МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационны систем и технологий

Кафедра Информационные системы

Дисциплина Проектирование и архитектура програмных систем

**КУРСОВОЙ ПРОЕКТ**

Тема\_\_\_\_\_Проектирование и разработка модуля

ведения графиков работы для клиники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_Кувшинов Т.А.\_\_\_\_\_\_\_\_\_\_/

подпись инициалы, фамилия

Курс\_\_\_\_\_\_третий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Группа\_\_\_ПИбд-31\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление/ специальность 09.03.04 Программная инженерия

Руководитель \_\_\_\_\_\_\_ст.преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, ученая степень, ученое звание

\_\_\_\_\_Корунова Надежда Владимировна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя, отчество

Дата сдачи:

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

Дата защиты:

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

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ульяновск

2025 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационны систем и технологий

Кафедра Информационные системы

Дисциплина Проектирование и архитектура програмных систем

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

студенту \_\_ПИбд-31\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_Кувшинво Т.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

группа фамилия, инициалы

**Тема проекта** \_\_\_\_ Проектирование и разработка модуля

ведения графиков работы для клиники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Срок сдачи законченного проекта** «30» мая 2025 г.

**Исходные данные к проекту** \_\_\_\_\_\_техническое задание к курсовой работе\_\_\_\_\_\_\_

(базовое предприятие, характер курсового проекта (работы):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

задание кафедры, инициативная НИР, рекомендуемая литература, материалы практики)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Содержание пояснительной записки** (перечень подлежащих разработке вопросов)

Описание предметной области

Требования к проекту

Оценка архитектурного решения

Проектное решение

Реализация приложения (контрольный пример)

**Перечень графического материала** (с точным указанием обязательных чертежей)

Диаграмма IDEF0

Диаграмма вариантов использования

Диаграмма классов

Диаграмма последовательности

Диаграмма компонентов

Диаграмма развертывания

Руководитель \_\_\_\_ст.преп.\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ /\_Н.В. Корунова\_\_\_\_\_\_/

должность подпись инициалы, фамилия

« 25 »\_февраля\_\_\_\_\_\_\_\_\_\_2025 г

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **/\_\_\_**Т.А. Кувшинов**\_\_\_\_\_\_\_\_\_\_\_\_/**

подпись инициалы, фамилия

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

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЗЫВ  
руководителя на курсовой проект**

студента \_\_\_\_\_\_Кувшинова Тимура Александровича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя и отчество

Факультет\_\_\_ИСТ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_группа\_ПИбд-31\_\_\_\_\_курс\_\_3\_\_\_\_

Дисциплина\_\_Проектирование и архитектура програмных систем\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема проекта (работы) \_\_\_\_\_ Проектирование и разработка модуля

ведения графиков работы для клиники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Структура пояснительной записки курсового проекта соответствует заданию и теме. В ходе работы над курсовым проектом сформировано проектное решение, предоставлены обоснование и реализация данного проектного решения в виде контрольного примера. Студент выполнил курсовой проект самостоятельно.

Программный продукт по проектному решению реализован в виде трех клиентов и сервера для обращения к БД.

Работа выполнена и сдана по графику в установленный срок.

Руководитель\_\_ ст.преп.\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ /Н.В. Корунова /

должность, учёная степень, ученое звание подпись инициалы, фамилия

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

**Оглавление**

[Введение 5](#_Toc196249192)

[Описание предметной области 6](#_Toc196249193)

[Требования к проекту 7](#_Toc196249194)

[Оценка архитектурного решения 10](#_Toc196249195)

[Проектное решение 16](#_Toc196249196)

[Реализация приложения 21](#_Toc196249197)

[Выводы по реализации приложения на основании проекта 24](#_Toc196249198)

[Заключение 25](#_Toc196249199)

[Библиографический список 26](#_Toc196249200)

[ПРИЛОЖЕНИЕ. Листинг кода 27](#_Toc196249201)

# Введение

Целью курсового проекта является клиент-серверной информационной системы для автоматизации процесса ведения графика работы медицинского персонала в клинике. Система должна обеспечивать удобное планирование смен, учёт отпусков и праздников, автоматическое формирование расписаний, а также предоставление доступа к информации через различные клиентские интерфейсы.

Пользователями разрабатываемой системы являются сотрудники медицинского учреждения (врачи, медицинские сёстры и административный персонал), технические специалисты, а также разработчики, осуществляющие сопровождение и развитие программного обеспечения.

Объектом исследования являются информационные системы, предназначенные для автоматизации процессов управления персоналом в учреждениях здравоохранения.

Предметом исследования в данной работе является архитектура и реализация программного обеспечения для формирования и управления расписанием работы медицинского персонала клиники с возможностью многоплатформенного доступа к функционалу.

В качестве методов исследования используется анализ предметной области, исследование и моделирование процессов, разработка проекта с использованием UML-диаграмм, а также реализация системы на языке программирования высокого уровня.

Для моделирования информационной системы был использован структурный подход (нотация IDEF0 - интструмент Ramus) и объектно-ориентированный подход (нотация UML – интструмент Visual Paradigm). При реализации использовался язык программирования C# с использование сторонних библиотек и фреймворков. В качестве системы управления базами данных использовался PostgreSQL. Клиентская часть разрабатывалась с использованием Windows Forms (для настольного приложения), Python (для Telegram-бота) и Flutter (для Android-приложения).

# Описание предметной области

Современные медицинские учреждения требуют эффективной системы управления рабочим временем персонала. Расписание работы медицинских сотрудников — это ключевой организационный инструмент, от которого напрямую зависит бесперебойная деятельность отделений, качество оказания медицинских услуг, а также удовлетворенность как пациентов, так и самих сотрудников.

В типичной клинике работают врачи различных специальностей, медицинские сёстры, санитарный персонал и административный штат. Все они подчиняются внутренним регламентам работы, графику смен, правилам учета отпусков и праздничных дней. Формирование и поддержание актуального расписания — задача, традиционно решаемая вручную или с помощью простых таблиц, что приводит к ошибкам, потере информации и неэффективному использованию ресурсов.

Основными процессами предметной области являются:

* учёт состава медицинского персонала с привязкой к отделениям;
* ведение графика смен для каждого отделения;
* учет отпусков, больничных, праздничных и выходных дней;
* формирование расписания на основе заданных параметров;
* автоматическая генерация график;
* предоставление персоналу доступа к актуальному расписанию через удобные каналы (мобильное приложение, Telegram-бот и др.);
* экспорт расписаний и статистики по загрузке персонала за период.

# Требования к проекту

**1. Назначение и цели создания системы**

Система «Клиника. Ведение графика работы медицинского персонала» предназначена для автоматизации процессов хранения, формирования, редактирования и предоставления расписаний сотрудников клиники (врачей, медсестёр и других специалистов), а также для уведомления сотрудников о предстоящих сменах и доступе к статистике по отработанному времени.

Целью создания системы является оптимизация управления рабочим временем персонала, уменьшение времени, затрачиваемого администраторами на формирование графиков, улучшение информирования сотрудников, повышение прозрачности и управляемости внутреннего трудового распорядка.

**2. Требования к системе**

Требования к пользователям системы

* Пользователи должны иметь базовые навыки работы с ПК, мобильными приложениями и мессенджерами.
* Администратор должен обладать расширенными правами и навыками для управления персоналом и расписанием.
  1. Требование к функциям (задачам), выполняемым системой

| **Функция** | **Исходные данные** | **Результат** |
| --- | --- | --- |
| Клиентская часть (неавторизованный пользователь) | | |
| Авторизация | Запрос к серверу приложений для авторизации пользователя в системе  Запрос содержит следующие данные:  login, пароль | В случае корректных данных:  Ответ от сервера приложений, переход на главную страницу приложения.  В случае некорректных данных:  Ответ от сервера приложений, с ошибкой и отображении ее на странице авторизации. |
| Клиентская часть (пользователь авторизованный) desktop client | | |
| Добавление мед. персонала | ФИО, должность, отделение, контакты | Новый сотрудник добавлен в систему |
| Удаление мед. персонала | ID сотрудника | Сотрудник удалён из списка |
| Просмотр мед. персонала | Запрос без параметров или с фильтром | Список сотрудников |
| Изменение мед. персонала | ID сотрудника, новые данные | Обновление информации о сотруднике |
| Добавление должностей | Название должности, описание | Должность добавлена в справочник |
| Удаление мед. должностей | ID должности | Должность удалена из справочника |
| Просмотр мед. должностей | - | Список должностей |
| Изменение мед. должностей | ID должности, новые данные | Обновление должности |
| Добавление отделения | Название, описание, этаж, номер | Новое отделение создано |
| Удаление отделения | ID отделения | Удаление отделения |
| Просмотр отделений | Запрос без параметров или с фильтром | Список отделений |
| Изменение отделения | ID отделения, новые данные | Обновление информации об отделении |
| Добавление праздника | Дата, название | Праздник добавлен в список исключений |
| Удаление праздника | ID праздника | Праздник удалён |
| Просмотр праздников | - | Список праздников |
| Изменение праздника | ID праздника, новые данные | Праздник отредактирован |
| Добавление запсиси | время, необходимый персонал, описание | Запсись добавленна в список исключений |
| Удаление запсиси | ID запсиси | Запсись удалена |
| Просмотр запсисей | - | Список запсисей |
| Изменение запсиси | ID запсиси, новые данные | Запсись отредактирована |
| Добавление отпуска | ID сотрудника, даты отпуска | Отпуск сотрудника зафиксирован |
| Удаление отпуска | ID отпуска | Отпуск удалён |
| Просмотр отпусков | Фильтр по сотруднику или дате | Список отпусков |
| Изменение отпуска | ID отпуска, новые даты | Данные отпуска обновлены |
| Сформировать расписание мед. персонала | Отделение, период, список сотрудников | График сформирован вручную |
| Изменить расписание мед. персонала | ID смены, новые параметры | Расписание изменено |
| Автоматическое формирование расписания по отделению | Отделение, период, правила формирования | Автоматически созданное расписание |
| Получение расписания отделения за период | Отделение, дата начала и окончания | Таблица смен за период |
| Экспорт расписания отделения за период | Отделение, период, формат (Excel) | Скачиваемый файл с расписанием |
| Сформировать статистику по мед. персоналу (кто сколько работал в месяц) | Период, ID сотрудников | Таблица с количеством смен/часов по сотрудникам |
| Клиентская часть (пользователь авторизованный) андроид приложение | | |
| Просмотр предстоящих записей | - | Предстоящие записи |
| Клиентская часть (пользователь авторизованный) тг бот | | |
| Уведомление сотрудника о приёме | Сотрудник, приём | Уведомление сотрудника |
| Серверная часть (сервер-приложение) | | |
| Добавление мед. персонала | ФИО, должность, отделение, контакты | Новый сотрудник добавлен в систему |
| Удаление мед. персонала | ID сотрудника | Сотрудник удалён из списка |
| Просмотр мед. персонала | Запрос без параметров или с фильтром | Список сотрудников |
| Изменение мед. персонала | ID сотрудника, новые данные | Обновление информации о сотруднике |
| Добавление должностей | Название должности, описание | Должность добавлена в справочник |
| Удаление мед. должностей | ID должности | Должность удалена из справочника |
| Просмотр мед. должностей | - | Список должностей |
| Изменение мед. должностей | ID должности, новые данные | Обновление должности |
| Добавление отделения | Название, описание, этаж, номер | Новое отделение создано |
| Удаление отделения | ID отделения | Удаление отделения |
| Просмотр отделений | Запрос без параметров или с фильтром | Список отделений |
| Изменение отделения | ID отделения, новые данные | Обновление информации об отделении |
| Добавление праздника | Дата, название | Праздник добавлен в список исключений |
| Удаление праздника | ID праздника | Праздник удалён |
| Просмотр праздников | - | Список праздников |
| Изменение праздника | ID праздника, новые данные | Праздник отредактирован |
| Добавление запсиси | время, необходимый персонал, описание | Запсись добавленна в список исключений |
| Удаление запсиси | ID запсиси | Запсись удалена |
| Просмотр запсисей | - | Список запсисей |
| Изменение запсиси | ID запсиси, новые данные | Запсись отредактирована |
| Добавление отпуска | ID сотрудника, даты отпуска | Отпуск сотрудника зафиксирован |
| Удаление отпуска | ID отпуска | Отпуск удалён |
| Просмотр отпусков | Фильтр по сотруднику или дате | Список отпусков |
| Изменение отпуска | ID отпуска, новые даты | Данные отпуска обновлены |
| Сформировать расписание мед. персонала | Отделение, период, список сотрудников | График сформирован вручную |
| Изменить расписание мед. персонала | ID смены, новые параметры | Расписание изменено |
| Автоматическое формирование расписания по отделению | Отделение, период, правила формирования | Автоматически созданное расписание |
| Получение расписания отделения за период | Отделение, дата начала и окончания | Таблица смен за период |
| Экспорт расписания отделения за период | Отделение, период, формат (Excel) | Скачиваемый файл с расписанием |
| Сформировать статистику по мед. персоналу (кто сколько работал в месяц) | Период, ID сотрудников | Таблица с количеством смен/часов по сотрудникам |

1. **Требования к атрибутам качества**

3.1 Требования к производительности

К количеству одновременно работающих пользователей системы на начальном этапе не должно превышать 100 пользователей

Количество одновременно выполняемых запросов к серверу не должно превышать не должно превышать 50.

Время отклика сервера на входящие запросы не должно превышать 1 секунды с при количестве одновременно работающих пользователей, не превышающем 50.

Время отклика сервера на входящие запросы не должно превышать 5 секунды с при количестве одновременно работающих пользователей, превышающем 50.

3.2 Требования к безопасности

Предъявляются требования к авторизации пользователей с использованием токенов доступа

3.3 Требования к эргономике и технической эстетике

Дизайн пользовательского интерфейса клиентского приложения должен быть интуитивно понятным, минималистичным, с использованием стандартных компонентов управления.

3.4 Требования к защите информации от несанкционированного доступа

Для получения полного доступа к функциям системы необходимо пройти процедуру аутентификации и авторизации.

