**Министерство образования Рязанской области**

**Областное государственное бюджетное профессиональное образовательное учреждение**

**«Касимовский нефтегазовый колледж»**

**Отчет по учебной практике**

**УП ПМ 02**

**«Осуществление интеграции программных модулей»**

|  |  |
| --- | --- |
| Руководитель | |
| Аначко С.Н. | |
| (фамилия, инициалы) | |
|  | |
| (подпись) (дата) | |
| Студент | 23И-1 |
| (группа) | |
| Пенкин Д.С | |
| (фамилия, инициалы) | |
|  | |
| (подпись) (дата) | |

Касимов 2025

**Областное государственное бюджетное профессиональное**

**образовательное учреждение**

**“Касимовский нефтегазовый колледж”**

**ДНЕВНИК**

**По практическому обучению**

|  |
| --- |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Учебная практика ПМ. 02\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  |
| **«Осуществление интеграции программных модулей»** |

Специальность 09.02.07 «Информационные системы и программирование»

Студент Пенкин Даниил Сергеевич

Группа 23И-1

|  |  |  |
| --- | --- | --- |
| **Дата** | **Вид деятельности** | **Подпись студента** |
| 05.05.2025 | Инструктаж по технике безопасности. |  |
| 06.05.2025 | Анализ предметной области. Разработка и оформление технического задания. |  |
| 07.05.2025 | Построение архитектуры программного средства. |  |
| 12.05.2025 | Построение диаграмм вариантов использования и диаграммы последовательности. |  |
| 13.05.2025 | Построение диаграмм компонентов. Построение диаграммы бизнес - процессов |  |
|  | Преобразование реляционной базы данных в модель сущность - связь |  |
| 14.05.2025 | Проектирование реляционной базы данных |  |
|  | Проектирование реляционной базы данных |  |
| 15.05.2025 | Сортировка и фильтрация данных. Поиск данных по одному или нескольким параметрам. Поиск в таблице. |  |
|  | Создание меню различных видов. |  |
| 16.05.2025 | Создание форм и отчетов |  |
| 17.05.2025 | Анализ современных САПР программного обеспечения |  |

Руководитель практики ОГБПОУ “КНГК”

Аначко С.Н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”\_\_\_”\_\_\_\_\_\_2025 г.

Содержание

1. [Анализ предметной области 3](#_Toc164685441)
2. [Функциональная модель информационной системы 4](#_Toc164685442)
3. [Моделирование процессов информационной системы 8](#_Toc164685443)
4. [Логическая модели базы данных 13](#_Toc164685444)
5. [Физическая модель данных 14](#_Toc164685445)
6. [Репозиторий проекта на GitHub 15](#_Toc164685446)
7. [Реализация базы данных 16](#_Toc164685448)
8. [Реализация информационной системы 18](#_Toc164685449)

# Анализ предметной области

**Назначение ИС:**

Информационная система предназначена для автоматизации работы платной поликлиники, обеспечивая удобную запись пациентов на прием к врачам, управление оказанными медицинскими услугами и их оплатой.

**Основные функции:**

1.**Запись пациентов на прием:**

* Онлайн-бронирование времени приема.
* Учет специализаций врачей и их расписания.

**2.Управление медицинскими услугами:**

* Ведение прайс-листа платных услуг.
* Фиксация оказанных услуг .

**3.Оплата услуг:**

* Формирование чеков и квитанций.

**4.Отчетность и аналитика:**

* Формирование отчетов по оказанным услугам, выручке.
* Контроль загрузки врачей.

Технические особенности:

**База данных:** Реляционная СУБД с хранением:

* Данных пациентов (ФИО, контакты, номер медкарты, история посещений).
* Данных врачей (ФИО, специализация, график работы).
* Учет услуг (наименование, стоимость, дата оказания).

