. **Отображение локального времени:** Учитывая специфику проекта – систему для круизного лайнера, учтено, что часовой пояс может измениться во время путешествия. На уровне python и SQL была установлена привязка к локальному времени (localtime), а функционал часов запущен в асинхронном потоке во избежание ошибок и помех работы основной системы.

7. **Отображение информации о доступе выбранного пассажира к выбранной двери:** На основе правил, прописанных в полях Комнаты, к которой привязана выбранная Дверь, в отдельном окне отображается информация о том, есть ли у Пассажира доступ к этой Двери.

Рисунок 3.1 – модели интерфейса «Панель управления для каждой из таблиц с данными» и «Отображение текущей таблицы с данными в реальном времени»

Процесс изменения данных максимально упрощён и включает следующие шаги:

1. В главном меню пользователь выбирает поле с названием необходимого ему вида данных.

2. В появившемся окне пользователь находит нужный ему элемент посредством «Пролистывания» или посредством поиска по идентификатору, при нажатии на соответствующие подписанные поля.

3. Пользователь изменяет некоторые данные в полях элемента и выбирает опцию «Изменить», если ему нужно было редактировать элемент, или «Удалить», если была необходимость удалить элемент.

4. Если пользователю необходимо добавить элемент, он должен заполнить поля ввода: Наименование, принадлежность, номер, статус, скорость

открытия, вместимость (В случае работы с данными о дверях) и выбрать опцию «Добавить».

5. На последнем этапе, если все обязательные поля были заполнены данными корректного формата, пользователь получает сообщение об успешном проведении операции и она выполняется.

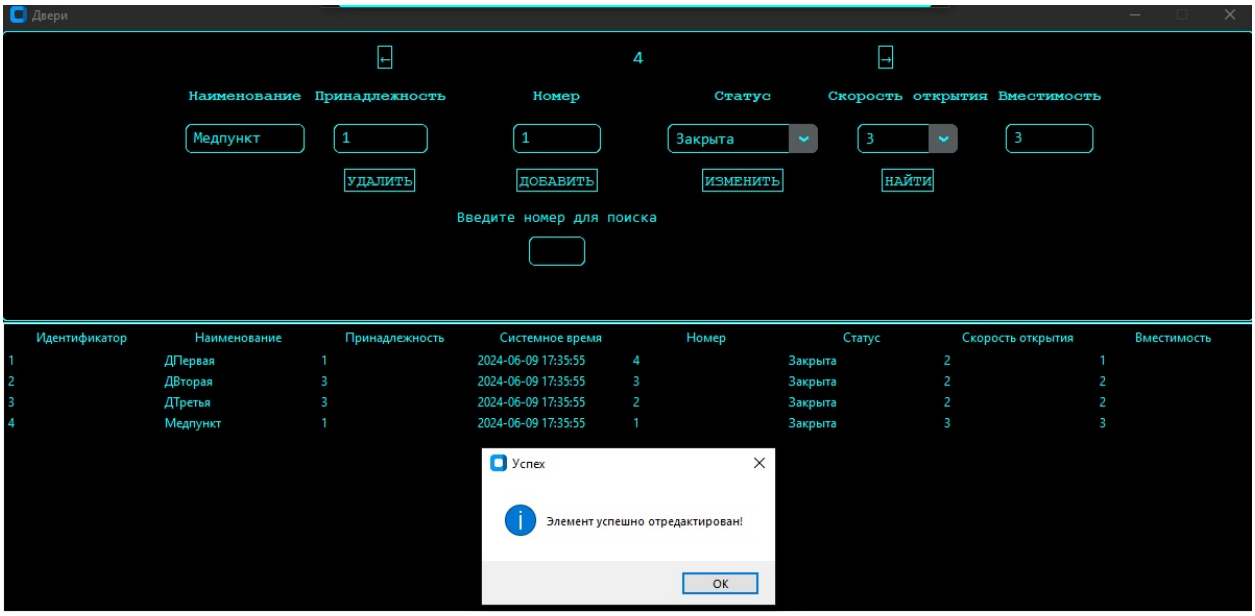

Рисунок 3.2 – модель интерфейса изменения данных элемента в данных о дверях

Процесс получения информации о доступе является еще более простым и состоит из следующих этапов:

1. В главном меню пользователь выбирает опцию «Проверка доступа».

2. В появившемся окне пользователь с помощью переключателя выбирает нужную ему группу поиска среди пассажиров - таблицу «Пассажиры» или «Дети».

3. Пользователь вводит идентификаторы пассажира и двери в поля "Пассажир "и "Дверь "соответственно и выбирает опцию «Проверить доступ»

4. На последнем этапе, если все обязательные поля были заполнены данными корректного формата и в БД содержатся данные о предоставленном/отказанном доступе, пользователь получает сообщение о том, есть ли у указанного пассажира доступ к указанной двери.

Рисунок 3.3 – модель интерфейса «отображение информации о доступе выбранного пассажира к выбранной двери».

3.5 Диаграмма размещения

Диаграмма размещения, отображаемая на рисунке 3.4, является фундаментальным инструментом для иллюстрации взаимосвязей между программными и аппаратными компонентами системы. Этот элемент визуализации служит для акцентирования значимости стратегического планирования в процессе разработки распределенных систем. Детальное и глубокое понимание этих взаимосвязей критически важно для успешного создания и функционирования распределенных информационных систем.

Каждый компонент системы, будь то программный или аппаратный, играет важную роль в обеспечении её общей эффективности и надежности. Подход, основанный на стратегическом планировании, способствует оптимизации этих взаимодействий и повышает вероятность успешной реализации и эксплуатации системы в целом.

Она является хорошим средством для показа маршрутов перемещения объектов и компонентов в распределенной системе.