Для надёжной передачи данных между компонентами системы используется шифрование на уровне транспортного протокола (TLS/SSL)

3.5 Требования к доступности

Непрерывная работа системы. Система должна быть доступна 99% времени

3.6 Требования к масштабируемости

Архитектура системы должна предусматривать возможность горизонтального масштабирования серверной части (API и БД).

Добавление новых клиентских платформ не должно требовать существенных изменений в серверной части.

3.7 Требования к сопровождаемости и расширяемости

Код клиентской и серверной части должен быть структурирован, документирован и соответствовать принятым стандартам кодирования.

Возможность добавления новых модулей и функций без необходимости переписывания текущей логики.

Использование многослойной архитектуры (разделение на слои: UI, бизнес-логика, доступ к данным).

3.8 Требования к совместимости

Система должна корректно работать на Windows 10/11 для Desktop-клиента.

Android-приложение должно поддерживаться на версиях Android 9.0 и выше.

Telegram-бот должен функционировать в стандартном Telegram-клиенте (desktop и mobile).

Серверная часть должна быть развёртываема на ОС Linux и Windows Server.

1. **Требования к сквозной функциональности**
   1. Обработка ошибок в системе. Должна быть предусмотрена обработка исключительных ситуаций для бесперебойной работы.
   2. Функция логирования ошибок системы (протоколирование) и логирование операций пользователя. Должна быть поддержка нескольких уровней логгирования, например, для ошибок – уровень «error», для отладочной информации – «debug».
   3. Валидация полей ввода в клиентском приложении. В системе должна быть предусмотрена проверка доступных полей на заполненность и соответствие различным ограничениям.
   4. Валидация данных приходящих на сервер. В системе должна быть предусмотрена проверка данных для записи в БД.

# Оценка архитектурного решения

Предоставим несколько потенциальных архитектурных решений и оценим их с помощью облегчённого метода оценки архитектур-альтернатив («Lightweight Architecture Alternative Assessment Method»).

**1. Построение дерева атрибутов качества.**

Для разрабатываемой системы можно выделить следующие категории и подкатегории атрибутов качества:

1. *Доступность: непрерывная работа системы*
2. *Производительность: Масштабируемость системы при высокой загрузке, быстрый отклик на запрос пользователя*
3. *Расширяемость: простота добавления нового функционала*
4. *Сложность реализации: начальная разработка архитектуры проекта*

**2. Ранжирование узлов дерева.**

На данном шаге необходимо определить приоритет каждой категории и подкатегории атрибутов качества и на основе полученных рангов распределить веса узлов дерева. Чем больше вес узла, тем приоритетнее данный атрибут качества для разрабатываемой системы.

Расставляются приоритеты и присваиваются значения весов согласно таблице весов ранга:



Результат ранжирования представлен на рисунке 1.

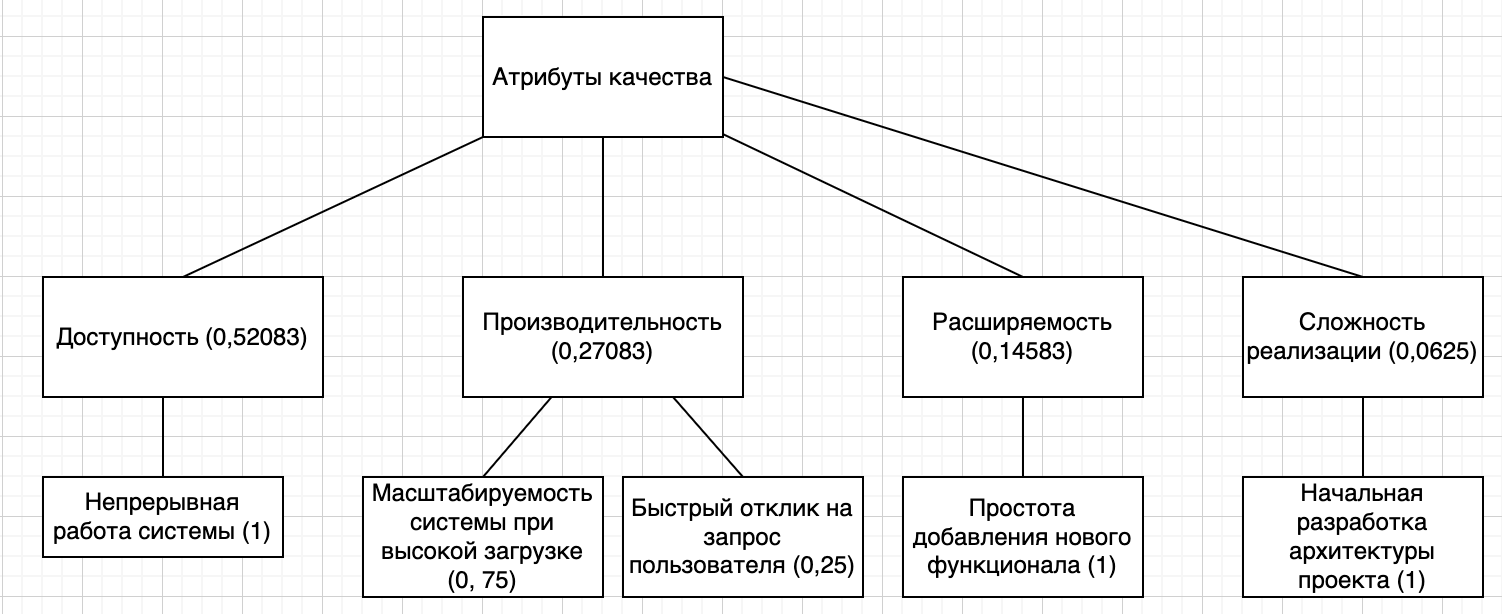


Рисунок 1 Атрибуты качества

**3. Предоставление архитектур-кандидатов.**

В качестве альтернативных решений будут рассматриваться следующие варианты архитектур:

а) Многокомпонентная архитектура

Многокомпонентная архитектура — это стиль проектирования программного обеспечения, при котором приложение строится как набор логически разделённых, но внутренне связанных компонентов. Каждый компонент отвечает за свою область функциональности (например, управление персоналом, формирование расписания, учет отпусков и т.д.), но все компоненты собираются и развертываются как единое приложение.

Преимущества:

* Разделение ответственности: каждый компонент отвечает за свою зону логики, что упрощает разработку и сопровождение.
* Переиспользуемость кода: общие модули (например, модели или вспомогательные утилиты) можно переиспользовать в нескольких частях приложения.
* Облегченная командная разработка: разработчики могут работать над разными компонентами, не мешая друг другу, благодаря четким границам и интерфейсам.
* Упрощенное развертывание: несмотря на логическую модульность, все компоненты входят в один исполняемый файл или размещаются на одной платформе, что упрощает внедрение и поддержку.

Недостатки:

* Ограниченная масштабируемость: нельзя масштабировать отдельные части системы независимо — масштабируется всё приложение целиком.
* Зависимость между компонентами: тесная связь между компонентами может привести к повышенной связанности, что затрудняет изменения в отдельных частях.
* Менее гибкое обновление: для внесения изменений в один компонент может потребоваться перекомпиляция и развёртывание всей системы.
* Выбор единой технологии: все компоненты, как правило, реализуются в одной технологической среде (например, .NET), что ограничивает свободу выбора инструментов.

Данная система в реализации микросервисной архитектуры будет выглядеть следующим образом:

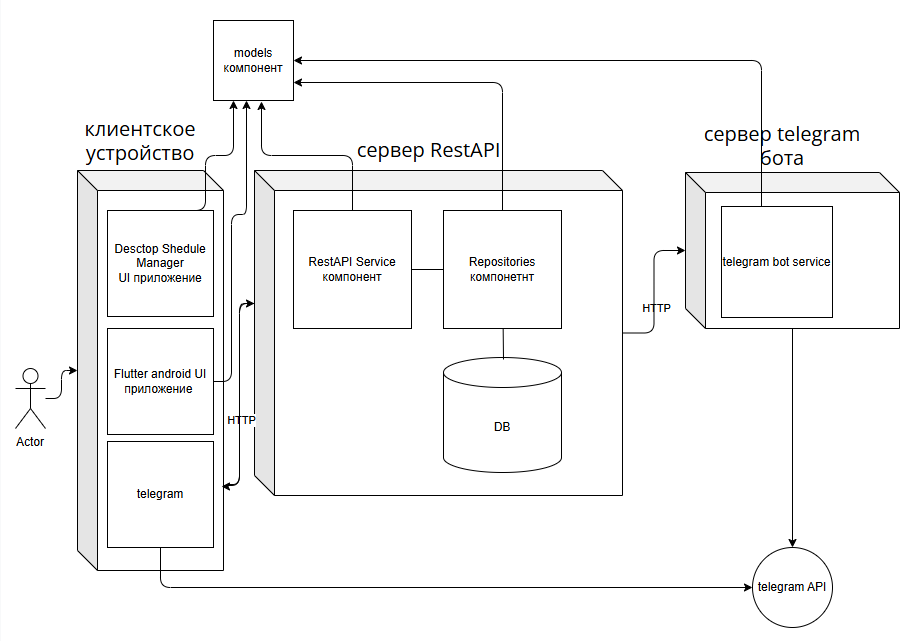


Рисунок 2 Многокомпонентная архитектура

б) Монолитная архитектура

Монолитная архитектура - это классический подход к проектированию приложений, когда весь код и логика хранятся в одном приложении, которое запускается на одном сервере.

Преимущества:

* Простота: монолит приложение относительно проще в разработке и управлении, чем система, состоящая из микросервисов.
* Легко масштабируемость: монолит можно развернуть на нескольких серверах и настроить балансировку нагрузки, что позволяет успешно масштабировать приложение.
* Удобное тестирование: для монолита тестирование происходит внутри единого процесса и необходимые связи между различными компонентами существуют уже на уровне программного кода.
* Простота развертывания: монолит легко развернуть на любом сервере, таким образом, минимизируя вероятность возникновения проблем.

Недостатки:

* Сложность масштабирования: монолит сложнее масштабировать, чем систему, разбитую на микросервисы.
* Высокий уровень связности: в монолите между компонентов может существовать тесная связь, что делает проект менее гибким и менее устойчивым к изменениям.
* Сложность внесения изменений: внесение изменений в одно компоненты может затронуть другие компоненты, что может привести к проблемам в работе всего приложения.
* Необходимость внесения изменений: монолит может стать слишком громоздким и сложно крупным для поддержки со временем, что может привести к большим затратам на разработку и поддержку.

Данная система в реализации монолитной архитектуры будет выглядеть следующим образом:

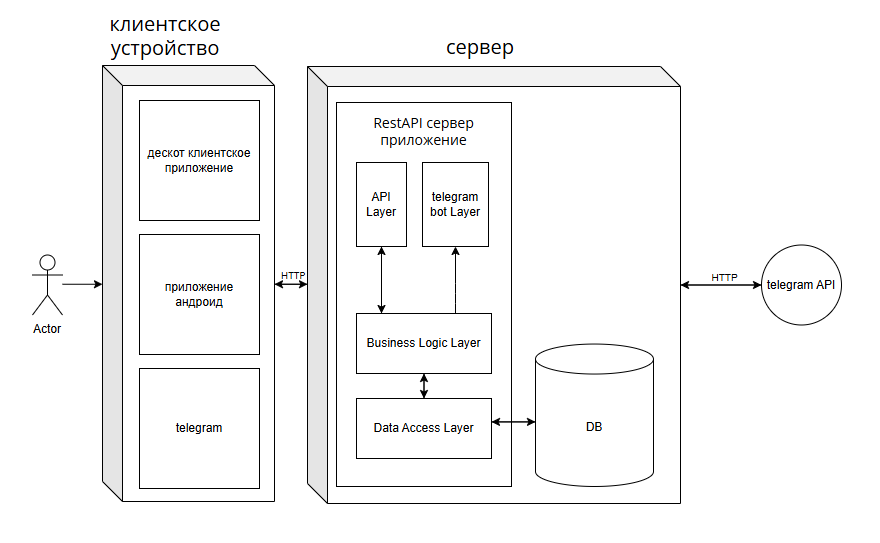


Рисунок 3 Монолитная архитектура

Сравним два решения согласно выделенным сценариям атрибутов качества системы.

**4. Оценка решений**

|  |  |  |  |
| --- | --- | --- | --- |
| **Сценарий** | **Вес** | **Архитектура 1** | **Архитектура 2** |
| 1.Непрерывная работа системы (99%) | 0,5208 | 4 | 3 |
| 2.1. Быстрый отклик на запрос пользователя до 2 секунд | 0,0677 | 3 | 4 |
| 2.2. Масштабируемость различных модулей системы при высокой загрузке | 0,2031 | 4 | 2 |
| 3. Простота добавления нового функционала | 0,1458 | 4 | 2 |
| 4. Начальная разработка архитектуры | 0,0625 | 3 | 4 |
| Итого |  | 3,8696 | 2,7812 |

**5. Анализ результатов**

По полученным результатам можно сделать вывод о том, что многокомпонентная архитектура является наиболее подходящим под имеющиеся нужды решением среди представленных вариантов.

Далее рассмотрим проектное решение на основе многокомпонентной архитектуры.

# Проектное решение

**Архитектурная модель**, используемая в данном приложении, представляет собой слоистую архитектуру. Она разбита на несколько слоев, каждый из которых отвечает за определенную функциональность, включая слой представления, слой моделей и слой работы с данными.

**Архитектурный стиль,** выбранный для данного приложения, — многокомпонентная архитектура. Данный стиль предполагает логическое разделение системы на отдельные компоненты, каждый из которых отвечает за определенную функциональность, но при этом все компоненты взаимодействуют друг с другом в рамках единой системы.

В составе проекта можно выделить следующие ключевые компоненты:

* Models — библиотека, содержащая общие модели данных, используемые всеми остальными компонентами;
* Repositories — компонент доступа к данным, реализующий работу с базой данных через модели;
* API — сервер приложений, реализующий бизнес-логику и предоставляющий HTTP-интерфейс для клиентских приложений;
* Desktop-клиент — приложение для Windows, взаимодействующее с API;
* Telegram-бот (на Python) — облегчённый клиентский интерфейс для оповещения и получения информации;
* Android-приложение (на Flutter) — мобильный клиент с пользовательским интерфейсом и запросами к API.