# Функциональная модель информационной системы

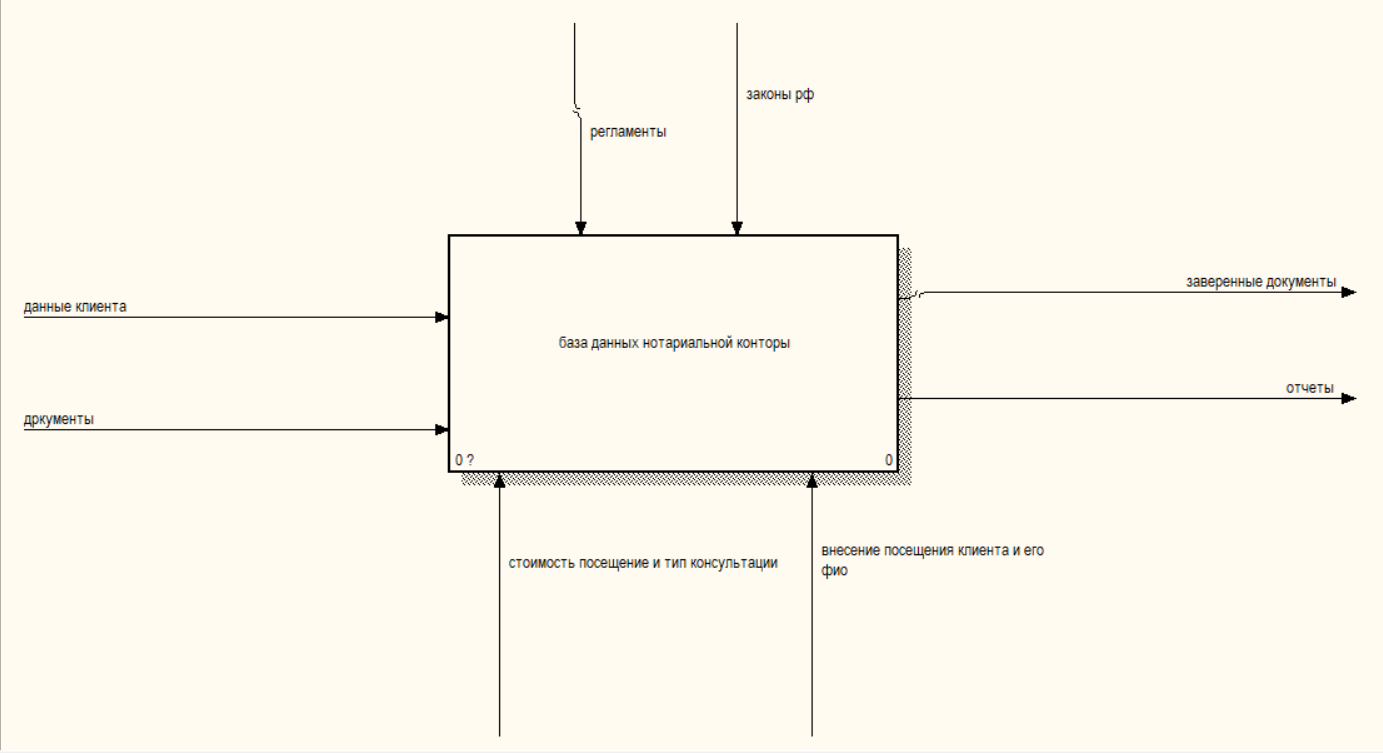


Рисунок 1 - IDEF0 диаграмма предприятия «Нотариальной конторы».

Вход:

Данные клиента

документы

Выход:

Заверенные документы

отчеты

В результате декомпозиции предприятия «Нотариальной конторы» получается диаграмма, представленная на рисунке 2.