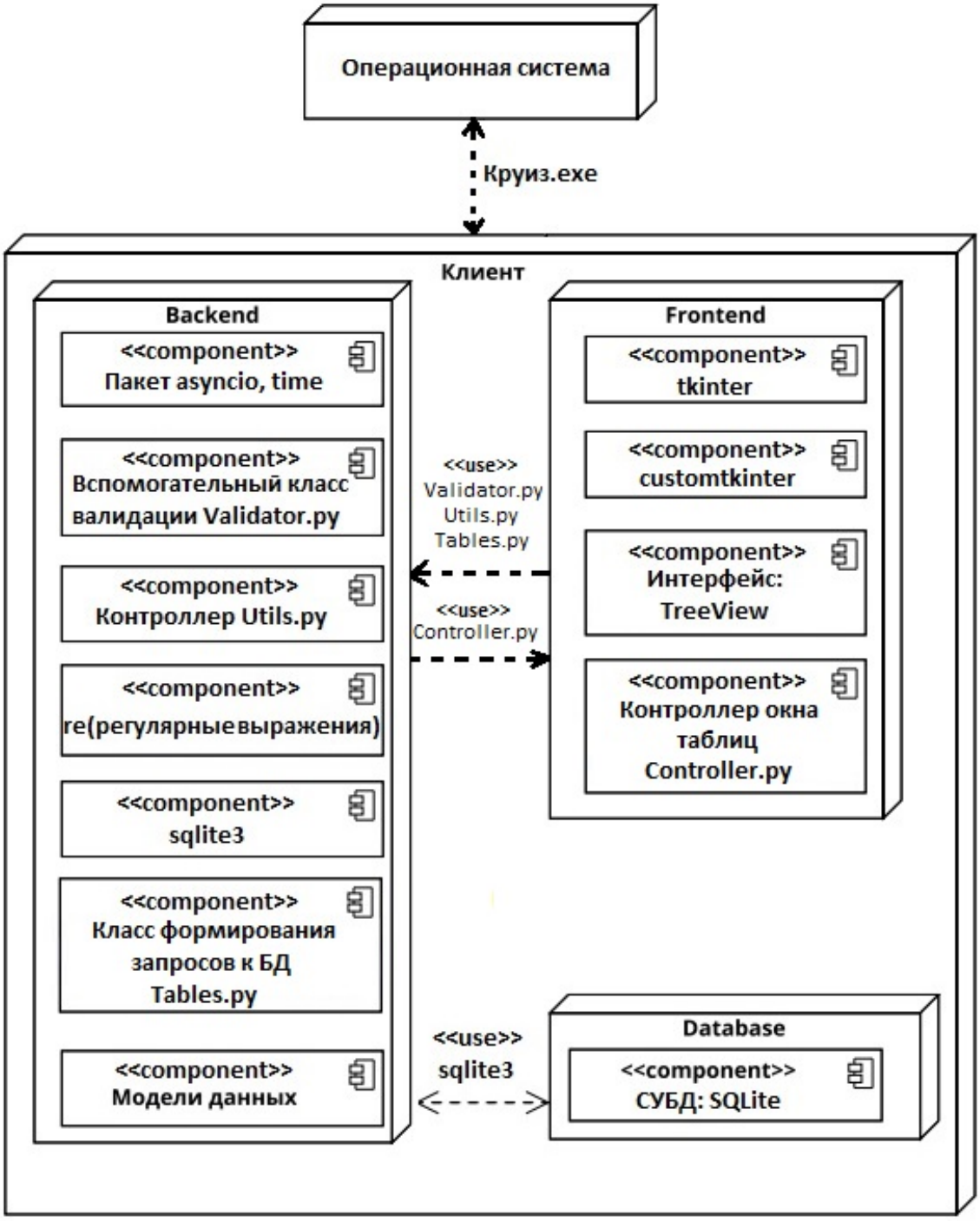

Рисунок 3.4 – Диаграмма размещения

3.6 Описание архитектуры приложения

Архитектура приложения, реализованная в рамках текущей работы, базируется на модели MVC (Model-View-Controller). Этот паттерн был избран из-за его способности к эффективному распределению функциональных обя-

занностей между структурными компонентами системы, а также способствованию упрощению процессов разработки и тестирования.

MVC реализован следующим образом:

1. Модель (Model) и Контроллер (Controller): Эти компоненты обеспечивают управление бизнес-логикой и обработку данных, а также связь между пользовательским интерфейсом и базой данных;

2. Представление (View): Реализовано через фронтенд, использующий пакеты tkinter, customtkinter и treeview, и отвечающий за визуализацию информации и интерактивное взаимодействие с пользователем.

Ключевым аспектом в управлении данными является использование системы управления базами данных SQLite. Этот выбор был обусловлен высокой надежностью SQLite, её устойчивостью к атакам типа "SQL-Инъекция "и гибкостью в обработке сложных запросов, что имеет критическое значение для эффективного функционирования бэкенда.

Применение архитектуры MVC принесло следующие ключевые преимущества:

- Четкое Разделение Обязанностей: Эффективное разграничение между пользовательским интерфейсом (класс main.py и controller.py) и back-end логикой (tables.py) значительно упрощает процесс разработки и последующей поддержки системы

- Гибкость и Масштабируемость: Благодаря MVC, архитектура приложения легко адаптируется и масштабируется, что позволяет разработчикам модифицировать или расширять отдельные части системы без влияния на другие;

- Упрощение Тестирования: Независимость компонентов архитектуры MVC облегчает процедуру тестирования, позволяя проводить её для каждого элемента в отдельности

Такой подход устанавливает предпочтительную форму записи организации функционала и способ его использования в тестировании. Он обеспечивает создание надежной, гибкой и масштабируемой системы, а также применим как в крупных, так и в малых проектах.

Модули:

В backend применяется модуль asyncio для организации работы системного времени в отдельном потоке.

Модуль re включается для работы с регулярными выражениями для задания правил написания, валидации полей.

Модуль sqlite3 используется для организации подключения и запросов к базе данных.

Для создания frontend используются модули tkinter, customtkinter и treeView. TreeView необходим для отображения виджета древа таблицы в реальном времени, customtkinter - для основного интерфейса приложения, а tkinter - для служебного упорядочивания элементов интерфейса customtkinter и treeview.

Классы:

main.py - в этом классе прописан front-end главного меню с последующей передачей вводимых данных в другие компоненты.

Utils.py - служебный класс, внутри которого реализованы функции запросов к базе данных, асинхронное обновление системного времени(Например, подключение к БД - create_connection(), чтение - read_single_row(), запрос - execute_query() и т.д.).