**Паттерны и фреймворки**

В рамках реализации проекта были использованы следующие архитектурные шаблоны (паттерны) и фреймворки:

Фреймворки:

* Windows Forms (WinForms) — фреймворк для создания пользовательского графического интерфейса на платформе .NET. Используется для разработки десктоп-клиента системы.
* Entity Framework Core — ORM-фреймворк, применяемый на серверной стороне приложения для взаимодействия с базой данных. Обеспечивает работу с данными на уровне объектов и упрощает реализацию репозиториев.
* ASP.NET Core — фреймворк для построения веб-приложений и API. Используется для создания серверной части системы, обрабатывающей запросы от клиентов и реализующей бизнес-логику.

Шаблоны проектирования (паттерны):

* Singleton — применяется как в клиентской, так и в серверной части системы.
  + В клиентском приложении используется для централизованного доступа к API (объект-обёртка для отправки HTTP-запросов к API и управление токеном авторизации).
  + В серверной части применяется для управления подключением к базе данных и обеспечения единого контекста взаимодействия с БД.\

**Итог применения паттернов и фреймворков:**

Использование перечисленных фреймворков позволило значительно ускорить разработку и обеспечить стабильную работу всех компонентов системы. Применение шаблонов проектирования, в частности паттерна Singleton, способствовало упрощению управления состоянием приложений и обеспечению централизованного доступа к ключевым ресурсам, таким как API и база данных.

Благодаря архитектурному разделению ответственности, основной бизнес-логики на стороне сервера и использованию проверенных фреймворков, удалось достичь высокой масштабируемости, упрощённого сопровождения, а также гибкости в дальнейшем развитии системы.

**IDE и языки программирования**

В процессе разработки системы использовались следующие языки программирования и интегрированные среды разработки:

* Язык программирования C#

Использовался для создания серверной части (API), бизнес-логики, а также десктоп-клиентского приложения.

Среда разработки: Visual Studio — мощная интегрированная среда, обеспечивающая удобную разработку, отладку, сборку и публикацию C#-приложений.

* Язык программирования Python

Применялся для реализации Telegram-бота, выполняющего функции уведомления пользователей.

Среда разработки: Visual Studio Code (VS Code) — лёгкая и расширяемая среда, подходящая для разработки на Python, с возможностью быстрого запуска и отладки.

* Язык программирования Dart (фреймворк Flutter)

Использовался для разработки мобильного Android-приложения.

Среда разработки: Android Studio — официальная среда разработки для Android, предоставляющая полный набор инструментов для создания, сборки и тестирования Flutter-приложений.

Такой выбор языков и сред разработки обусловлен их широкой поддержкой, удобством в реализации задач разного уровня и совместимостью с архитектурой системы.

**Структурная модель проекта**

На рисунках 4, 5 предоставлена контекстная диаграмма и декомпозиция в нотации IDEF0.

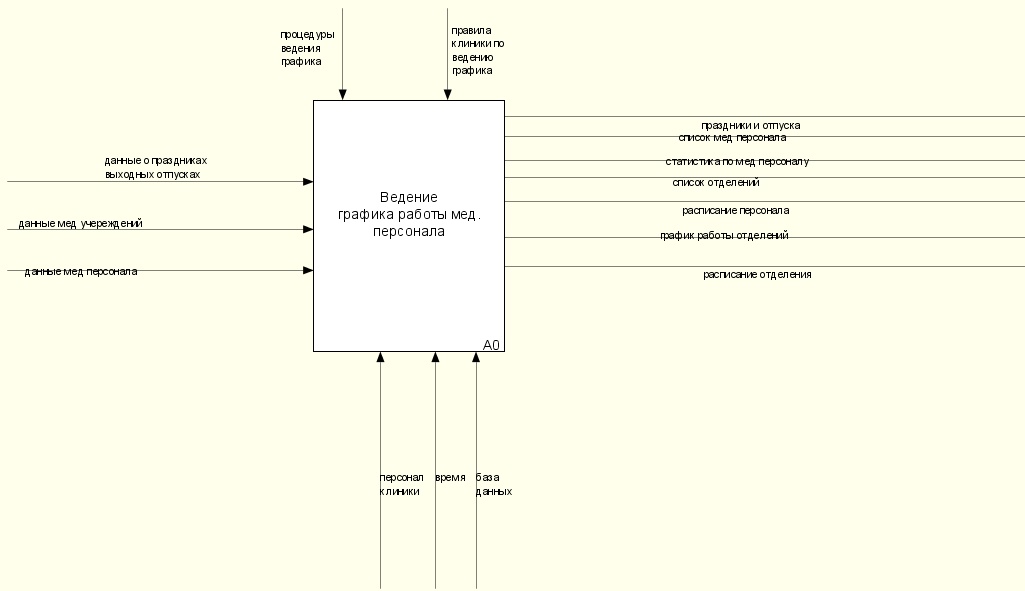


Рисунок 4 IDEF0 контекстная диаграмма

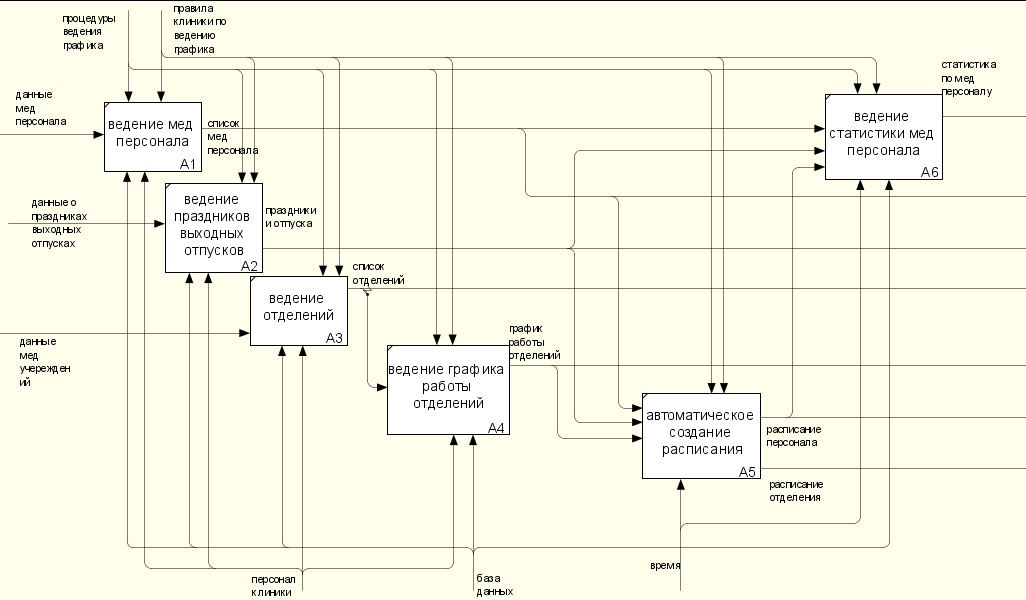


Рисунок 5 IDEF0 диаграмма декомпозиции

**Модель вариантов использования** **(Use Case Diagram)**

Диаграмма вариантов использования описывает взаимодействие внешних участников (акторов) с системой через различные функциональные сценарии (прецеденты). Она позволяет наглядно отразить, какие именно действия доступны различным категориям пользователей, и какие функции реализуются внутри системы.

На рисунке 6 изображена диаграмма вариантов использования, включающая три основных типа акторов:

1. Неавторизованный пользователь

Этот актор представляет нового пользователя, ещё не вошедшего в систему.

Доступные прецеденты:

* Регистрация — создание нового аккаунта с указанием имени, почты и пароля.
* Авторизация — вход в систему с использованием уже существующего логина и пароля.

2. Медицинский персонал (персонал)

Этот актор представляет сотрудников клиники, таких как врачи, медсёстры и т.д.

Доступные прецеденты:

* Получение расписания сотрудника — просмотр своего рабочего расписания (например, через Android-приложение или Telegram-бот).
* Уведомление о предстоящем приёме — получение автоматических уведомлений о назначенных приёмах/сменах через Telegram-бот.

3. Администратор

Актор, обладающий полным доступом к административному функционалу системы (например, через Desktop-приложение).

Доступные прецеденты:

* Формирование статистики по мед. персоналу — генерация отчётов (например, кто сколько отработал за месяц).
* Экспорт расписания за период — выгрузка смен и дежурств медперсонала в PDF или другой формат.
* Создание графика работы отделений — задание общего графика работы для конкретных отделений клиники.
* Формирование расписания мед. персонала — назначение смен для сотрудников.
* Изменение расписания — ручное редактирование расписания в случае необходимости.
* Автоматическое формирование расписания — генерация смен по заданным правилам автоматически.
* Ведение списка мед. персонала — добавление, редактирование и удаление данных сотрудников.
* Ведение списка отделений — управление отделениями (создание, изменение, удаление).
* Управление праздниками и отпусками — указание праздничных дней и периодов отпусков, с учётом которых формируется расписание.

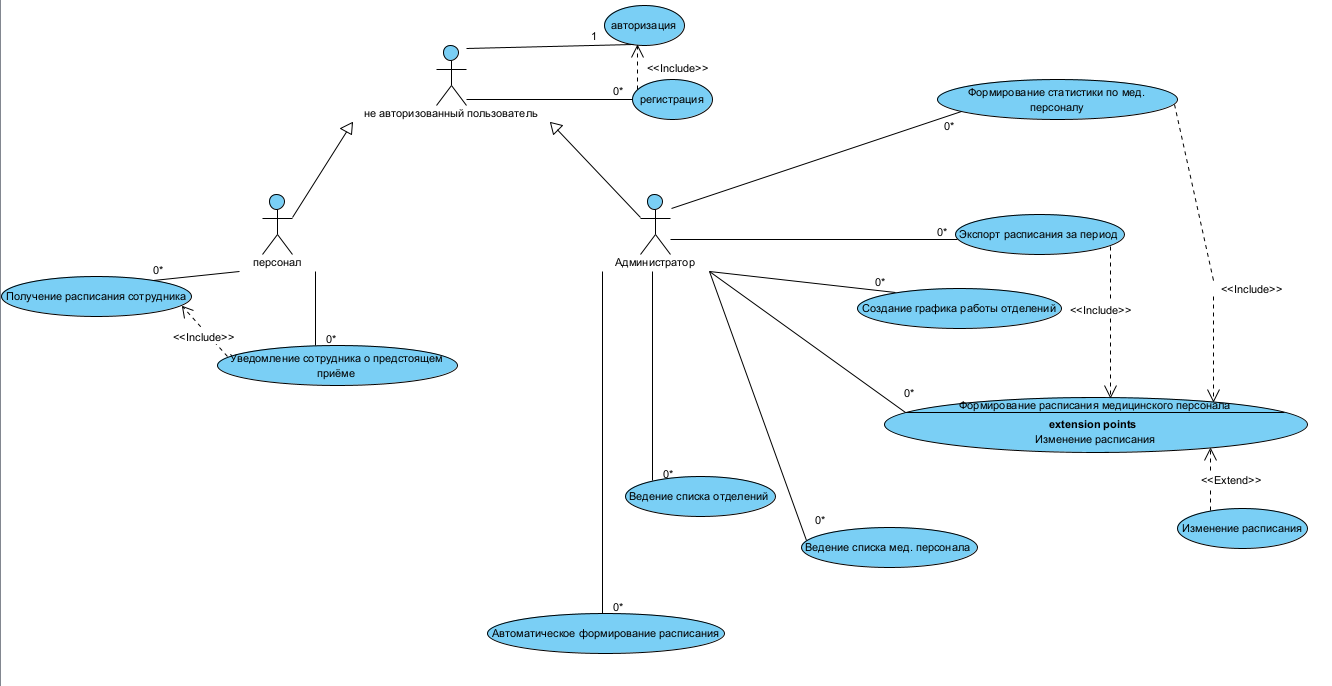


Рисунок 6 Диаграмма вариантов использования

**Модель классов (Class Diagram)**

Данная диаграмма демонстрирует общую структуру иерархии классов системы, их коопераций, атрибутов, методов, интерфейсов и взаимосвязей между ними.

Для начала рассмотрим пакет, содержащий модели

1. **Интерфейс model**
   * Содержит атрибут: id: int
   * Реализуется всеми основными классами системы
2. **Класс Рабочий**
   * Атрибуты:
     + должность: Должность
     + отделение: Отделение
     + предстоящие приёмы: Приём[]
     + ФИО: string
     + отпуски: Отпуск[]
     + пароль: string
     + login: string
   * Реализует интерфейс model
3. **Класс Должность**
   * Атрибут: name: string
   * Связан с Рабочим через ассоциацию
4. **Класс Отделение**
   * Атрибуты:
     + персонал: Рабочий[]
     + график работы на день: ГрафикРаботыНаДень[]
   * Реализует интерфейс model
   * Связано с Рабочим и ГрафикомРаботыНаДень (композиция)
5. **Класс ГрафикРаботыНаДень**
   * Атрибуты:
     + начало работы: time
     + конец работы: time
     + работает: bool
   * Реализует интерфейс model
6. **Класс Приём**
   * Атрибуты:
     + время: date
     + необходимый персонал: Должность[]
     + персонал: Рабочий[]
     + отделение: Отделение
   * Реализует интерфейс model
7. **Класс Отпуск**
   * Атрибут: дни: Выходной[]
   * Реализует интерфейс model
   * Связан с Рабочим через композицию
8. **Интерфейс ВыходнойДень**
   * Атрибут: день: date
   * Реализуется классами Выходной и Праздник
9. **Класс Выходной**
   * Реализует интерфейс ВыходнойДень
10. **Класс Праздник**
    * Атрибут: название: string
    * Реализует интерфейс ВыходнойДень

**Основные связи:**

* Рабочий связан с Должностью (ассоциация)
* Рабочий связан с Отделением (ассоциация)
* Рабочий связан с Приёмами (ассоциация)
* Рабочий связан с Отпусками (композиция)
* Отделение содержит ГрафикиРаботы (композиция)
* Отпуск содержит Выходные дни (композиция)
* Приём связан с Отделением (ассоциация)

Все основные классы реализуют интерфейс model, который определяет общий идентификатор. Интерфейс ВыходнойДень используется для представления дней отпуска и праздников.