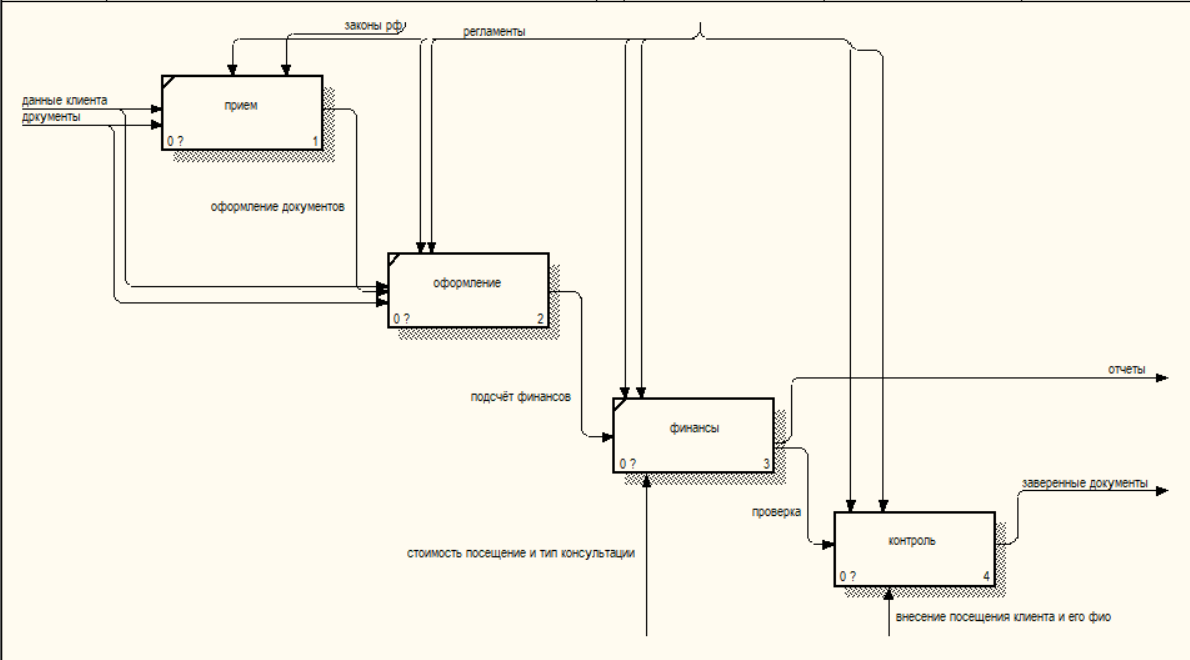


Рисунок 2 - IDEF0 диаграмма декомпозиции бизнес-процессов предприятия «Нотариальной конторы». Уровень А0.

На этой диаграмме выделены основные процессы:

* Оформление документов
* Подсчёт финансов
* Проверка
* контроль

Дальнейшая декомпозиция процесса «Нотариальной конторы» представлена на рисунке 3.