Validator.py - служебный класс для задания правил написания (валидации). Необходим для проверки корректности вводимых данных(Например, метод FKValid() - для валидации внешних ключей, overvalidation() для дополнительной проверки корректности перед валидацией)

Controller.py - в этом классе прописана front-end логика панели управления таблицами данных с последующей передачей вводимых данных в другие компоненты, а также выводом полученных данных на экран.

Tables.py - служебный класс, созданный для формирования запросов к БД. Вся логика взаимодействия с данными внутри базы, таким как добавление - add_element(), удаление - delete_element() и т.д., содержится в этом классе.

4 Рабочий проект

4.1 Описание классов системы

- "main "является фундаментальным классом для главного меню. Помимо основных свойств, таких как connection, экземпляров utils, controller и tables, содержит в себе стартовые виджеты.

- "Controller "– расширяет "main ", представляя панель управления таблицами, необходимую для взаимодействия пользователя с данными внутри таблицы.

- "Utils "применяется для отправки запросов в БД. Содержит только самые необходимые методы, взаимодействующие с БД непосредственно.

- "Tables "служит для формирования необходимых сложных запросов к БД. Сформировав строку, класс использует "Utils "для отправки.

- "Validator "представляет собой служебный класс, конфигурирующий интерфейс таким образом, чтобы пользователь не мог ввести данные неподходящего формата.

Можно выделить следующий список классов, их полей методов, использованных при разработке приложения (таблица 4.1).

Таблица 4.1 – Описание классов, используемых в приложении

Название класса	Модуль, к которому относится класс	Описание поля/метода	Поле/метод
1	2	3	4
main	Главный модуль	isES – поле для определения режима работы программы (Обычный или ЧС)	bool isES
	Главный модуль	window – экземпляр класса customtkinter.CTk(), представляющий окно	CTk window

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	frame – экземпляр класса customtkinter.CTkFrame(), представляющий поле для виджетов	CTkFrame frame
	Главный модуль	connection – строка, содержащая путь для подключения к БД	string connection
	Главный модуль	ask_lb, timer_lb – виджеты текста в главном меню	CTkLabel ask_lb CTkLabel timer_lb
	Главный модуль	pns_btn, rms_btn, drs_btn, psr_btn, psr_ua_btn – виджеты кнопок в главном меню	CTkButton pns_btn CTkButton rms_btn CTkButton drs_btn CTkButton psr_btn CTkButton psr_ua_btn
	Главный модуль	on_closing – метод для вывода вопроса о том, уверен ли пользователь в своём решении выйти	on_closing() Возвращает: ничего
Controller	Главный модуль	styles_init – метод для инициализации стилей и добавления его в список ttk.element_names	styles_init(st). Принимает на вход экземпляр класса Style, который будет настраивать. Возвращает: ничего
	Главный модуль	show_table – метод для перехода к панели управления выбранной таблицей. Отображает, настраивает и группирует каждый элемент по сетке	show_table(which, window, tables, connection). Принимает на вход название таблицы, экземпляр окна, экземпляр tables, и строку подключения connection. Возвращает: ничего

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	fields_creation – метод, вызываемый в show_table и дополняющий его. Определяет и направленно настраивает поля ввода информации	fields_creation(which, frameEx, window). На вход принимает название таблицы, экземпляр поля для виджетов, экземпляр окна. Возвращает: tuple (validatedTFs, combolist) – кортеж сформированных полей ввода
	Главный модуль	clear – метод, очищающий поля ввода	clear(keyLabel, combinedControls, window). На вход принимает таблицу отображения Id текущей строки для очищения, кортеж полей ввода, экземпляр окна. Возвращает: ничего
	Главный модуль	TreeCreate – метод, отображающий текущую таблицу в виде виджета древа	TreeCreate(tree, table, tableWin, connection). На вход принимает экземпляр класса ttk.TreeView, который будет пересоздавать, название таблицы, экземпляр поля виджетов, строку подключения. Возвращает: TreeView.tree – созданный виджет древа

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	TreeRefresh – метод, обновляющий виджет древа	TreeRefresh(tree, table, connection). На вход принимает экземпляр класса ttk.TreeView, который будет обновлять, название таблицы, строку подключения. Возвращает: Ничего
	Главный модуль	MoveTo – метод для перехода просмотра к конкретному элементу таблицы по индентификатору	MoveTo(id, table, combinedControls, currentLb, window, connection). На вход принимает идентификатор элемента для перехода, имя таблицы, кортеж полей вывода, табличку отображения текущего Id, экземпляр окна и строку подключения. Возвращает: Ничего
	Главный модуль	show_acc – метод для перехода к окну проверки доступа	show_acc(connection, isES, tables, window). На вход принимает строку подключения к БД, флаг ЧС, экземпляр класса tables, экземпляр окна. Возвращает: Ничего

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	toggle_ES – метод обработки состояний переключателя, применяемый в определении одной из двух таблиц для показа	toggle_ES(switch, psrlb, tables,connection, tree, isES). На вход принимает экземпляр переключателя, текстовое поле для изменения выводимой информации, строку подключения, экземпляр виджета древа и флаг ЧС. Возвращает: Ничего
Utils	Главный модуль	create_connection – метод создаёт подключение к базе данных	create_connection(path). На вход принимает строковый путь к файлу базы данных. Возвращает: sqlite3.connection connection - экземпляр созданного подключения
	Главный модуль	read_single_row – считывает единственную строку с БД	read_single_row(id, connection, table). На вход принимает идентификатор, по которому будет считываться строка, экземпляр подключения, название таблицы Возвращает: строку в формате кортежа