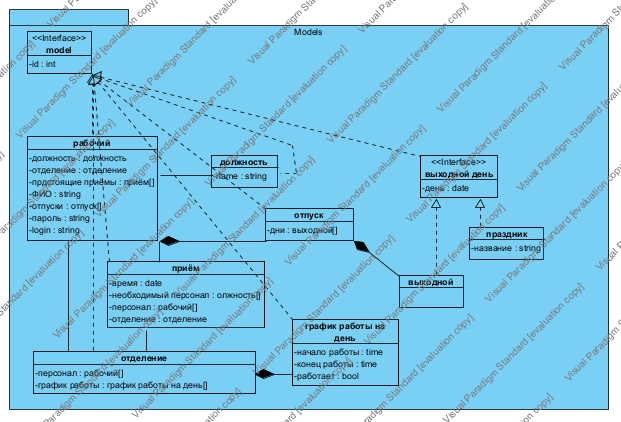


Рисунок 7 диаграмма классов пакет моделей

1. **Интерфейс**repository
   * Определяет базовые операции для работы с хранилищем:
     + create(object: model) – создание объекта
     + read(): model[] – чтение списка объектов
     + update(objectBefore: model, objectAfter: model) – обновление объекта
     + delete(object: model) – удаление объекта
   * Реализуется всеми репозиториями системы
2. **Класс**РабочийRepository
   * Реализует интерфейс repository
   * Предоставляет CRUD-операции для сущности **Рабочий**
3. **Класс**ПриёмRepository
   * Реализует интерфейс repository
   * Предоставляет CRUD-операции для сущности **Приём**
4. **Класс**ДолжностьRepository
   * Реализует интерфейс repository
   * Предоставляет CRUD-операции для сущности **Должность**
5. **Класс**ОтделениеRepository
   * Реализует интерфейс repository
   * Предоставляет CRUD-операции для сущности **Отделение**
6. **Класс**ПраздникRepository
   * Реализует интерфейс repository
   * Предоставляет CRUD-операции для сущности **Праздник**

**Основные связи:**

* Все репозитории (РабочийRepository, ПриёмRepository, ДолжностьRepository, ОтделениеRepository, ПраздникRepository) реализуют интерфейс repository.
* Интерфейс repository работает с базовым типом model, который является родительским для всех сущностей системы.

Данная диаграмма представляет **паттерн "Репозиторий"**, который обеспечивает абстракцию доступа к данным и позволяет централизованно управлять операциями CRUD (Create, Read, Update, Delete) для всех сущностей системы.

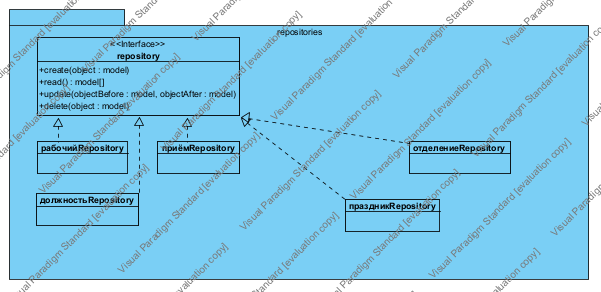


Рисунок 8 диаграмма классов пакет хранилища

Диаграмма классов пакета "Api" представляет собой структуру, состоящую из нескольких классов и одного базового класса ControllerBase. Все классы реализуют интерфейс или абстрактный класс (предположительно), связанный с различными функциями ввода данных.

Класс ControllerBase :

* + Это абстрактный базовый класс, от которого наследуются другие классы.
  + Он реализует несколько интерфейсов, каждый из которых связан с конкретной областью функциональности.

**Классы и их описание**

**1.** ControllerBase

* Роль : Базовый класс для всех контроллеров.
* Связь : Реализует следующие интерфейсы:
  + ведение праздников
  + ведение отделений
  + ведение должностей
  + ведение приёмов
  + ведение персонала

Это означает, что ControllerBase служит общей точкой расширения для реализации функциональности каждого из этих модулей.

**2.** ведение праздников (Class: праздники)

* Функционал : Управление данными о праздниках.
* Связь : Наследуется от ControllerBase.

**3.** ведение отделений (Class: отделения)

* Функционал : Управление информацией об отделениях.
* Связь : Наследуется от ControllerBase.

**4.** ведение должностей (Class: должности)

* Функционал : Управление должностями сотрудников.
* Связь : Наследуется от ControllerBase.

**5.** ведение приёмов (Class: приёмы)

* Функционал : Управление записью на приёмы к врачам.
* Связь : Наследуется от ControllerBase.

**6.** ведение персонала (Class: персонал)

* Функционал : Управление данными о сотрудниках.
* Связь : Наследуется от ControllerBase.

Диаграмма показывает иерархию классов, где все функциональные модули (ведение праздников, ведение отделений, ведение должностей, ведение приёмов, ведение персонала) наследуются от общего базового класса ControllerBase. Это позволяет унифицировать подход к реализации различных аспектов системы, обеспечивая повторное использование кода и логическую организацию API.

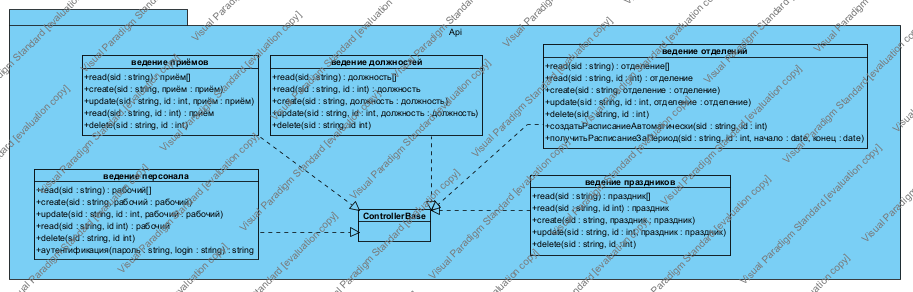


Рисунок 9 диаграмма классов пакет АПИ

Диаграмма классов пакета "Десктоп приложения" (рисунок 10) описывает структуру пользовательского интерфейса программы, связанного с управлением различными аспектами организации. Основу этой диаграммы составляет абстрактный класс Form, который реализует функциональность, общую для всех форм в приложении. От этого базового класса наследуются конкретные формы, каждая из которых отвечает за определённую задачу.

**Основные элементы диаграммы:**

Класс Form

* Назначение : Абстрактный класс, предоставляющий общую функциональность и структуру для всех форм.
* Реализации :
  + СформаАутентификации — форма входа в систему.
  + СформаУправленияПриёмами — форма управления записями о приёме пациентов/клиентов.
  + СформаОтчётов — форма формирования и просмотра отчётов.
  + СформаУправленияПерсоналом — форма управления данными сотрудников.
  + СформаУправленияПраздниками — форма управления событиями и праздниками.
  + СформаУправленияДолжностями — форма управления должностями сотрудников.
  + СформаУправленияОтделениями — форма управления подразделениями компании.

Формы

1. СформаАутентификации
   * Назначение : Форма авторизации пользователя в системе.
   * Атрибуты :
     + sid: string — идентификатор сессии (приватное поле).
   * Функционал : Предоставляет поля для ввода логина и пароля, обработку данных и проверку прав доступа.
2. СформаУправленияПриёмами
   * Назначение : Управление записями о приёме пациентов или клиентов.
   * Особенности : Позволяет добавлять, редактировать и удалять записи о приёмах.
3. СформаОтчётов
   * Назначение : Генерация и просмотр различных отчётных документов.
   * Особенности : Поддерживает фильтрацию данных, экспорт отчётов в разные форматы.
4. СформаУправленияПерсоналом
   * Назначение : Работа с данными сотрудников: добавление, удаление, редактирование информации.
5. СформаУправленияПраздниками
   * Назначение : Управление календарём праздников и событий.
   * Особенности : Добавление, редактирование и удаление событий.
6. СформаУправленияДолжностями
   * Назначение : Управление должностями сотрудников.
   * Особенности : Настройка названий должностей, уровней доступа, окладов.
7. СформаУправленияОтделениями
   * Назначение : Управление отделами или подразделениями компании.
   * Особенности : Добавление, редактирование и удаление отделений.

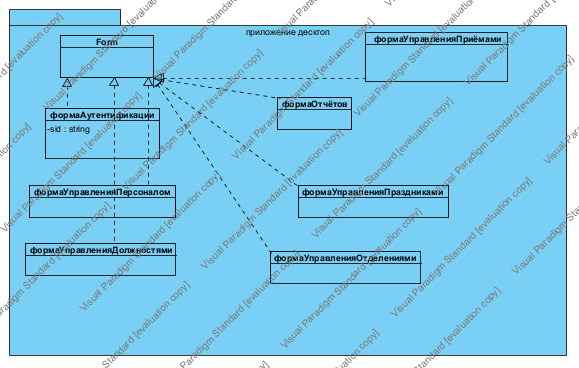


Рисунок 10 диаграмма классов пакет Десктоп приложения

На диаграмме классов изображена структура пакета Telegram-бота, который включает два основных класса:

1. Класс "аутентификация" (authentication)

* **Атрибуты**:
  + sid (тип: string, видимость: private) - идентификатор сессии
* **Операции**:
  + аутентификация (видимость: public) - метод для выполнения процесса аутентификации

2. Класс "уведомлятьель" (notifier)

* Не содержит явно указанных атрибутов и операций на диаграмме

Связи между классами

* Между классами существует зависимость (Dependency):
  + Класс "уведомлятьель" зависит от атрибута sid класса "аутентификация"

Визуальное представление

* Оба класса находятся внутри пакета "тгБот" (Telegram Bot)
* Классы отображены в виде прямоугольников с разделением на секции (имя класса, атрибуты, операции)
* Цвет фона классов и пакета - голубой (rgb(122, 207, 245))

Диаграмма показывает базовую структуру взаимодействия между компонентами аутентификации и уведомлений в Telegram-боте.

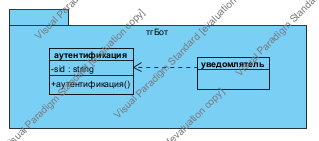


Рисунок 11 диаграмма классов пакет тг бота

Текстовое описание диаграммы классов (рисунок 12)

**Пакет:** android приложение

Классы:

1. аутентификация
   * **Атрибуты:**
     + - sid : string (приватное поле)
   * **Зависимости:**
     + Зависит от класса форма с расписанием (стрелка направлена от форма с расписанием к аутентификация).
2. форма с расписанием
   * **Зависимости:**
     + Использует функционал класса аутентификация.

Связи:

* Между классами существует **зависимость (Dependency)**:
  + Класс форма с расписанием зависит от класса аутентификация.

Стиль диаграммы:

* Классы отображены в прямоугольниках с голубым фоном (rgb(122, 207, 245)).
* Названия классов и атрибутов написаны стандартным шрифтом (Dialog, 11pt).
* Связь между классами показана пунктирной стрелкой с направлением от форма с расписанием к аутентификация.

Назначение:

Диаграмма отражает структуру Android-приложения, где:

* аутентификация отвечает за процесс проверки пользователя (содержит идентификатор сессии sid).
* форма с расписанием использует модуль аутентификации для доступа к данным.

Это упрощённая модель, которая может быть расширена дополнительными классами и связями.

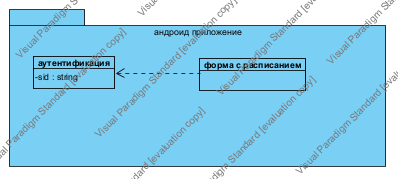


Рисунок 12 диаграмма классов пакет андроид приложения

**Модель последовательности действий (Sequence Diagram)**

Для описания жизненного цикла заявленной сквозной функциональности на основе диаграммы последовательности (Sequence Diagram), мы можем выделить основные этапы взаимодействия между участниками системы, представленными в виде жизненных линий (Lifelines). Диаграмма описывает процесс создания работника через API, включая валидацию данных, обработку запроса, логирование и обработку ошибок.

**Участники (Lifelines)**

1. форма создания работника — пользовательский интерфейс для заполнения данных нового сотрудника.
2. API должностей — компонент, управляющий созданием и валидацией данных должностей.
3. валидация запроса — служба проверки корректности входных данных.
4. логирование создания работника — модуль фиксации действий пользователя и системных событий.
5. ошибка данных — компонент, отвечающий за обработку и возврат ошибок при некорректном вводе данных.
6. база данных / репозиторий — слой хранения данных о работниках.

**Основные этапы жизненного цикла**

**1.** Инициализация запроса

* Пользователь заполняет форму создания работника.
* Отправляет данные через форму создания работника → отправляется сообщение "создать работника" в API должностей .

**2.** Валидация данных

* API должностей передает данные на проверку в валидацию запроса .
* Выполняется проверка:
  + Корректность формата данных.
  + Соблюдение бизнес-правил (например, уникальность ID).
* При успешной валидации:
  + Возвращается сообщение "валидация запроса" в API должностей .
* При неудачной валидации:
  + Возбуждается исключение или вызывается ошибка данных .
  + Отправляется сообщение "ошибка данных" обратно к форме или клиенту.

**3.** Логирование операции

* После успешной валидации система отправляет команду "логирование создания работника" в соответствующий модуль.
* Логируется:
  + Дата и время события.
  + Имя пользователя.
  + Тип операции ("Создание работника").
* Подтверждение логирования возвращается в API должностей .

**4.** Сохранение данных

* API должностей отправляет запрос в базу данных / репозиторий с командой "сохранить работника" .
* База данных выполняет операцию сохранения и возвращает результат:
  + Успех: "создано" .
  + Ошибка: "ошибка загрузки данных" .

**5.** Обработка результата

* Если данные успешно сохранены:
  + Отправляется подтверждение "работник создан" обратно в форму создания работника .
* Если произошла ошибка:
  + Вызывается ошибка данных .
  + Формируется ответ "ошибка создания" , который отправляется пользователю.