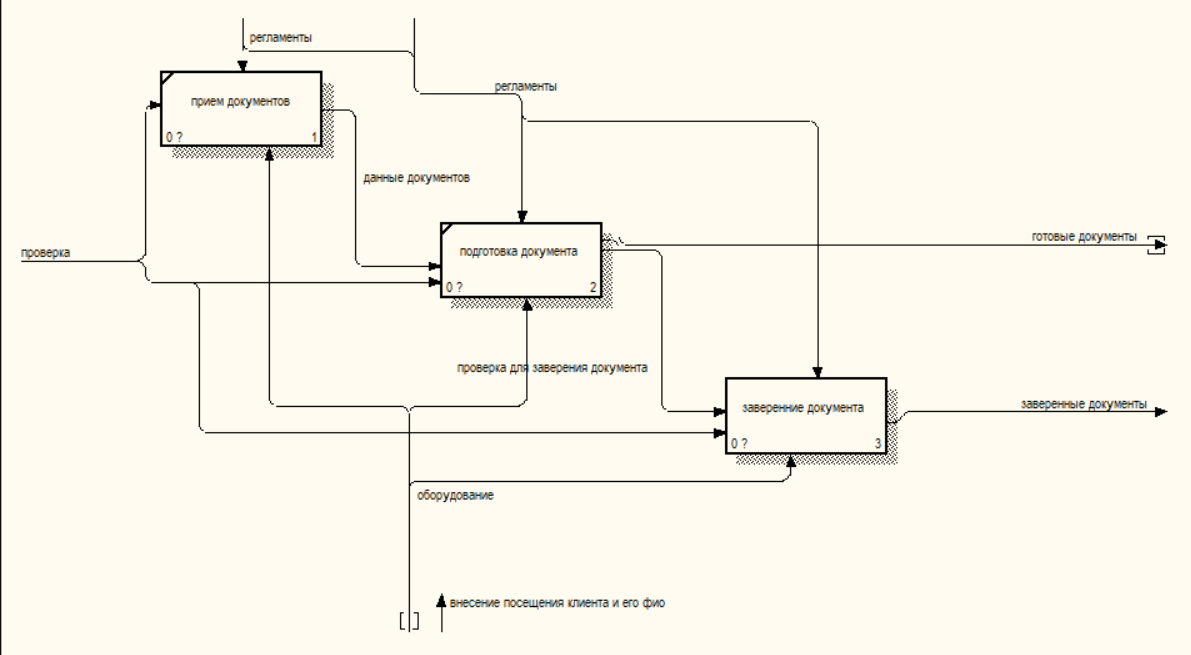


Рисунок 3 - IDEF0 диаграмма. Декомпозиция процесса «Нотариальной конторы» .

Здесь основными процессами являются:

* прием документов
* подготовка документа
* заверение документа
* заверенные документы

Декомпозиции процесса «Нотариальной конторы» представлена на рисунке 4

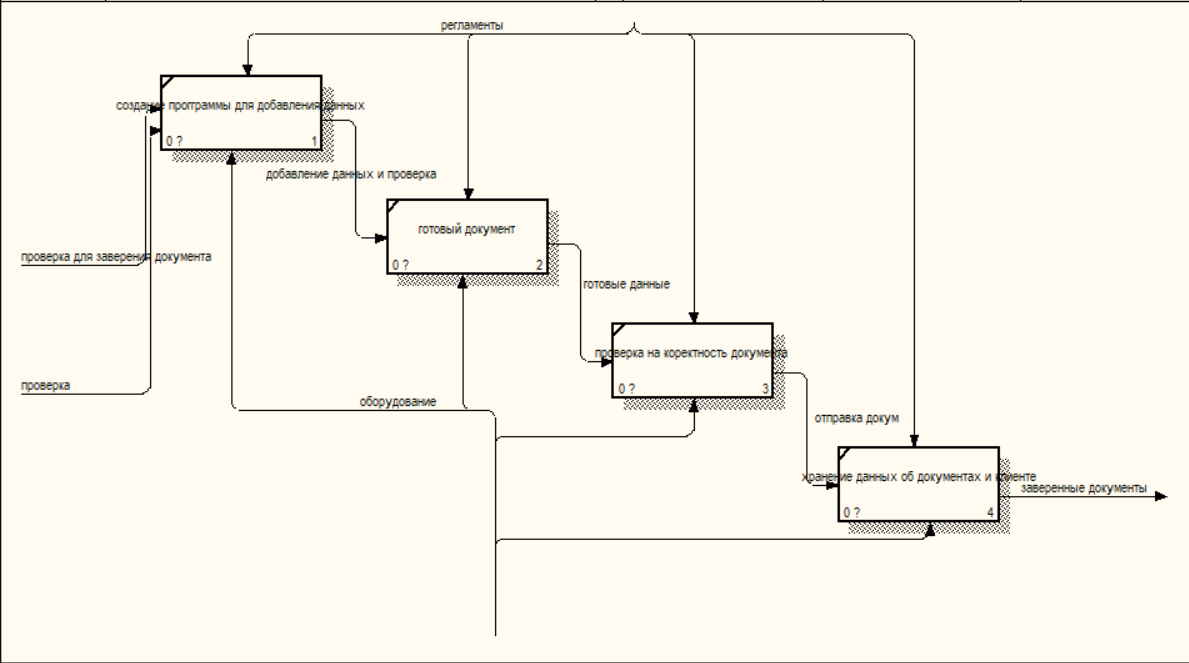


Рисунок 4 - IDEF0 диаграмма. Декомпозиция процесса определение перечня работ.