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	execute_query – посылает команду в БД	execute_query(connection, query). На вход принимает экземпляр подключения и строку команды для выполнения. Возвращает: bool True/bool False – флаг о том, успешно ли выполнен запрос
	Главный модуль	execute_read_query – отдельный метод для послания запроса в БД на чтение	execute_read_query(connection, query). На вход принимает экземпляр подключения и строку команды для выполнения. Возвращает: строку в формате кортежа
	Главный модуль	timetick – асинхронный метод для отображения и вывода текущего времени в главном меню	timetick(timerLb). На вход принимает, куда будет выводиться время. Возвращает: ничего
	Главный модуль	asyncMLoop – метод для обеспечения асинхронной работы окна с часами	asyncMLoop(wndw). На вход принимает окно главного меню. Возвращает: ничего
	Главный модуль	asyncStart – асинхронный метод для настройки взаимодействия главного меню с часами в асинхронном порядке и обновления полей со временем в БД	asyncStart(window, timer_lb, connection). На вход принимает экземпляр окна, поле для вывода времени, строку подключения к БД. Возвращает: Ничего

Продолжение таблицы 4.1

1	2	3	4
Tables	Главный модуль	self_definition – метод переводит названия таблиц данных с русского языка на английский для последующих запросов к БД	self_definition(which). На вход принимает строку названия на русском языке. Возвращает: строку названия на английском языке
	Главный модуль	timeAppend – метод, переводящий строку времени из формата «ЧЧММСС» в формат «ЧЧ:ММ:СС»	timeAppend(timeString). На вход принимает неотформатированную строку времени. Возвращает: строку времени в отформатированном виде
	Главный модуль	penalty_check – метод, сравнивающий данные о штрафных ограничениях данных о Дверях и данных о Штрафах	penalty_check(pennies, type). На вход принимает кортеж из найденных по принадлежности штрафов и строку типа штрафа, с которой нужно сопоставить элементы кортежа. Возвращает: флаг о том, совпали ли типы
	Главный модуль	check_access – метод реализации опции «Проверка доступа», проверяет введённые данные на наличие в базе и выводит информацию о предоставлении доступа.	check_access(psr_Tf, dor_Tf, switch, connection). На вход принимает экземпляры текстовых полей куда были введены идентификаторы для поиска, жкземпляр переключателя, строку подключения к БД. Возвращает: ничего

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	start_accesses – метод заполнения данных о доступах пассажиров к дверям по каждому из критериев ограничений.	start_accesses(connection, psr_switch). На вход принимает строку подключения к БД и экземпляр переключателя. Возвращает: ничего
	Главный модуль	add_element – метод, реализующий опцию формирования запроса добавления элемента в БД.	add_element(table, combinedControls, currentLb, tree, tableWin, connection, window). На вход принимает строку названия таблицы, кортеж полей ввода, экземпляр текстового поля для отображения идентификатора текущего элемента, экземпляр виджета древа, экземпляр контейнера для виджетов, строку подключения, экземпляр окна. Возвращает: ничего

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	delete_element – метод, реализующий опцию формирования запроса удаления элемента в БД.	delete_element(table, key, currentLb, combinedControls, tree, connection, window). На вход принимает строку названия таблицы, идентификатор текущего элемента, кортеж полей ввода, экземпляр текстового поля для отображения идентификатора текущего элемента, экземпляр виджета древа, строку подключения, экземпляр окна. Возвращает: ничего
	Главный модуль	update_element – метод, реализующий опцию формирования запроса изменения элемента в БД.	update_element(table, combinedControls, key, tree, connection). На вход принимает строку названия таблицы, кортеж полей ввода, идентификатор текущего элемента, экземпляр виджета древа, строку подключения. Возвращает: ничего

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	search_element – метод, реализующий опцию поиска элемента в БД.	search_element(combinedControls, table, tf, currentLb, connection, window). На вход принимает кортеж полей ввода, строку названия таблицы, текстовое поля для воода идентификатора поиска, экземпляр текстового поля для отображения идентификатора текущего элемента, строку подключения. Возвращает: ничего
	Главный модуль	list_table – метод, реализующий опцию "пролистывания "текущей таблицы данных.	list_table(direction, table, connection, window). На вход принимает флаг направления (вперёд или назад), строку названия таблицы, строку подключения к БД, экземпляр окна. Возвращает: массив данных текущего элемента

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	configure_list – метод, дополняющий list_table и выводящий данные текущего элемента в поля ввода .	configure_list(direction, table, combinedControls, currentLb, connection, window). На вход принимает флаг направления (вперёд или назад), строку названия таблицы, кортеж полей ввода, экземпляр текстового поля для отображения идентификатора текущего элемента, строку подключения к БД, экземпляр окна. Возвращает: массив данных текущего элемента
Validator	Главный модуль	connection – строка для подключения к базе данных	string connection
	Главный модуль	validation_digits – метод, использующийся для регистрации правила валидации цифр	validation_digits(stringVal). На вход принимает строку для проверки соответствия правилу. Возвращает: ту часть строки, что соответствует маске цифр
	Главный модуль	validation_text – метод, использующийся для регистрации правила валидации символов Кириллицы	validation_text(stringVal). На вход принимает строку для проверки соответствия правилу. Возвращает: ту часть строки, что соответствует маске символов Кириллицы

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	validation_char – метод, использующийся для регистрации правила валидации единичного символа Кириллицы	validation_char(stringVal). На вход принимает строку для проверки соответствия правилу. Возвращает: ту часть строки, что соответствует маске единичного символа Кириллицы
	Главный модуль	validation_time – метод, использующийся для регистрации правила валидации текстового поля для ввода времени	validation_time(stringVal). На вход принимает строку для проверки соответствия правилу. Возвращает: ту часть строки, что соответствует маске формата времени
	Главный модуль	FKValid – метод, проверяющий наличие элемента с идентификатором для создания внешнего ключа	FKValid(number, tableName). На вход принимает номер для поиска, строку названия таблицы. Возвращает: флаг о том, был ли найден элемент
	Главный модуль	validate_single – метод, задающий правила ввода для единичного поля ввода	validate_single(element, window, type). На вход принимает экземпляр поля ввода, экземпляр окна, строку типа валидации. Возвращает: поле ввода с изменёнными правилами

Продолжение таблицы 4.1

1	2	3	4
	Главный модуль	validate_whole – метод, задающий правила ввода для массива полей ввода	validate_whole(tfS, which, window). На вход принимает массив полей ввода, строку названия таблицы данных, экземпляр окна. Возвращает: массив полей ввода с изменёнными правилами
	Главный модуль	overvalidation – метод, проверяющий уже введённые данные на корректность перед отправкой	overvalidation(table, combinedControls). На вход принимает строку названия таблицы, кортеж полей ввода. Возвращает: флаг соответствия корректности

4.2 Модульное тестирование разработанного web-сайта

Модульный тест для класса User из модели данных представлен на рисунке 4.1.