**Обработка ошибок**

* На любом этапе может быть вызвано исключение.
* Сообщения вроде "ошибка данных" , "ошибка создания" , "ошибка загрузки данных" сигнализируют о проблемах.
* Все ошибки могут быть дополнительно залогированы.
* В случае критической ошибки система может вернуть HTTP-код ошибки или уведомление о внутренней проблеме сервера.

**Завершение**

* По завершении всех этапов (успешно или с ошибкой) жизненная линия операции закрывается.
* Пользователь получает обратную связь о результате выполнения операции.

**Вывод**

Таким образом, жизненный цикл сквозной функциональности "Создание работника" включает следующие ключевые этапы:

1. Получение данных от клиента.
2. Валидация введенных данных.
3. Логирование действия.
4. Сохранение данных в базе.
5. Возврат результата пользователю.
6. Обработка возможных ошибок на каждом этапе.

Это позволяет обеспечить прозрачность, контролируемость и надежность выполнения бизнес-процесса.

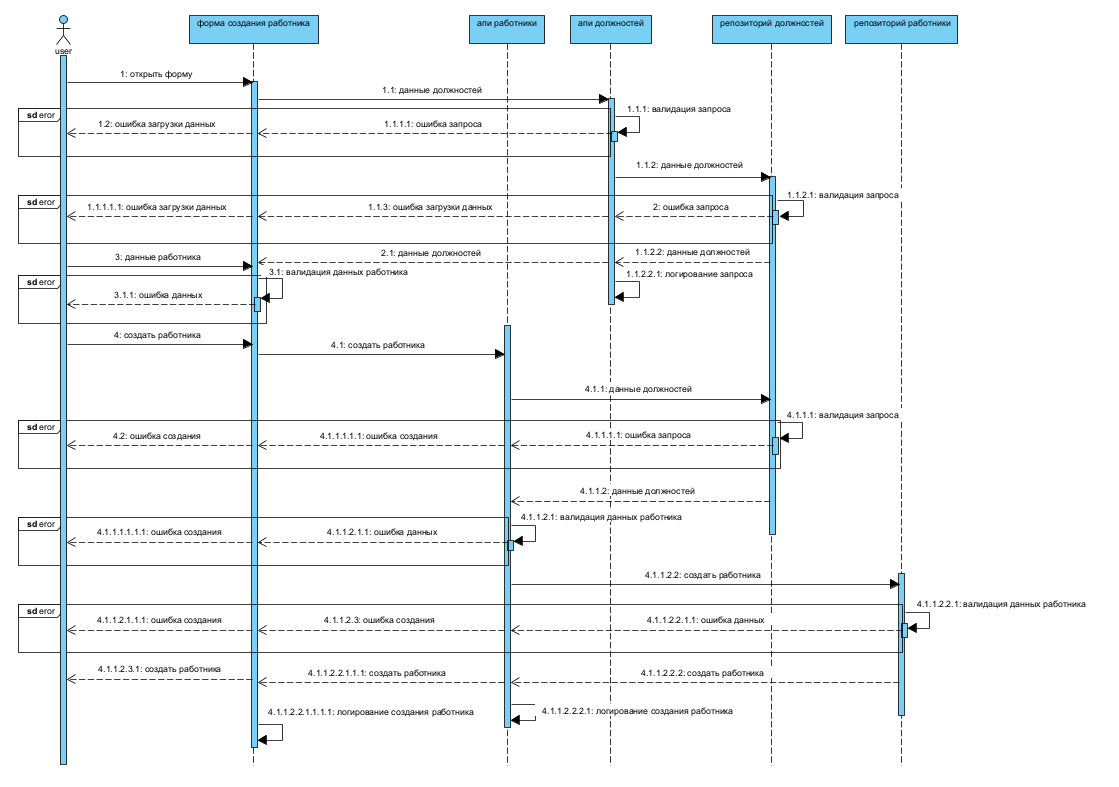


Рисунок 13 Sequence Diagram

**Модель состояния (State Diagram)**

Диаграмма состояний описывает жизненный цикл пользовательского интерфейса desktop-приложения, начиная с момента запуска и заканчивая выходом из системы. В ней представлены основные состояния приложения, а также переходы между ними в зависимости от действий пользователя.

Каждое состояние представляет собой экран или форму, на которой находится пользователь:

1. открыта форма авторизации
   * Начальное состояние.
   * Пользователь должен ввести логин и пароль для входа в систему.
   * Если данные введены неверно, происходит обновление формы с соответствующим уведомлением.
2. открыта главная форма
   * После успешного входа пользователь попадает на главную форму.
   * Здесь доступны кнопки для перехода к различным модулям управления:
     + Управление праздниками
     + Управление персоналом
     + Управление отделениями
     + Управление должностями
     + Формирование отчётов
3. открыта форма управления праздниками
   * Доступна после нажатия соответствующей кнопки на главной форме.
   * Предоставляет возможность добавления, удаления и редактирования праздников.
   * Кнопки:
     + "Добавить"
     + "Удалить"
4. открыта форма управления записями (приемами)
   * Аналогично предыдущему — позволяет управлять записью пациентов на прием.
5. открыта форма управления отделениями
6. открыта форма управления работниками
7. открыта форма управления должностями
8. открыта форма управления отчетами
   * Содержит две кнопки:
     + "Сформировать отчёт" — добавляет отчёт на форму.
     + "Экспорт в Excel" — выгружает отчёт в формате Excel.

**Переходы (Transitions)**

Переходы между состояниями происходят по следующим действиям:

**1.** Форма авторизации → Главная форма

* [нажатие кнопки войти]
  + При верных данных — открывается главная форма.
  + При неверных данных — форма обновляется с сообщением об ошибке.

**2.** Главная форма → Модульные формы

* [нажатие кнопки "управление праздниками"] → открыта форма управления праздниками
* [нажатие кнопки "управление персоналом"] → открыта форма управления работниками
* [нажатие кнопки "управление отделениями"] → открыта форма управления отделениями
* [нажатие кнопки "управление должностями"] → открыта форма управления должностями
* [нажатие кнопки "формирование отчётов"] → открыта форма управления отчетами

**3.** Модульные формы → Главная форма

* [нажатие кнопки "выйти"] — возврат на главную форму

**4.** Главная форма → Выход из приложения

* [нажатие кнопки "выход"] — завершение работы с приложением

**5.** Форма управления отчетами

* [нажатие кнопки "сформировать отчёт"] → отчёт добавляется на форму
* [нажатие кнопки "экспорт в excell"] → отчёт выгружается в Excel

**6.** Форма управления праздниками

* [нажатие кнопки "добавить"] → праздник добавляется в список
* [нажатие кнопки "удалить"] → праздник удаляется из списка

**Итоговая последовательность работы**

1. Запуск приложения → Открытие формы авторизации.
2. Ввод учетных данных → Передача на проверку.
3. Проверка логина/пароля :
   * Верные → Открытие главной формы.
   * Неверные → Обновление формы с ошибкой.
4. Работа на главной форме :
   * Выбор нужного раздела → Переход в соответствующую форму.
5. Выполнение операций в разделах :
   * Добавление/удаление/редактирование данных.
6. Формирование отчетов :
   * Создание отчета и его экспорт в Excel.
7. Завершение работы :
   * Выход из модулей → Выход из главной формы → Завершение программы.

Диаграмма состояний демонстрирует полный жизненный цикл взаимодействия пользователя с desktop-приложением. Она наглядно показывает, как пользователь перемещается между различными модулями системы, какие действия он может выполнять и как система реагирует на эти действия. Такая структура обеспечивает понятное и логичное поведение приложения, что способствует удобству использования и простоте поддержки кода.

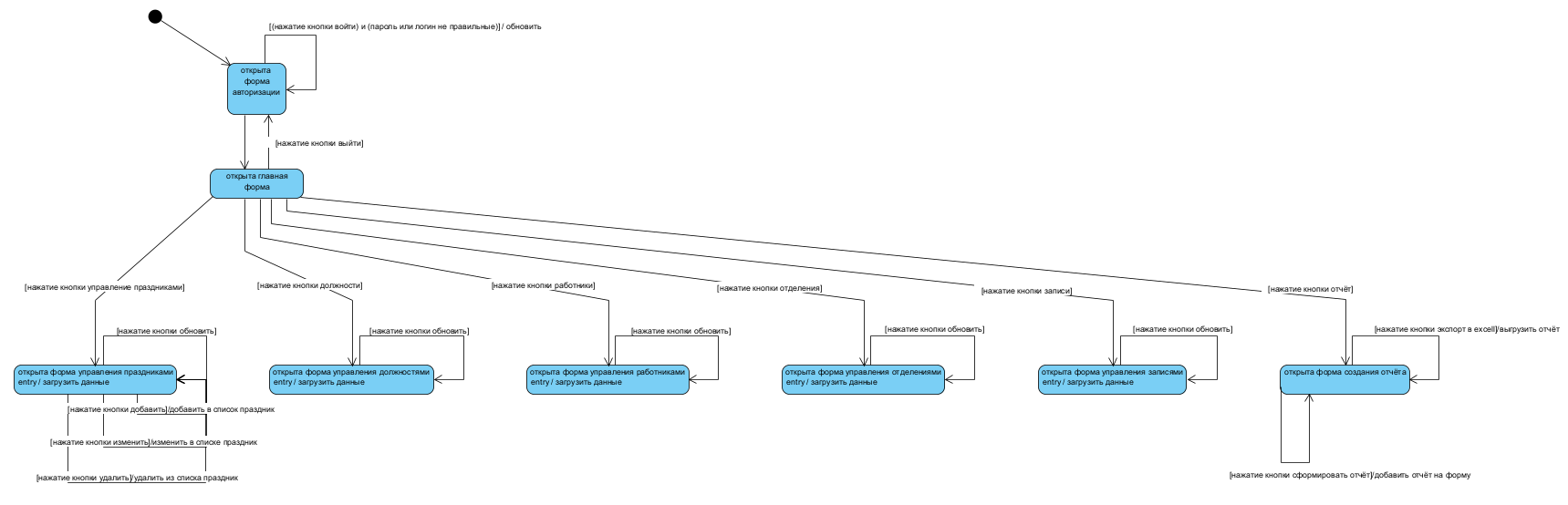


Рисунок 14 State diagram desctop приложения

Диаграмма состояний описывает жизненный цикл и поведение пользовательского интерфейса мобильного Android-приложения. Она включает основные экраны приложения, переходы между ними и реакции на действия пользователя.

**Состояния (States)**

Каждое состояние представляет собой отдельный экран или виджет (UI-компонент), на котором находится пользователь:

**1.** открыт виджет входа

* Начальное состояние.
* Пользователь вводит логин и пароль для авторизации.
* При успешном входе:
  + Переход к главной форме.
* При неверных данных:
  + Отображается сообщение об ошибке.

**2.** открыт виджет записей

* Доступен после успешного входа.
* Позволяет просматривать список записей пациентов.
* Возможные действия:
  + Обновление данных (свайп сверху вниз).
  + Выход из аккаунта.

Диаграмма состояний демонстрирует полный жизненный цикл взаимодействия пользователя с Android-приложением. Она показывает, как пользователь перемещается между различными экранами, какие действия он может выполнять и как система реагирует на эти действия. Такая структура обеспечивает понятное и логичное поведение мобильного приложения, что способствует удобству использования и простоте поддержки кода.

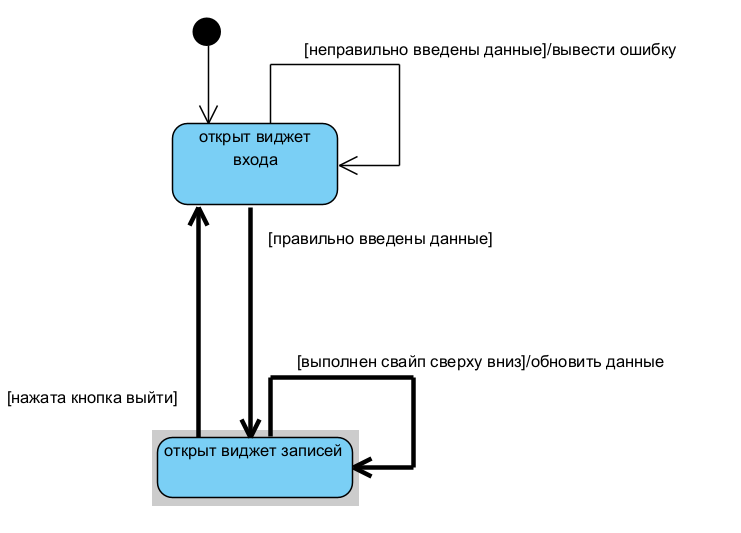


Рисунок 15 State diagram андроид приложения

Диаграмма Рисунок 16 отражает жизненный цикл и поведение Telegram-бота, включая основные состояния, в которых он может находиться, а также переходы между ними в ответ на внешние события или внутренние триггеры.

**Состояния:**

1. [Initial State] Начальное состояние
   * Представляет начальную точку работы бота.
   * Из этого состояния происходит первый переход к чтению конфигурации.
2. Чтение конфига
   * Бот загружает свои настройки (например, токен, параметры подключения, настройки уведомлений).
   * После завершения переходит к регистрации в API Telegram.
3. Регистрация в API Telegram
   * Бот устанавливает соединение с серверами Telegram и регистрирует свой webhook или начинает polling.
   * После успешной регистрации бот переходит в режим ожидания сообщений.
4. Ожидание вызова от API
   * Бот находится в состоянии ожидания входящих запросов от Telegram (например, команды пользователя, callback-запросы, новые сообщения).
   * В этом состоянии возможны два основных перехода:
     + Получение запроса от API → Отправка сообщения пользователю.
     + Таймер (прошло N времени без активности) → Остается в том же состоянии (циклический переход).
5. Отправка в API Telegram запроса на отправить пользователю сообщения о записи
   * При получении соответствующего запроса от API, бот формирует и отправляет сообщение пользователю через Telegram API.
   * После выполнения действия возвращается обратно в состояние ожидания.