Основные процессы:

* добавление и проверка данных
* готовый документ
* проверка на коректность документов
* хранение данных об документах и клиенте
* заверенные документы

# Моделирование процессов информационной системы



Рисунок 5 – Диаграмма вариантов использования

# Диаграмма классов



Рисунок 6 – Диаграмма классов

Диаграмма деятельности



Рисунок 7 – диаграмма деятельности

Диаграмма последовательности



Рисунок 8 – Диаграмма последовательности

Диаграмма состояний



Рисунок 9 – Диаграмма состояний

# Логическая модель базы данных



Рисунок 10 – Логическая модель базы данных

# Физическая модель базы данных



Рисунок 11 – Физическая модель базы данных

# Репозиторий проекта на github

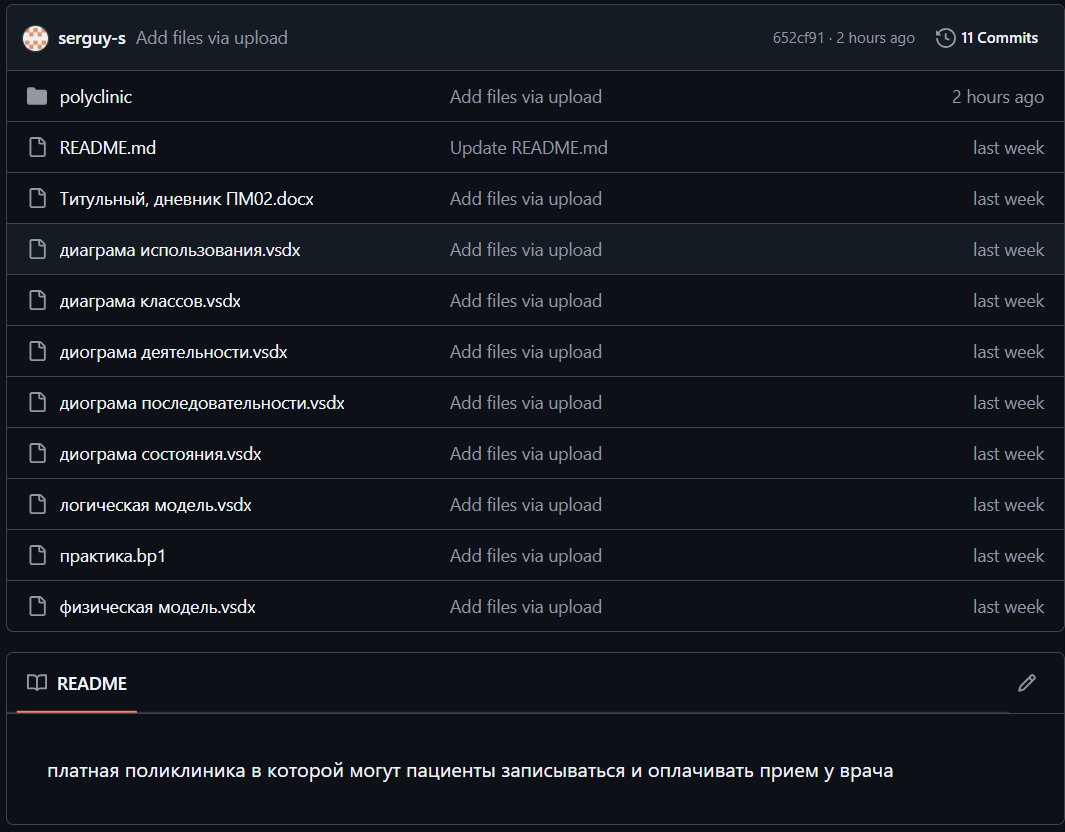


Рисунок 12 – Репозиторий проекта на GitHub

7 Реализация базы данных

Файл config.py в котором будем хранить название базы данных:

DB\_NAME = "access\_control.db"

Файл для создания базы данных db\_bazaa\_dannih.py:

import sqlite3

def get\_connection():

return sqlite3.connect("access\_control.db")

def initialize\_db():

"""Инициализирует базу данных, создает таблицы"""

conn = get\_connection()

cursor = conn.cursor()

try:

# Создание таблицы Role

cursor.execute('''

CREATE TABLE IF NOT EXISTS Role (

role\_id INTEGER PRIMARY KEY AUTOINCREMENT,

role\_name TEXT NOT NULL UNIQUE

)

''')

# Создание таблицы Position

cursor.execute('''

CREATE TABLE IF NOT EXISTS Position (

position\_id INTEGER PRIMARY KEY AUTOINCREMENT,

position\_name TEXT NOT NULL UNIQUE

)

''')

# Создание таблицы Employee

cursor.execute('''