4.3 Системное тестирование разработанного web-сайта

```python
from django.test import TestCase
from .models import *
User = get_user_model()


class ShpoTestCases(TestCase):

    def setUp(self) -> None:
        self.user = User.objects.create(username='testtestovich', password='
            testtestovich', first_name='Sad', last_name='')

    def test_2(self):

        self.assertEqual(self.user.first_name, 'Sad')
        self.assertEqual(self.user.last_name, 'Cat')
        print((self.user))
        print((self.user.first_name))
        print((self.user.last_name))
```

Рисунок 4.1 – Модульный тест класса User

ЗАКЛЮЧЕНИЕ

Преимущества аддитивных технологий заключается в разнообразии процессов, позволяющих применять их в различных областях производства. Существенным ограничением же является и экономическая составляющая, которая не позволит внедрить аддитивное производство повсеместно.

Компании, видя, как развиваются информационные технологии, пытаются использовать их выгодно для своего бизнеса, запуская свой сайт для того, чтобы заявить о своем существовании, проинформировать потенциального клиента об услугах или продуктах, которые предоставляет. Для продвижения компании «Русатом – Аддитивные технологии» был разработан веб-сайт на основе системы «1С-Битрикс: Управление сайтом».

Основные результаты работы:

1. Проведен анализ предметной области. Выявлена необходимость использовать 1С-Битрикс.

2. Разработана концептуальная модель web-сайта. Разработана модель данных системы. Определены требования к системе.

3. Осуществлено проектирование web-сайта. Разработана архитектура серверной части. Разработан пользовательский интерфейс web-сайта.

4. Реализован и протестирован web-сайт. Проведено модульное и системное тестирование.

Все требования, объявленные в техническом задании, были полностью реализованы, все задачи, поставленные в начале разработки проекта, были также решены.

Готовый рабочий проект представлен адаптивной верстkой сайта. Сайт находится в публичном доступе, поскольку опубликован в сети Интернет.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Словарь данных и система управления базами данных / Data Dictionary and Database Management System, 2019. – Текст : электронный. URL: https://studfile.net/preview/942920/page:17/

2. Бойко И. Объектно-ориентированные СУБД / И. Бойко. – Киев : Высшая школа, 2014. –- 398 с. – Текст : непосредственный.

3. Рыкунов В. Охранные системы и технические средства физической защиты. / Security Focus 2022. –- 398 с. – ISBN: 978-5-9901176-3-1. – Текст : непосредственный.

4. Гринченко, Н.Н. Проектирование баз данных. СУБД Microsoft Access: Учебное пособие для вузов. / Н.Н. Гринченко и др. - М.: РиС, 2013. – 240 с. – ISBN: 978-5-9912-0295-4. Текст : непосредственный.

5. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. / Т. Коннолли. - М.: Вильямс И.Д., 2017. - 1440 с. – ISBN: 978-5-8459-2020-1. Текст : непосредственный.

6. scDataCom [Электронный ресурс] From Keys to Credentials: The History of Access Control, 2024. – Текст : электронный. URL: https://www.scdatacom.net/blog/from-keys-to-credentials-the-history-of-access-controlnbsp

7. Лукин, В.Н. Введение в проектирование баз данных. / В.Н. Лукин. - М.: Вузовская книга, 2015. – 144 с. – ISBN: 978-5-9502-0761-7. Текст : непосредственный.

8. Medium.com. / «SQLite – Как организовывать таблицы» Автор:А.Шагин, 2020. – Текст : электронный. URL: https://medium.com/nuances-of-programming/sqlite-как-организовывать-таблицы-81cce38af5b2

9. Мюллер, Р.Д. Проектирование баз данных и UML. / Р.Д. Мюллер; Пер. с англ. Е.Н. Молодцова. - М.: Лори, 2013. – 420 с. ISBN: 978-5-85-582322-6. Текст : непосредственный.

10. kisi [Электронный ресурс] Mobile access control guide: 2024. – Текст : электронный. URL: https://www.getkisi.com/guides/mobile-access-control-guide

11. Васильев А.Н. Программирование на Python в примерах и задачах/ Бомбора 2023. – 616с. –ISBN 978-5-04-103199-2 Текст : непосредственный.

12. Свейгард Э. Python. Чистый код для продолжающих / Питер, 2023 – 384 с. –- ISBN 978-5-4461-1852-6. -– Текст : непосредственный.

13. Richmond Security / Locked. Secured. Protected/ Learning About Access Control Systems: Control Cards, 2024. – Текст : электронный. URL: https://www.richmondsecurity.com/learning-about-access-control-systems-control-cards/

14. Вайсфельд М. Объектно-ориентированный подход. 5-е межд. изд. / Питер, 2024 –- 256 с. –- ISBN 978-5-4461-1431-3. – Текст : непосредственный.

15. Британская ассоциация индустрии безопасности. Руководство по составлению спецификаций на СКУД / Второе переиздание,тираж 500 2014. – 170 с. Текст : непосредственный.

16. aatel [Электронный ресурс] How do Access Control Systems work?: 2024. – Текст : электронный. URL: https://www.aatel.com/portfolio/how-do-access-control-systems-work/

17. hidglobal [Электронный ресурс] Chronicling the Evolution of Access Control Credentials: Jim Dearing, 2021. – Текст : электронный. URL: https://blog.hidglobal.com/2021/03/chronicling-evolution-access-control-credentials

18. Ворона В.А.,Тихонов В.А. Системы контроля и управления доступом. / Горячая Линия-Телеком 2018 г. –- 272 с. ISBN:978-5-9912-0059-2. Текст : непосредственный.

19. python.org / Documentation: tkinter / Python interface to Tcl/Tk, 2023. – Текст : электронный. URL: https://docs.python.org/3/library/tkinter.html

20. Фаулер М. Asyncio и конкурентное программирование на Python / ДМК Пресс, 2023 –- 398 с. –- ISBN 978-5-93700-166-5. – Текст : непосредственный.