**Переходы:**

* Чтение конфига → Регистрация в API Telegram  
  Автоматический переход после завершения загрузки конфигурации.
* Регистрация в API Telegram → Ожидание вызова от API  
  Переход после успешного подключения к Telegram API.
* Ожидание вызова от API → Отправка сообщения  
  Выполняется при получении запроса от API Telegram (например, команда /start или событие от пользователя).
* Отправка сообщения → Ожидание вызова от API  
  После отправки сообщения бот возвращается в режим ожидания новых запросов.
* Ожидание вызова от API → Ожидание вызова от API (самоперевод)  
  Циклический переход, активируемый по истечении определенного времени без внешних событий (таймер).

**Основные события:**

* [пришел запрос от API] — внешнее событие, запускающее обработку запроса от пользователя.
* [прошло N времени] — внутреннее событие таймера для проверки активности или выполнения периодических задач.

**Заключение:**

Диаграмма показывает последовательность действий Telegram-бота, начиная с инициализации и заканчивая обработкой входящих сообщений. Все состояния и переходы организованы логично, обеспечивая бесперебойное функционирование бота в рамках взаимодействия с Telegram API. Такая модель позволяет легко добавлять новые состояния и расширять логику работы бота.

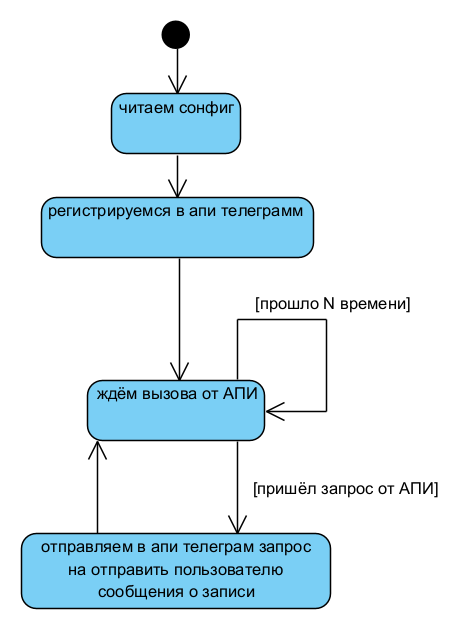


Рисунок 17 State diagram тг бот

**Модель компонентов (Component Diagram)**

Диаграмма компонентов (Component Diagram) представляет архитектуру программной системы в терминах взаимодействия между компонентами, она отражает структуру API-уровня приложения.

**Компоненты**

1. Program.cs
   * Точка входа в приложение.
   * Зависит от:
     + models.dll
     + Repositories.dll
     + microsoft.extensions
2. models.dll
   * Библиотека, содержащая определения классов предметной области (моделей).
   * Используется:
     + Program.cs
     + Repositories.dll
   * Стереотип: library
3. Repositories.dll
   * Библиотека, реализующая слой доступа к данным.
   * Зависит от:
     + models.dll
   * Используется:
     + Program.cs
   * Также зависит от:
     + EFCore (Entity Framework Core)
   * Стереотип: library
4. EFCore
   * Реализация Entity Framework Core — ORM для работы с базами данных.
   * Используется:
     + Repositories.dll
   * Зависит от:
     + PostgreSQL
   * Стереотип: library
5. microsoft.extensions
   * Библиотека расширений Microsoft, (внедрения зависимостей), логгирования и других служебных функций.
   * Используется:
     + Program.cs
   * Стереотип: library
6. PostgreSQL
   * Физический файл или драйвер, обеспечивающий подключение к СУБД PostgreSQL.
   * Используется:
     + EFCore
   * Стереотип: file

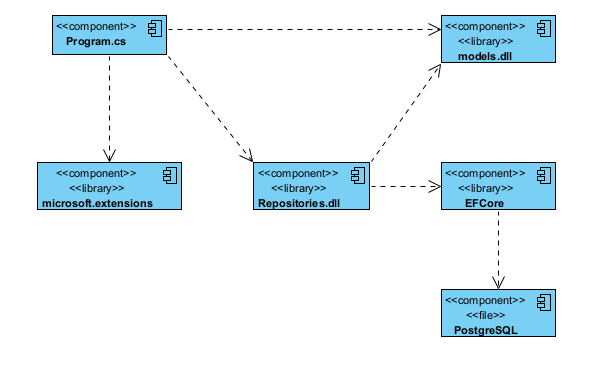
****

Рисунок 18 Component Diagram API

Диаграмма компонентов (Component Diagram) под названием "desctop" отражает архитектурную структуру настольного приложения. Она показывает, как различные программные модули взаимодействуют между собой, и какие зависимости существуют между ними.

**Компоненты**

1. program.cs
   * Точка входа в приложение.
   * Зависит от:
     + Models.dll
     + Microsoft.Extensions
2. Models.dll
   * Библиотека, содержащая определения классов предметной области (моделей).
   * Используется:
     + program.cs
   * Стереотип: library
3. Microsoft.Extensions
   * Библиотека расширений Microsoft(внедрения зависимостей), логгирования и других служебных функций.
   * Используется:
     + program.cs
   * Стереотип: library

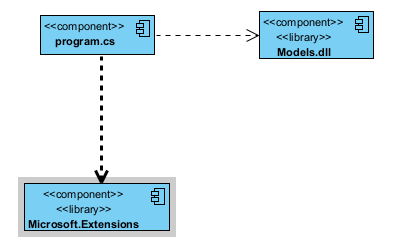


Рисунок 19 Component Diagram Desctop

Диаграмма компонентов (Component Diagram) под названием "Андроид Приложение" отражает архитектурную структуру мобильного приложения на платформе Android. Она показывает, как различные программные модули взаимодействуют между собой и какие зависимости существуют между ними.

**Компоненты**

1. flutterApp
   * Основное приложение Flutter.
   * Точка входа в пользовательский интерфейс и логику приложения.
   * Зависит от:
     + Google.Libraries
     + Java.Libraries
2. Google.Libraries
   * Библиотеки от Google, используемые в приложении.
   * Используется:
     + flutterApp
   * Стереотип: library
3. Java.Libraries
   * Встроенные или сторонние Java-библиотеки, необходимые для работы Android-приложения.
   * Используется:
     + flutterApp
   * Стереотип: library

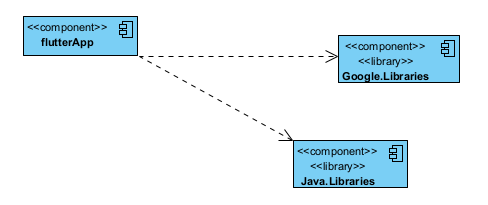


Рисунок 20 Component Diagram Андроид Приложение

Диаграмма компонентов (Component Diagram) под названием "tg bot" отражает архитектурную структуру Telegram-бота. Она показывает, как различные программные модули взаимодействуют между собой и какие зависимости существуют между ними.

**Компоненты**

1. telegram\_api
   * Представляет API Telegram, через которое бот взаимодействует с серверами Telegram.
   * Используется:
     + bot
   * Стереотип: library
2. bot
   * Основной компонент — логика самого Telegram-бота.
   * Точка входа в приложение.
   * Зависит от:
     + telegram\_api

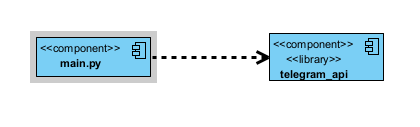


Рисунок 21 tg bot

**Модель развёртывания (Deployment Diagram)**

Диаграмма развертывания (Deployment Diagram) под названием "Deployment Diagram1" отражает физическую архитектуру системы. Она показывает, как программные элементы (артефакты) размещаются на аппаратных узлах и как они взаимодействуют между собой.

Это особенно важно для понимания того, где и как компоненты приложения будут развернуты в реальной инфраструктуре.

1. Telegram\_service
   * Представляет серверную часть Telegram.
   * Используется клиентами для обмена сообщениями с ботом.
2. API
   * Сервис API, через который происходит обмен данными между клиентами и базой данных.
3. PostgreSQL
   * База данных PostgreSQL, которая хранит данные приложения.
   * Связана с API.
4. telegram\_App
   * Клиентский мессенджер Telegram, установленный на устройстве пользователя.
   * Взаимодействует с Telegram\_service.
5. Desctop\_App
   * Настольное приложение, которое может взаимодействовать с API.
6. Moblie\_APP
   * Мобильное приложение Flutter, использующее Google и Java библиотеки.
   * Взаимодействует с API.
7. user
   * Пользователь, взаимодействующий с системой через клиентские приложения.

Telegram\_service, API, PostgreSQL разворачиваются на сервере, а telegram\_App, Desctop\_App, Moblie\_APP устанавливаются пользователем на собственное устройство.

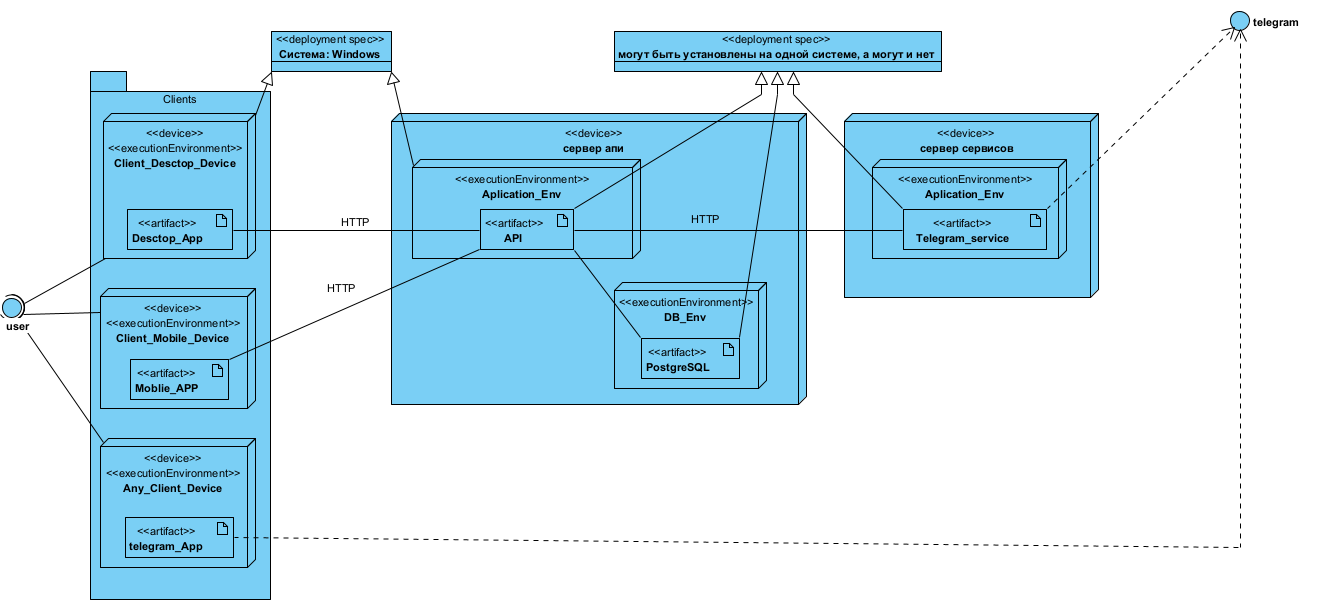


Рисунок 22 Deployment Diagram

**Генерация кода по проекту**

Генерация кода была выполнена. Пример генерации кода:

// ==== D:\piaps\generated\GeneratedSolution\Api\Program.cs ====

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

// ==== D:\piaps\generated\GeneratedSolution\Api\Controllers\ведение\_должностей.cs ====

using Microsoft.AspNetCore.Mvc;

using System;

namespace Api

{

[ApiController]

[Route("api/dol")]

public class \_ : ControllerBase {

[HttpGet("get-all")]

public Models.Models.[] Read() {

throw new System.NotImplementedException("Not implemented");

}

[HttpGet("get/{id}")]

public Models.Models. Read(int id) {

throw new System.NotImplementedException("Not implemented");

}

[HttpPost("add")]

public void Create([FromBody] Models.Models. ) {

throw new System.NotImplementedException("Not implemented");

}

[HttpPut("update/{id}")]

public void Update(int id, [FromBody] Models.Models. ) {

throw new System.NotImplementedException("Not implemented");

}

[HttpDelete("delete/{id}")]

public void Delete(int id) {

throw new System.NotImplementedException("Not implemented");

}

private repositories.Repository Repository;

private ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Api\Controllers\ведение\_отделений.cs ====

using Microsoft.AspNetCore.Mvc;

using System;

namespace Api

{

[ApiController]

[Route("api/otd")]

public class \_ : ControllerBase

{

[HttpGet("get-all")]

public Models.Models.[] Read() {

throw new System.NotImplementedException("Not implemented");

}

[HttpGet("get/{id}")]

public Models.Models. Read(int id) {

throw new System.NotImplementedException("Not implemented");

}

[HttpPost("add")]

public void Create([FromBody] Models.Models. ) {

throw new System.NotImplementedException("Not implemented");

}

[HttpPut("update/{id}")]

public void Update(int id, [FromBody] Models.Models. ) {

throw new System.NotImplementedException("Not implemented");

}

[HttpDelete("delete/{id}")]

public void Delete(int id) {

throw new System.NotImplementedException("Not implemented");

}

[HttpGet("createrasp/{id}")]

public void (int id) {

throw new System.NotImplementedException("Not implemented");

}

[HttpGet("rasp/{id}")]

public void (int id, DateOnly , DateOnly ) {

throw new System.NotImplementedException("Not implemented");

}

private repositories.Repository Repository;

private ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Api\Controllers\ведение\_персонала.cs ====

using Microsoft.AspNetCore.Mvc;

using System;

namespace Api

{

[ApiController]

[Route("api/per")]

public class \_ : ControllerBase

{

[HttpGet("get-all")]

public Models.Models.[] Read() {

throw new System.NotImplementedException("Not implemented");

}

[HttpPost("add")]

public void Create([FromBody] Models.Models. ) {

throw new System.NotImplementedException("Not implemented");

}

[HttpPut("update/{id}")]

public void Update(int id, [FromBody] Models.Models. ) {

throw new System.NotImplementedException("Not implemented");

}

[HttpGet("get/{id}")]

public Models.Models. Read( int id) {

throw new System.NotImplementedException("Not implemented");