CREATE TABLE IF NOT EXISTS Employee (

employee\_id INTEGER PRIMARY KEY AUTOINCREMENT,

role\_id INTEGER NOT NULL,

position\_id INTEGER NOT NULL,

last\_name TEXT NOT NULL,

first\_name TEXT NOT NULL,

middle\_name TEXT,

phone\_number TEXT,

FOREIGN KEY (role\_id) REFERENCES Role(role\_id),

FOREIGN KEY (position\_id) REFERENCES Position(position\_id)

)

''')

# Создание таблицы Access\_Point

cursor.execute('''

CREATE TABLE IF NOT EXISTS Access\_Point (

access\_point\_id INTEGER PRIMARY KEY AUTOINCREMENT,

access\_point\_name TEXT NOT NULL UNIQUE,

room TEXT

)

''')

# Создание таблицы Access\_Schedule

cursor.execute('''

CREATE TABLE IF NOT EXISTS Access\_Schedule (

schedule\_id INTEGER PRIMARY KEY AUTOINCREMENT,

employee\_id INTEGER NOT NULL,

day\_of\_week TEXT NOT NULL,

entry\_time TEXT NOT NULL,

exit\_time TEXT NOT NULL,

FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id)

)

''')

# Создание таблицы Access\_Log

cursor.execute('''

CREATE TABLE IF NOT EXISTS Access\_Log (

log\_id INTEGER PRIMARY KEY AUTOINCREMENT,

employee\_id INTEGER NOT NULL,

access\_point\_id INTEGER NOT NULL,

event\_id INTEGER NOT NULL,

date TEXT NOT NULL,

time TEXT NOT NULL,

FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id),

FOREIGN KEY (access\_point\_id) REFERENCES Access\_Point(access\_point\_id)

)

''')

# Создание таблицы Service

cursor.execute('''

CREATE TABLE IF NOT EXISTS Service (

service\_id INTEGER PRIMARY KEY AUTOINCREMENT,

service\_name TEXT NOT NULL UNIQUE,

description TEXT

)

''')

conn.commit()

print("База данных успешно инициализирована")

except Exception as e:

print(f"Ошибка при инициализации БД: {e}")

conn.rollback()

finally:

conn.close()

if \_\_name\_\_ == "\_\_main\_\_":

initialize\_db()

8 Реализация информационной системы

Файл Access\_Log.py

from database.db\_baaza\_dannih import get\_connection

class Access\_Log:

def \_\_init\_\_(self, log\_id=None, employee\_id=None, access\_point\_id=None,

event\_id=None, date=None, time=None):

self.log\_id = log\_id

self.employee\_id = employee\_id

self.access\_point\_id = access\_point\_id

self.event\_id = event\_id

self.date = date

self.time = time

def save(self):

conn = get\_connection()

try:

cursor = conn.cursor()

if self.log\_id is None:

cursor.execute('''

INSERT INTO Access\_Log

(employee\_id, access\_point\_id, event\_id, date, time)

VALUES (?, ?, ?, ?, ?)

''', (self.employee\_id, self.access\_point\_id,

self.event\_id, self.date, self.time))

self.log\_id = cursor.lastrowid

else:

cursor.execute('''

UPDATE Access\_Log SET

employee\_id = ?,

access\_point\_id = ?,

event\_id = ?,

date = ?,

time = ?

WHERE log\_id = ?

''', (self.employee\_id, self.access\_point\_id,

self.event\_id, self.date, self.time, self.log\_id))

conn.commit()

except Exception as e:

print(f"Ошибка при сохранении записи доступа: {e}")

raise

finally:

conn.close()

def delete(self):

"""Удаляет запись из журнала доступа"""

if self.log\_id is not None:

conn = get\_connection()

try:

cursor = conn.cursor()

cursor.execute('DELETE FROM Access\_Log WHERE log\_id = ?', (self.log\_id,))

conn.commit()

except Exception as e:

print(f"Ошибка при удалении записи: {e}")

raise

finally:

conn.close()

@staticmethod

def get\_all():

conn = get\_connection()

try:

cursor = conn.cursor()

cursor.execute('SELECT \* FROM Access\_Log')

return [Access\_Log(\*row) for row in cursor.fetchall()]

finally:

conn.close()