ПРИЛОЖЕНИЕ А

Представление графического материала

Графический материал, выполненный на отдельных листах, изображен на рисунках А.1–А.0.

ПРИЛОЖЕНИЕ Б

Фрагменты исходного кода программы

main.tex

```
\input{setup.tex}

% Режим шаблона (должен быть включен один из трех)
\ВКРtrue
%\Практикаtrue
%\Курсоваятrue

\newcommand{\Дисциплина}{<<Проектирование и архитектура программных систем>>} % для курсовой
\newcommand{\КодСпециальности}{09.03.04} % Курсовая
\newcommand{\Специальность}{Программная инженерия} % Курсовая
\newcommand{\Тема}{Программное обеспечение для системы контроля и} % ВКР Курсовая
\newcommand{\ТемаВтораяСтрока}{управления доступом на круизном судне}
\newcommand{\ГдеПроводитсяПрактика}{Юго-Западном государственном университете} % для практики
\newcommand{\РуководительПрактПредпр}{} % для практики
\newcommand{\ДолжнРуководительПрактПредпр}{директор} % для практики
\newcommand{\РуководительПрактУнивер}{Чаплыгин А. А.} % для практики
\newcommand{\ДолжнРуководительПрактУнивер}{к.т.н. доцент} % для практики
\newcommand{\Автор}{Д.Д.Дараган}
\newcommand{\АвторРод}{Дарагана Д.Д.}
\newcommand{\АвторПолностьюРод}{Дарагана Даниила Дмитриевича} % для практики
\newcommand{\Шифр}{20-06-0258}
\newcommand{\Курс}{4} % для практики
\newcommand{\Группа}{ПО-026}
\newcommand{\Руководитель}{Е. И. Аникина} % для ВКР и курсовой
\newcommand{\Нормоконтроль}{А. А. Чаплыгин} % для ВКР
\newcommand{\ЗавКаф}{А. В. Малышев} % для ВКР
\newcommand{\ДатаПриказа}{«15» апреля 2024~г.} % для ВКР
\newcommand{\НомерПриказа}{1616-с} % для ВКР
\newcommand{\СрокПредоставления}{«11» июня 2024~г.} % для ВКР, курсового

\begin{document}
\maketitle
\ifПрактика{}\else{
    \input{ЛистЗадания}
    \input{Реферат}}\fi
\tableofcontents
\input{Обозначения}
\ifПрактика{}\else{\input{Введение}}\fi
\input{Анализ}
\input{ТехЗадание}
\input{ТехПроект}
\ifПрактика{}\else{
    \input{РабочийПроект}
    \input{Заключение}
}\fi
\input{СписокИсточников}
```

```
\ifВКР{\input{Плакаты}}\fi
\ifПрактика{}\else{\input{Код}}\fi
\end{document}
```

ТехПроект.tex

```
\section{Технический проект}
\subsection{Общая характеристика организации решения задачи}

Необходимо спроектировать и разработать приложение, которое обеспечит функционирование СКУД на круизном лайнере AIDABlu.

Приложение представляет собой панель управления для работы с данными в базе данных системы. Панель содержит текстовую и графическую информацию (TreeView).

Приложение является десктопным, т.е. располагается и запускается внутри лишь одной операционной системы. Каждое окно приложения (за исключением главного) – это панель управления для конкретной таблицы базы данных. Приложение разработано на языке Python v3.10. Управление данными БД реализовано с помощью библиотеки sqlite3 и SQL -- языка структурированных запросов к базе данных.

\subsection{Общие сведения о программно-информационной системе}

Полное наименование системы: Программное обеспечение для системы контроля и управления доступом на круизном судне.

Краткое обозначение системы: \textquotedbl СКУД на круизном лайнере \textquotedbl.

Описание системы: \textquotedbl СКУД на круизном лайнере \textquotedbl предназначена для корпораций, организующих круизные путешествия, предоставляя им платформу для удобного контроля и управления данными всей бизнес-системы круизного лайнера на время поездки. Система создана для обеспечения комфортной и безопасной поездки каждого пассажира и функционирования в форс-мажорных ситуациях.

Условия эксплуатации: \textquotedbl СКУД на круизном лайнере \textquotedbl предназначена для использования как в нормальных, так и в чрезвычайных условиях работы.

Архитектура системы: Программное обеспечение основано на десктопной архитектуре, используя современные технологии разработки, включая TKinter и CustomTkinter для Front-End части и sqlite3 для реализации запросов к БД в Back-End. Система использует базу данных SQLite.

Технологии и инструменты: В разработке использовались tkinter, customtkinter, sqlite3, asyncio, time

\subsection{Обоснование выбора технологии проектирования}

\subsubsection{Описание используемых технологий и языков программирования}