}

[HttpDelete("delete/{id}")]

public void Delete( int id) {

throw new System.NotImplementedException("Not implemented");

}

public string ( string , string login) {

throw new System.NotImplementedException("Not implemented");

}

private repositories.Repository Repository;

private ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Api\Controllers\ведение\_праздников.cs ====

using Microsoft.AspNetCore.Mvc;

using System;

namespace Api

{

[ApiController]

[Route("api/pra")]

public class \_ : ControllerBase

{

[HttpGet("get-all")]

public Models.Models.[] Read() {

throw new System.NotImplementedException("Not implemented");

}

[HttpGet("get/{id}")]

public Models.Models. Read( int id) {

throw new System.NotImplementedException("Not implemented");

}

[HttpPost("add")]

public void Create([FromBody] Models.Models. ) {

throw new System.NotImplementedException("Not implemented");

}

[HttpPut("update/{id}")]

public void Update( int id, [FromBody] Models.Models. ) {

throw new System.NotImplementedException("Not implemented");

}

[HttpDelete("delete/{id}")]

public void Delete( int id) {

throw new System.NotImplementedException("Not implemented");

}

private repositories.Repository Repository;

private ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Api\Controllers\ведение\_приёмов.cs ====

using Microsoft.AspNetCore.Mvc;

using System;

namespace Api

{

[ApiController]

[Route("api/pri")]

public class \_ : ControllerBase

{

[HttpGet("get-all")]

public Models.Models.[] Read() {

throw new System.NotImplementedException("Not implemented");

}

[HttpPost("add")]

public void Create([FromBody] Models.Models. ) {

throw new System.NotImplementedException("Not implemented");

}

[HttpPut("update/{id}")]

public void Update(int id, [FromBody] Models.Models. ) {

throw new System.NotImplementedException("Not implemented");

}

[HttpGet("get/{id}")]

public Models.Models. Read(int id) {

throw new System.NotImplementedException("Not implemented");

}

[HttpDelete("delete/{id}")]

public void Delete(int id) {

throw new System.NotImplementedException("Not implemented");

}

private repositories.Repository Repository;

private ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Api\obj\Debug\net8.0\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\generated\GeneratedSolution\Api\obj\Debug\net8.0\Api.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("Api")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

[assembly: System.Reflection.AssemblyProductAttribute("Api")]

[assembly: System.Reflection.AssemblyTitleAttribute("Api")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\generated\GeneratedSolution\Api\obj\Debug\net8.0\Api.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::Microsoft.AspNetCore.Builder;

global using global::Microsoft.AspNetCore.Hosting;

global using global::Microsoft.AspNetCore.Http;

global using global::Microsoft.AspNetCore.Routing;

global using global::Microsoft.Extensions.Configuration;

global using global::Microsoft.Extensions.DependencyInjection;

global using global::Microsoft.Extensions.Hosting;

global using global::Microsoft.Extensions.Logging;

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Net.Http.Json;

global using global::System.Threading;

global using global::System.Threading.Tasks;

// ==== D:\piaps\generated\GeneratedSolution\Loger\логер.cs ====

using System;

public class {

public void Debag(ref string text) {

throw new System.NotImplementedException("Not implemented");

}

public void Info(ref string text) {

throw new System.NotImplementedException("Not implemented");

}

}

// ==== D:\piaps\generated\GeneratedSolution\Loger\obj\Debug\net8.0\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\generated\GeneratedSolution\Loger\obj\Debug\net8.0\Loger.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("Loger")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

[assembly: System.Reflection.AssemblyProductAttribute("Loger")]

[assembly: System.Reflection.AssemblyTitleAttribute("Loger")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\generated\GeneratedSolution\Loger\obj\Debug\net8.0\Loger.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

// ==== D:\piaps\generated\GeneratedSolution\Models\model.cs ====

using System;

namespace Models {

public interface Model {

}

}

// ==== D:\piaps\generated\GeneratedSolution\Models\выходной.cs ====

using System;

namespace Models.Models

{

public class : \_

{

public DateOnly ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Models\выходной\_день.cs ====

using System;

namespace Models.Models

{

public interface \_ : Model

{

}

}

// ==== D:\piaps\generated\GeneratedSolution\Models\график\_работы\_на\_день.cs ====

using System;

namespace Models.Models

{

public class \_\_\_ : Model

{

private TimeOnly \_;

private TimeOnly \_;

private bool ;

public int id;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Models\должность.cs ====

using System;

namespace Models.Models

{

public class

{

private string name;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Models\отделение.cs ====

using System;

namespace Models.Models

{

public class : Model

{

private [] ;

private \_\_\_[] \_;

public int id;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Models\отпуск.cs ====

using System;

namespace Models.Models

{

public class : Model

{

private [] ;

public int id;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Models\праздник.cs ====

using System;

namespace Models.Models

{

public class : \_

{

private string ;

public DateOnly ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Models\приём.cs ====

using System;

namespace Models.Models

{

public class : Model

{

private DateOnly ;

private [] \_;

private [] ;

private ;

public int id;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Models\рабочий.cs ====

using System;

namespace Models.Models

{

public class : Model

{

private ;

private ;

private [] \_;

private string ;

private [] ;

private string ;

private string login;

public int id;

}

}

// ==== D:\piaps\generated\GeneratedSolution\Models\obj\Debug\net8.0\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\generated\GeneratedSolution\Models\obj\Debug\net8.0\Models.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("Models")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

[assembly: System.Reflection.AssemblyProductAttribute("Models")]

[assembly: System.Reflection.AssemblyTitleAttribute("Models")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\generated\GeneratedSolution\Models\obj\Debug\net8.0\Models.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

// ==== D:\piaps\generated\GeneratedSolution\repositories\repository.cs ====

using System;

namespace repositories {

public interface Repository {

void Create(ref Models.Model object\_);

Models.Model[] Read();

void Update(ref Models.Model objectBefore, ref Models.Model objectAfter);

void Delete(ref Models.Model object\_);

}

}

// ==== D:\piaps\generated\GeneratedSolution\repositories\должностьRepository.cs ====

using System;

namespace repositories {

public class Repository : Repository {

public void Delete(ref Models.Model object\_) {

throw new System.NotImplementedException("Not implemented");

}

public void Update(ref Models.Model objectBefore, ref Models.Model objectAfter) {

throw new System.NotImplementedException("Not implemented");

}

public Models.Model[] Read() {

throw new System.NotImplementedException("Not implemented");

}

public void Create(ref Models.Model object\_) {

throw new System.NotImplementedException("Not implemented");

}

private Models.Models. ;

private ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\repositories\отделениеRepository.cs ====

using System;

namespace repositories {

public class Repository : Repository {

public void Delete(ref Models.Model object\_) {

throw new System.NotImplementedException("Not implemented");

}

public void Update(ref Models.Model objectBefore, ref Models.Model objectAfter) {

throw new System.NotImplementedException("Not implemented");

}

public Models.Model[] Read() {

throw new System.NotImplementedException("Not implemented");

}

public void Create(ref Models.Model object\_) {

throw new System.NotImplementedException("Not implemented");

}

private Models.Models. ;

private ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\repositories\праздникRepository.cs ====

using System;

namespace repositories {

public class Repository : Repository {

public void Delete(ref Models.Model object\_) {

throw new System.NotImplementedException("Not implemented");

}

public void Update(ref Models.Model objectBefore, ref Models.Model objectAfter) {

throw new System.NotImplementedException("Not implemented");

}

public Models.Model[] Read() {

throw new System.NotImplementedException("Not implemented");

}

public void Create(ref Models.Model object\_) {

throw new System.NotImplementedException("Not implemented");

}

private Models.Models. ;

private ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\repositories\приёмRepository.cs ====

using System;

namespace repositories {

public class Repository : Repository {

public void Delete(ref Models.Model object\_) {

throw new System.NotImplementedException("Not implemented");

}

public void Update(ref Models.Model objectBefore, ref Models.Model objectAfter) {

throw new System.NotImplementedException("Not implemented");

}

public Models.Model[] Read() {

throw new System.NotImplementedException("Not implemented");

}

public void Create(ref Models.Model object\_) {

throw new System.NotImplementedException("Not implemented");

}

private Models.Models. ;

private ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\repositories\рабочийRepository.cs ====

using System;

namespace repositories {

public class Repository : Repository {

public void Delete(ref Models.Model object\_) {

throw new System.NotImplementedException("Not implemented");

}

public void Update(ref Models.Model objectBefore, ref Models.Model objectAfter) {

throw new System.NotImplementedException("Not implemented");

}

public Models.Model[] Read() {

throw new System.NotImplementedException("Not implemented");

}

public void Create(ref Models.Model object\_) {

throw new System.NotImplementedException("Not implemented");

}

private Models.Models. ;

private ;

}

}

// ==== D:\piaps\generated\GeneratedSolution\repositories\obj\Debug\net8.0\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\generated\GeneratedSolution\repositories\obj\Debug\net8.0\repositories.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("repositories")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

[assembly: System.Reflection.AssemblyProductAttribute("repositories")]

[assembly: System.Reflection.AssemblyTitleAttribute("repositories")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\generated\GeneratedSolution\repositories\obj\Debug\net8.0\repositories.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\Program.cs ====

namespace приложение\_десктоп

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

// To customize application configuration such as set high DPI settings or default font,

// see https://aka.ms/applicationconfiguration.

ApplicationConfiguration.Initialize();

Application.Run(new ФормаАутентификации());

}

}

}

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\формаАутентификации.cs ====

using System;

namespace \_ {

public class : Form {

private string sid;

}

}

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\формаОтчётов.cs ====

using System;

namespace \_ {

public class : Form {

}

}

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\формаУправленияДолжностями.cs ====

using System;

namespace \_ {

public class : Form {

}

}

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\формаУправленияОтделениями.cs ====

using System;

namespace \_ {

public class : Form {

}

}

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\формаУправленияПерсоналом.cs ====

using System;

namespace \_ {

public class : Form {

}

}

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\формаУправленияПраздниками.cs ====

using System;

namespace \_ {

public class : Form {

}

}

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\формаУправленияПриёмами.cs ====

using System;

namespace \_ {

public class : Form {

}

}

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\obj\Debug\net8.0-windows\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\obj\Debug\net8.0-windows\приложение десктоп.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("приложение десктоп")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

[assembly: System.Reflection.AssemblyProductAttribute("приложение десктоп")]

[assembly: System.Reflection.AssemblyTitleAttribute("приложение десктоп")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

[assembly: System.Runtime.Versioning.TargetPlatformAttribute("Windows7.0")]

[assembly: System.Runtime.Versioning.SupportedOSPlatformAttribute("Windows7.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\generated\GeneratedSolution\приложение десктоп\obj\Debug\net8.0-windows\приложение десктоп.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::System;

global using global::System.Collections.Generic;

global using global::System.Drawing;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

global using global::System.Windows.Forms;

Отличие сгенерированного кода по диаграмме классов и кодом в программе при реализации заключается в:

1. В сгенерированном коде зависимости между классами прописаны жестко.
2. В сгенерированном коде показаны поля класса, а нужно было свойства для облегчения доступа к данным.
3. В интерфейсах нет свойств
4. Не реализована ни одна функция
5. Названия на русском

# Реализация приложения

**Описание технологий разработки приложения**

Языки программирования – C#, Python, Dart

Используемая СУБД – PostgreSQL Для взаимодействия с СУБД средствами языка программирования используется библиотека EntityFrameworkCore.PostgreSQL Для автоматического управления SQL-миграциями используется менеджер миграций из библиотеки EntityFrameworkCore

Для логгирования средствами языка программирования C# с использование библиотеки Serilog.AspNetCore

Для генерации отчётов в формате EXCEL используется библиотека ClosedXML

**Демонстрация приложения**

В качестве демонстрации работы приложения приводится основной сценарий работы по контрольному примеру.

Описание основных экранов программы. На рис. 23 - 29 предоставлен основной сценарий работы с desctop приложением

На рисунке 23 форма вода, надо ввести логин и пароль сотрудника, что бы войти.

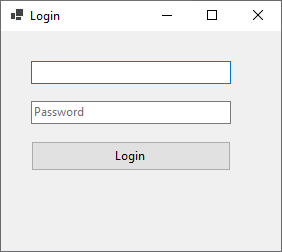
****

Рисунок 23 desctop app форма авторизации

На освновной форме главного меню можно открывать разные разделы приложения

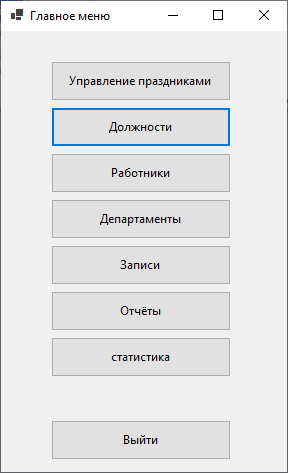
****

Рисунок 24 desctop app основное меню

Форма для управления приёмами, можно редактировать, добавлять удалять приёмы.

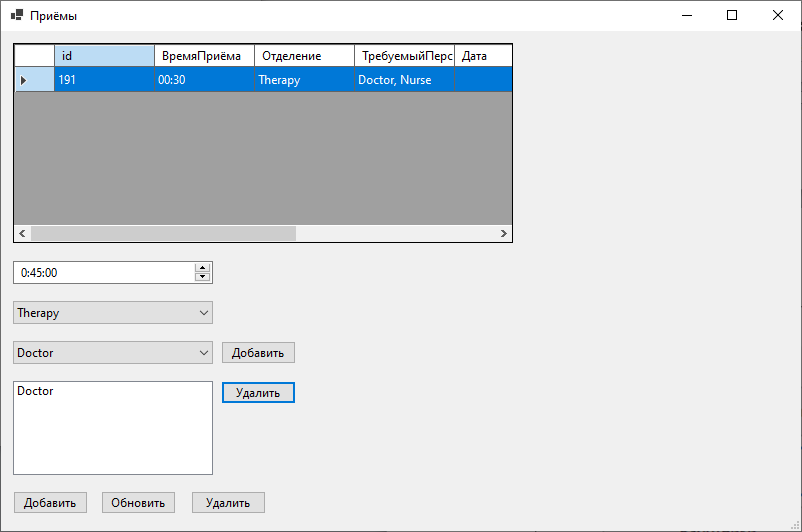
****

Рисунок 25 desctop app управление приёмами

При успешном добавлении приёма будет всплывашка

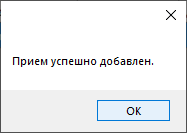
****

Рисунок 26 desctop app приём успешно добавлен

Форма для работы с отделениями позволяет добавлять изменять удалять отделения а также работать с расписанием отделения.