@staticmethod

def get\_by\_id(log\_id):

conn = get\_connection()

try:

cursor = conn.cursor()

cursor.execute('SELECT \* FROM Access\_Log WHERE log\_id = ?', (log\_id,))

row = cursor.fetchone()

return Access\_Log(\*row) if row else None

finally:

conn.close()

Файл Access\_Point.py

from database.db\_baaza\_dannih import get\_connection

class Access\_Point:

def \_\_init\_\_(self, access\_point\_id=None, access\_point\_name=None, room=None):

self.access\_point\_id = access\_point\_id

self.access\_point\_name = access\_point\_name

self.room = room

def save(self):

conn = get\_connection()

try:

cursor = conn.cursor()

if self.access\_point\_id is None:

cursor.execute('''

INSERT INTO Access\_Point

(access\_point\_name, room)

VALUES (?, ?)

''', (self.access\_point\_name, self.room))

self.access\_point\_id = cursor.lastrowid

else:

cursor.execute('''

UPDATE Access\_Point SET

access\_point\_name = ?,

room = ?

WHERE access\_point\_id = ?

''', (self.access\_point\_name, self.room, self.access\_point\_id))

conn.commit()

except Exception as e:

print(f"Ошибка при сохранении точки доступа: {e}")

raise

finally:

conn.close()

@staticmethod

def get\_all():

conn = get\_connection()

try:

cursor = conn.cursor()

cursor.execute('SELECT \* FROM Access\_Point')

return [Access\_Point(\*row) for row in cursor.fetchall()]

finally:

conn.close()

файл menu.py

def show\_main\_menu():

print("\n=== Система контроля доступа 😊===")

print("1. Управление сотрудниками🙌")

print("2. Управление журналом доступа🤷‍♂️")

print("0. Выход😢")

return input("Выберите действие:🙃 ")

Файл menu\_orders.py

from models.Access\_Log import Access\_Log

def get\_all\_access\_logs():

"""Возвращает все записи журнала доступа"""

return Access\_Log.get\_all()

def get\_access\_log\_by\_id(log\_id):

"""Возвращает запись журнала по ID"""

return Access\_Log.get\_by\_id(log\_id)

def manage\_access\_events(order\_id):

"""Управление событиями доступа"""

while True:

print("\n=== Управление событиями доступа ===")

logs = get\_all\_access\_logs()

if not logs:

print("Нет записей в журнале.")

else:

for log in logs:

print(f"{log.log\_id}. Сотрудник ID: {log.employee\_id} | "

f"Точка доступа: {log.access\_point\_id} | "

f"Дата: {log.date} {log.time} | "

f"Тип: {'Вход' if log.event\_id == 1 else 'Выход'}")

print("\n1. Добавить запись доступа😶‍🌫️")

print("2. Удалить запись🤩")

print("3. Изменить запись✌️")

print("0. Назад🤯")

choice = input("Выберите действие:🙃 ")

if choice == "1":

print("\n=== Новая запись доступа ===")

try:

new\_log = Access\_Log(

employee\_id=int(input("ID сотрудника: ")),

access\_point\_id=int(input("ID точки доступа: ")),

event\_id=int(input("Тип события (1-Вход, 2-Выход): ")),

date=input("Дата (ГГГГ-ММ-ДД): "),

time=input("Время (ЧЧ:ММ): ")

)

new\_log.save()

print("✅ Запись добавлена.")

except ValueError:

print("❌ Ошибка ввода данных!")

elif choice == "2":

try:

log\_id = int(input("Введите ID записи для удаления: "))

log = get\_access\_log\_by\_id(log\_id)

if log:

log.delete()

print("✅ Запись удалена.")

else:

print("❌ Запись не найдена!")

except ValueError:

print("❌ Неверный ID записи!")

elif choice == "3":

try:

log\_id = int(input("Введите ID записи для редактирования: "))

log = get\_access\_log\_by\_id(log\_id)

if log:

print("\nОставьте поле пустым, чтобы не изменять")

new\_event = input(f"Новый тип события [{log.event\_id}]: ")

new\_date = input(f"Новая дата [{log.date}]: ")

new\_time = input(f"Новое время [{log.time}]: ")

if new\_event: log.event\_id = int(new\_event)

if new\_date: log.date = new\_date

if new\_time: log.time = new\_time

log.save()

print("✅ Запись обновлена.")