```

```
28 В процессе разработки приложения используются программные средства и языки
     программирования. Каждое программное средство и каждый язык
     программирования применяется для круга задач, при решении которых они
     необходимы.
29
30 \subsubsection {tkinter}
31 Выбор tkinter для Front-End обосновывается его простотой и кроссплатформенной
     поддержкой, а также минимальными требованиями к интерфейсу в пользу
     быстродействия.
32
33 Пакет tkinter («интерфейс Tk») - это интерфейс Python для создания GUI.
     Tkinter доступен на большинстве платформ Unix, включая macOS, а также на
     системах Windows.
34
35 Tkinter входит в состав большинства инсталляций Python, что делает его
     легкодоступным для разработчиков, которые хотят создавать приложения с
     графическим интерфейсом, не требуя дополнительных инсталляций или
     библиотек.
36
37 \subsubsection {customtkinter}
38 Выбор customtkinter для Front-End обосновывается его расширенным функционалом
     по сравнению с tkinter. Customtkinter предлагает большой список
     параметров для настройки виджетов и является полностью совместимым с
     элементами tkinter.
39
40 CustomTkinter - это библиотека пользовательского интерфейса для настольных
     компьютеров на основе Tkinter, которая обеспечивает современный вид и
     полностью настраиваемые виджеты.
41
42 \subsubsection {SQLite}
43 Выбор SQLite в качестве системы управления базами данных также обосновывается
     его удобством как на этапе проектирования, так и на этапе реализации.
44
45 SQLite - это библиотека на языке C, которая предоставляет легкую дисковую
     базу данных, не требующую отдельного серверного процесса и позволяющую
     обращаться к базе данных с помощью нестандартного варианта языка запросов
     SQL. Некоторые приложения могут использовать SQLite для внутреннего
     хранения данных. Также можно создать прототип приложения с использованием
     SQLite, а затем перенести код на более крупную базу данных, такую как
     PostgreSQL или Oracle.
46
47 \subsubsection {asyncio и time}
48 Модули asyncio и time были использованы для актуализации работы системы в
     реальном времени и многопоточном режиме.
49
50
51 \subsubsection{Язык структурированных запросов к базе данных SQL}
52 SQL - это стандартизированный язык программирования, который используется для
     управления реляционными базами данных и выполнения различных операций над
     данными в них.
53
54 SQL используется для следующего:
55 \begin{itemize}
56   \item изменение структуры таблиц данных и индексов базы данных;
```

```
57   \item добавление, обновление и удаление строк данных;
58   \item извлечение подмножеств информации из реляционных систем управления
        базами данных (РСУБД).
59 \end{itemize}
60
61
62 \subsubsection{Язык программирования Python}
63
64 \paragraph{Достоинства языка Python}
65 Python - очень продуктивный язык. Благодаря простоте Python разработчики
       могут сосредоточиться на решении проблемы. Написание кода экономит время и
        освобождает его для более ёмкой работы с другими составляющими проекта.
66
67 Python поставляется под лицензией OSI с открытым исходным кодом. Это делает
       его свободным для использования и распространения. Можно загружать
       исходный код, изменять его и даже распространять свою версию Python. Это
       полезно для организаций, которые хотят изменить некоторые специфические
       функции и использовать свою версию для разработки.
68
69 Стандартная библиотека Python огромна, в ней можно найти практически все
       функции, необходимые для решения любой задачи. Таким образом, не придется
       зависеть от внешних библиотек.
70
71 Во многих языках, таких как C/C++, для запуска программы на разных платформах
        необходимо изменять код. С Python дело обстоит иначе. Вы пишете один раз
       и запускаете программу в любом месте.
72
73 \paragraph{Недостатки языка Python}
74
75 Язык программирования Python использует большой объем памяти. Это может быть
       недостатком при создании приложений, когда предпочтение отдаётся
       оптимизации памяти.
76
77 Python используется для программирования на стороне сервера. Пользователь не
       видит Python на стороне клиента или в мобильных приложениях.
78
79 Python - динамически типизированный язык, поэтому тип данных переменной может
        измениться в любой момент. Переменная, содержащая целое число, в будущем
       может стать строкой, что может привести к ошибкам времени выполнения.
80
81 \subsection{Проектирование пользовательского интерфейса}
82 На основании требований к пользовательскому интерфейсу, представленных в
       пункте 2.3 технического задания, был разработан графический интерфейс
       десктопного приложения с применением python tkinter, customtkinter и
       SQLite. Этот процесс подчеркивает важность интуитивно понятного и
       эффективного взаимодействия с пользователем. Разработанный интерфейс
       ориентирован на обеспечение легкости в использовании и интуитивного
       понимания функционала приложения, предоставляя пользователю простое и
       эффективное взаимодействие с приложением.
83
84 1. \textbf{Навигация по таблицам с данными:} Реализация функции навигации на
       основе полей psr\underline{ }btn, psr\underline{ }ua\underline{ }btn, drs\
       underline{ }btn, rms\underline{ }btn, pns\underline{ }btn, acs\underline{
       };btn
```

2. \textbf {Панель управления для каждой из таблиц с данными:} диверсификация интерфейсов происходит на основе метода show\underline{ }table класса ;Controller

3. \textbf{Отображение текущей таблицы с данными в реальном времени:} актуальность этого графического элемента (дерева) поддерживается за счёт методов TreeRefresh и TreeCreate класса Controller;

4. \textbf{Воздействие на внесённые данные базы внутри приложения:} Добавление, изменение или удаление элементов реализованы в методах add\underline{ }element, update\underline{ }element и delete\underline{ }element класса Tables соответственно, внутри которых формирование строк запросов к БД реализовано с помощью класса Utils.

5. \textbf{Навигация и лёгкий поиск среди элементов среди данных:} Пользователь может переходить по элементам последовательно (метод MoveTo класса Controller), или же использовать отдельное текстовое поле для метода search\underline{ }element класса Tables, перейдя сразу к искомому элементу таблицы.

6. \textbf{Отображение локального времени:} Учитывая специфику проекта -- систему для круизного лайнера, учтено, что часовой пояс может измениться во время путешествия. На уровне python и SQL была установлена привязка к локальному времени (localtime), а функционал часов запущен в асинхронном потоке во избежание ошибок и помех работы основной системы.

7. \textbf{Отображение информации о доступе выбранного пассажира к выбранной двери:} На основе правил, прописанных в полях Комнаты, к которой привязана выбранная Дверь, в отдельном окне отображается информация о том, есть ли у Пассажира доступ к этой Двери.

```
\begin{figure} [ht]
  \centering
  \includegraphics[width=1\linewidth]{images/Example2}
  \caption{модели интерфейса <<Панель управления для каждой из таблиц с
     данными>> и <<Отображение текущей таблицы с данными в реальном времени
     >>}
  \label{fig:example2}
\end{figure}
```

Процесс изменения данных максимально упрощён и включает следующие шаги:

1. В главном меню пользователь выбирает поле с названием необходимого ему вида данных.

2. В появившемся окне пользователь находит нужный ему элемент посредством << Пролистывания>> или посредством поиска по идентификатору, при нажатии на соответствующие подписанные поля.

3. Пользователь изменяет некоторые данные в полях элемента и выбирает опцию <<Изменить>>, если ему нужно было редактировать элемент, или <<Удалить>>, если была необходимость удалить элемент.