else:

print("❌ Запись не найдена!")

except ValueError:

print("❌ Ошибка ввода данных!")

elif choice == "0":

break

else:

print("❌ Неверный ввод!")

def menu\_orders():

"""Главное меню работы с заказами"""

while True:

print("\n=== Журнал доступа ===")

print("1. Показать все записи")

print("2. Добавить запись")

print("3. Удалить запись")

print("4. Управление записями доступа")

print("0. Назад в главное меню")

choice = input("Выберите действие: ")

if choice == "1":

logs = get\_all\_access\_logs()

print("\nВсе записи доступа:")

for log in logs:

print(f"{log.log\_id}: Сотр.ID {log.employee\_id} | "

f"Точка {log.access\_point\_id} | {log.date} {log.time}")

elif choice == "2":

print("\n=== Новая запись ===")

try:

new\_log = Access\_Log(

employee\_id=int(input("ID сотрудника: ")),

access\_point\_id=int(input("ID точки доступа: ")),

event\_id=int(input("Тип (1-Вход, 2-Выход): ")),

date=input("Дата (ГГГГ-ММ-ДД): "),

time=input("Время (ЧЧ:ММ): ")

)

new\_log.save()

print("✅ Запись добавлена.")

except ValueError:

print("❌ Ошибка ввода данных!")

elif choice == "3":

try:

log\_id = int(input("ID записи для удаления: "))

log = get\_access\_log\_by\_id(log\_id)

if log:

log.delete()

print("✅ Запись удалена.")

else:

print("❌ Запись не найдена!")

except ValueError:

print("❌ Неверный ID записи!")

elif choice == "4":

try:

order\_id = int(input("Введите ID заказа для управления: "))

manage\_access\_events(order\_id)

except ValueError:

print("❌ Неверный ID заказа!")

elif choice == "0":

break

else:

print("❌ Неверный ввод!")

Файл menu\_employee.py

from models.Employee import Employee

from models.Role import Role

from models.Position import Position

def menu\_employee():

"""Функция меню управления сотрудниками"""

while True:

print("\n=== Меню сотрудников 😘===")

print("1. Список сотрудников🤑")

print("2. Добавить сотрудника🤦‍♂️")

print("0. Назад😢")

choice = input("Выберите пункт:🤞 ").strip()

if choice == "1":

print("\nСписок сотрудников:")

for emp in Employee.get\_all():

print(f"{emp.employee\_id}: {emp.last\_name} {emp.first\_name}")

elif choice == "2":

print("\nДобавление сотрудника:")

emp = Employee(

last\_name=input("Фамилия: "),

first\_name=input("Имя: "),

phone\_number=input("Телефон: "),

role\_id=1, # Временное значение

position\_id=1 # Временное значение

)

emp.save()

print("✅ Сотрудник добавлен")

elif choice == "0":

break

else:

print("❌ Неверный ввод")

# Важно: убедитесь, что функция существует в глобальной области видимости

if \_\_name\_\_ == "\_\_main\_\_":

menu\_employee()

Файл main.py

import os

import sys

from pathlib import Path

# Добавляем путь к проекту в PYTHONPATH

project\_root = Path(\_\_file\_\_).parent.parent

sys.path.append(str(project\_root))

from database.db\_baaza\_dannih import initialize\_db

from ui.menu import show\_main\_menu

def setup\_paths():

"""Настраивает пути для корректного импорта модулей"""

project\_root = os.path.dirname(os.path.abspath(\_\_file\_\_))

sys.path.append(project\_root)

def main():

setup\_paths()

initialize\_db()

while True:

choice = show\_main\_menu()

if choice == "1":

from ui.menu\_employee import menu\_employee

menu\_employee()

elif choice == "2":

from ui.menu\_orders import menu\_orders

menu\_orders()

elif choice == "0":

print("Выход из системы...")

break

else:

print("Неверный ввод, попробуйте снова.")

if \_\_name\_\_ == "\_\_main\_\_":

main()