4. Если пользователю необходимо добавить элемент, он должен заполнить поля ввода: Наименование, принадлежность, номер, статус, скорость открытия, вместимость (В случае работы с данными о дверях) и выбрать опцию << Добавить>>.

5. На последнем этапе, если все обязательные поля были заполнены данными корректного формата, пользователь получает сообщение об успешном проведении операции и она выполняется.

```
\begin{figure} [ht]
  \centering
  \includegraphics[width=1\linewidth]{images/Example1}
  \caption{модель интерфейса изменения данных элемента в данных о дверях}
  \label{fig:example1}
\end{figure}
```

Процесс получения информации о доступе является еще более простым и состоит из следующих этапов:

1. В главном меню пользователь выбирает опцию <<Проверка доступа>>.

2. В появившемся окне пользователь с помощью переключателя выбирает нужную ему группу поиска среди пассажиров - таблицу <<Пассажиры>> или <<Дети>>.

3. Пользователь вводит идентификаторы пассажира и двери в поля \textquotedbl Пассажир \textquotedbl и \textquotedbl Дверь \textquotedbl соответственно и выбирает опцию <<Проверить доступ>>

4. На последнем этапе, если все обязательные поля были заполнены данными корректного формата и в БД содержатся данные о предоставленном/отказанном доступе, пользователь получает сообщение о том, есть ли у указанного пассажира доступ к указанной двери.

```
\begin{figure} [ht]
  \centering
  \includegraphics[width=1\linewidth]{images/Example3}
  \caption{модель интерфейса <<отображение информации о доступе выбранного
     пассажира к выбранной двери>>.}
  \label{fig:example3}
\end{figure}

\subsection{Диаграмма размещения}
```

Диаграмма размещения, отображаемая на рисунке \ref{fig:commonscheme4}, является фундаментальным инструментом для иллюстрации взаимосвязей между программными и аппаратными компонентами системы. Этот элемент визуализации служит для акцентирования значимости стратегического планирования в процессе разработки распределенных систем. Детальное и глубокое понимание этих взаимосвязей критически важно для успешного создания и функционирования распределенных информационных систем.

Каждый компонент системы, будь то программный или аппаратный, играет важную роль в обеспечении её общей эффективности и надежности.

Подход, основанный на стратегическом планировании, способствует оптимизации этих взаимодействий и повышает вероятность успешной реализации и эксплуатации системы в целом.

```
\begin{figure} [ht]
  \centering
  \includegraphics[width=1\linewidth]{images/CommonScheme4}
  \caption{Диаграмма размещения}
  \label{fig:commonscheme4}
\end{figure}
```

Она является хорошим средством для показа маршрутов перемещения объектов и компонентов в распределенной системе.

Описание архитектуры приложения

Архитектура приложения, реализованная в рамках текущей работы, базируется на модели MVC (Model-View-Controller). Этот паттерн был избран из-за его способности к эффективному распределению функциональных обязанностей между структурными компонентами системы, а также способствованию упрощению процессов разработки и тестирования.

MVC реализован следующим образом:

1. Модель (Model) и Контроллер (Controller): Эти компоненты обеспечивают управление бизнес-логикой и обработку данных, а также связь между пользовательским интерфейсом и базой ;данных

2. Представление (View): Реализовано через фронтенд, использующий пакеты tkinter, customtkinter и treeview, и отвечающий за визуализацию информации и интерактивное взаимодействие с пользователем.

Ключевым аспектом в управлении данными является использование системы управления базами данных SQLite. Этот выбор был обусловлен высокой надежностью SQLite, её устойчивостью к атакам типа \textquotedbl SQL-Инъекция \textquotedbl и гибкостью в обработке сложных запросов, что имеет критическое значение для эффективного функционирования бэкенда.

Применение архитектуры MVC принесло следующие ключевые преимущества:

```
\begin{itemize}
  \item Четкое Разделение Обязанностей: Эффективное разграничение между
    пользовательским интерфейсом (класс main.py и controller.py) и back-end
    логикой (tables.py) значительно упрощает процесс разработки и
    последующей поддержки системы
  \item Гибкость и Масштабируемость: Благодаря MVC, архитектура приложения
    легко адаптируется и масштабируется, что позволяет разработчикам
  модифицировать или расширять отдельные части системы без влияния на ;другие
  \item Упрощение Тестирования: Независимость компонентов архитектуры MVC
    облегчает процедуру тестирования, позволяя проводить её для каждого
    элемента в отдельности
\end{itemize}
```

Такой подход устанавливает предпочтительную форму записи организации функционала и способ его использования в тестировании. Он обеспечивает создание надежной, гибкой и масштабируемой системы, а также применим как в крупных, так и в малых проектах.

Модули:

В backend применяется модуль asyncio для организации работы системного времени в отдельном потоке.

Модуль re включается для работы с регулярными выражениями для задания правил написания, валидации полей.

Модуль sqlite3 используется для организации подключения и запросов к базе данных.

Для создания frontend используются модули tkinter, customtkinter и treeView. TreeView необходим для отображения виджета древа таблицы в реальном времени, customtkinter - для основного интерфейса приложения, а tkinter - для служебного упорядочивания элементов интерфейса customtkinter и treeview.

Классы:

main.py - в этом классе прописан front-end главного меню с последующей передачей вводимых данных в другие компоненты.

Utils.py - служебный класс, внутри которого реализованы функции запросов к базе данных, асинхронное обновление системного времени(Например, подключение к БД - create\underline{ }connection(), чтение - read\underline{ }single\underline{ }row(), запрос - execute\underline{ }query() и т.д.).

Validator.py - служебный класс для задания правил написания (валидации). Необходим для проверки корректности вводимых данных(Например, метод FKValid() - для валидации внешних ключей, overvalidation() для дополнительной проверки корректности перед валидацией)

Controller.py - в этом классе прописана front-end логика панели управления таблицами данных с последующей передачей вводимых данных в другие компоненты, а также выводом полученных данных на экран.

Tables.py - служебный класс, созданный для формирования запросов к БД. Вся логика взаимодействия с данными внутри базы, таким как добавление - add\underline{ }element(), удаление - delete\underline{ }element() и т.д., содержится в этом классе.

Место для диска