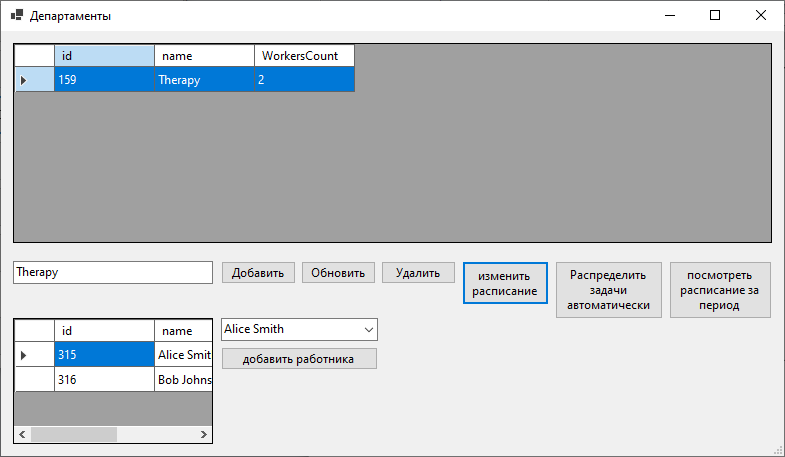
****

Рисунок 27 desctop app упраление отделениями

Форма с расписанием сотрудников отделения, здесь же можно его и редактировать.

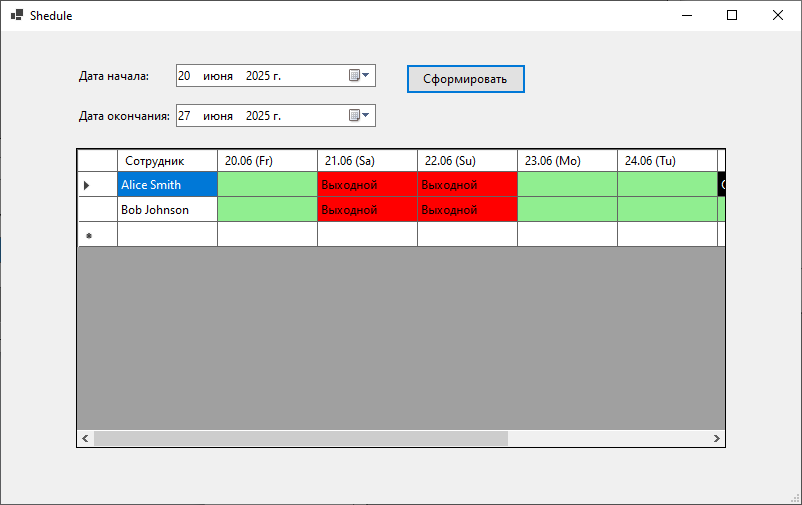
****

Рисунок 28 desctop app посмотреть расписание сотрудников отделения

Форма с статистикой по сотрудникам, моно выбирать даты и делать статистику по отделениям.

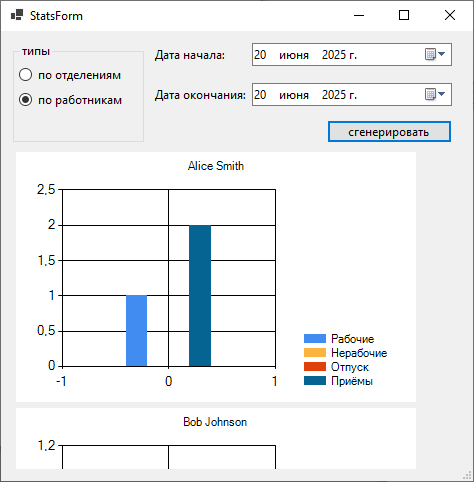
****

Рисунок 29 desctop app статистика по работникам

В окне авторизации в мобильном приложении надо ввести правильные логин пароль и тогда вы перейдёте на окно с расписанием

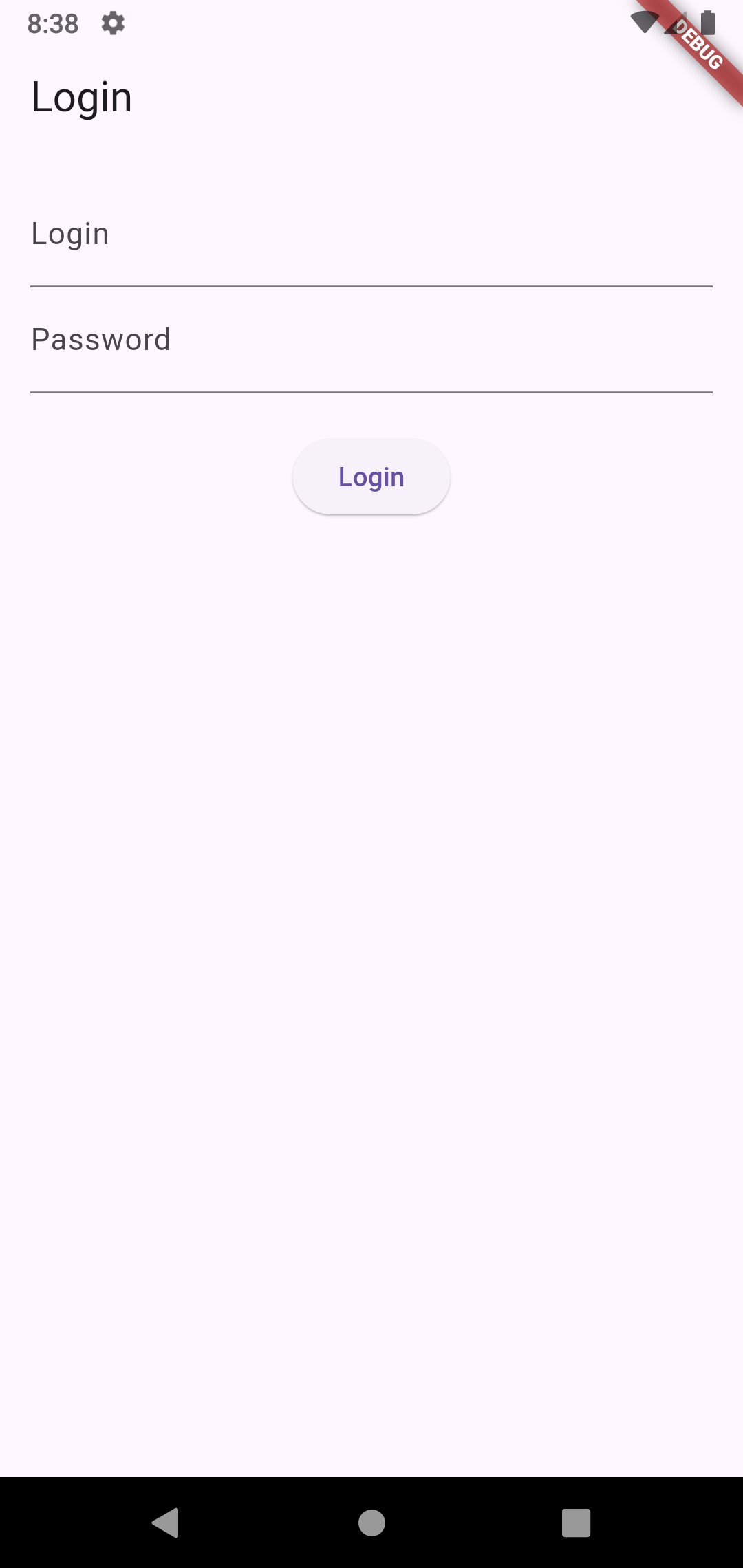


Рисунок 30 Android приложение виджет входа

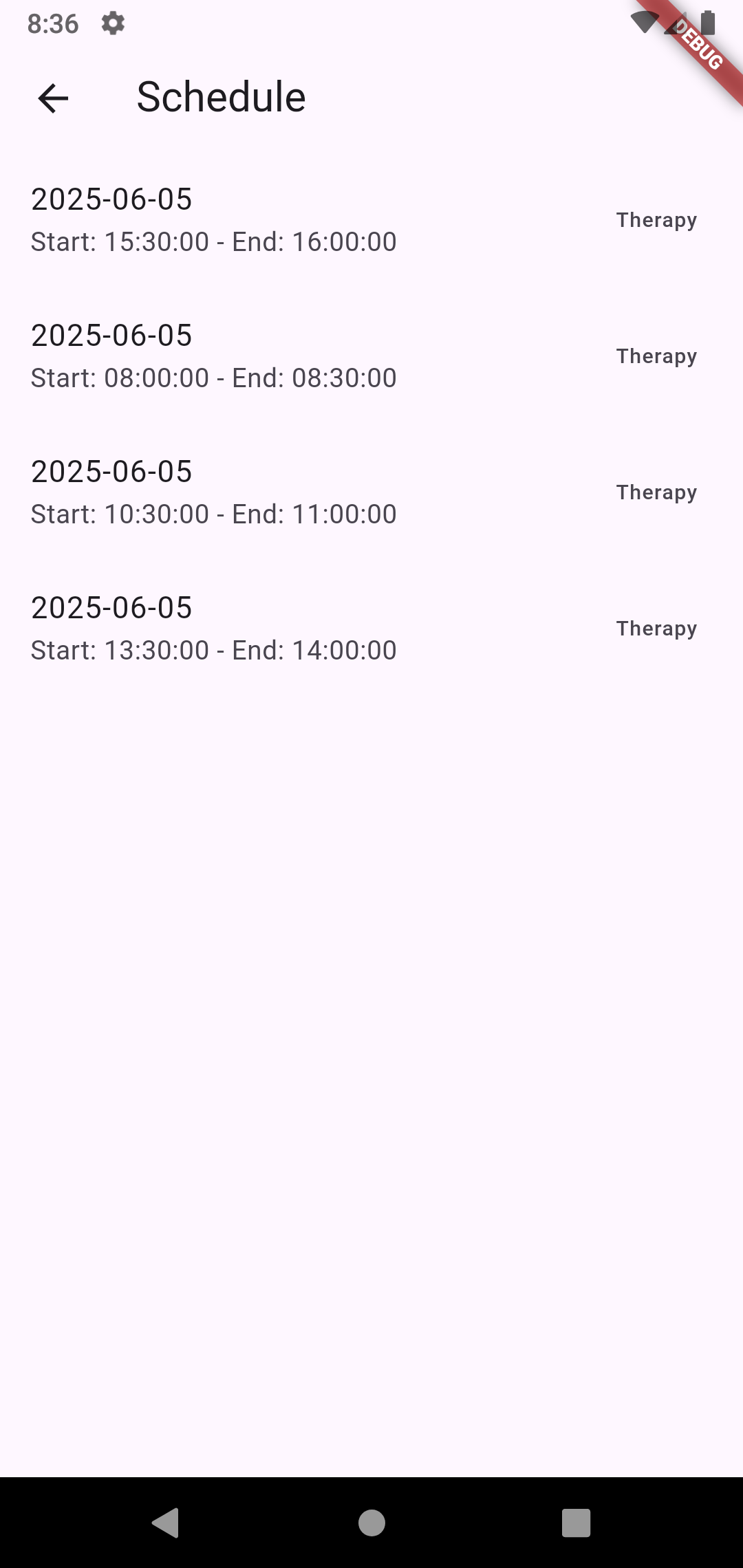


Рисунок 31 Android приложение расписание

Для взаимодействия с тг ботом нужно писать сообщения боту МойПриём

Для начала надо написать /start, потом ввести логин, пароль, и после этого тг бот будет присылать напоминания о приёмах.

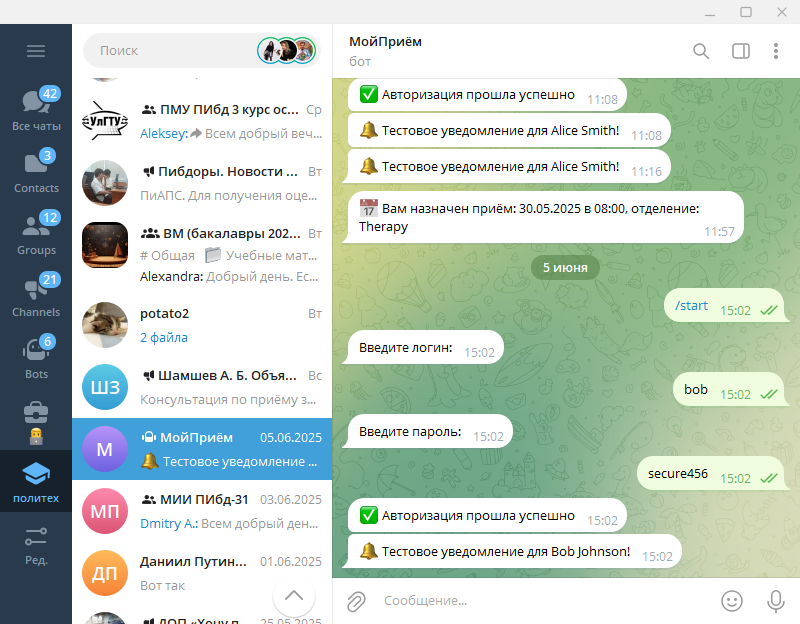


Рисунок 32 telegram диалог с ботом

**Диаграмма классов, сгенерированная по коду**

На рис. 33-35 предоставлены диаграммы пакетов с классами и интерфейсами.

На рис. 33 диаграмма с пакетом Repositories(пакет для работы с бд) и связанными классами.

На рис 34 диаграмма с пакетом Models, это модели сущностей предметной области.

На рис 35 диаграмма с пакетом Desctop, это все формы для работы с приложением. UI так сказать.

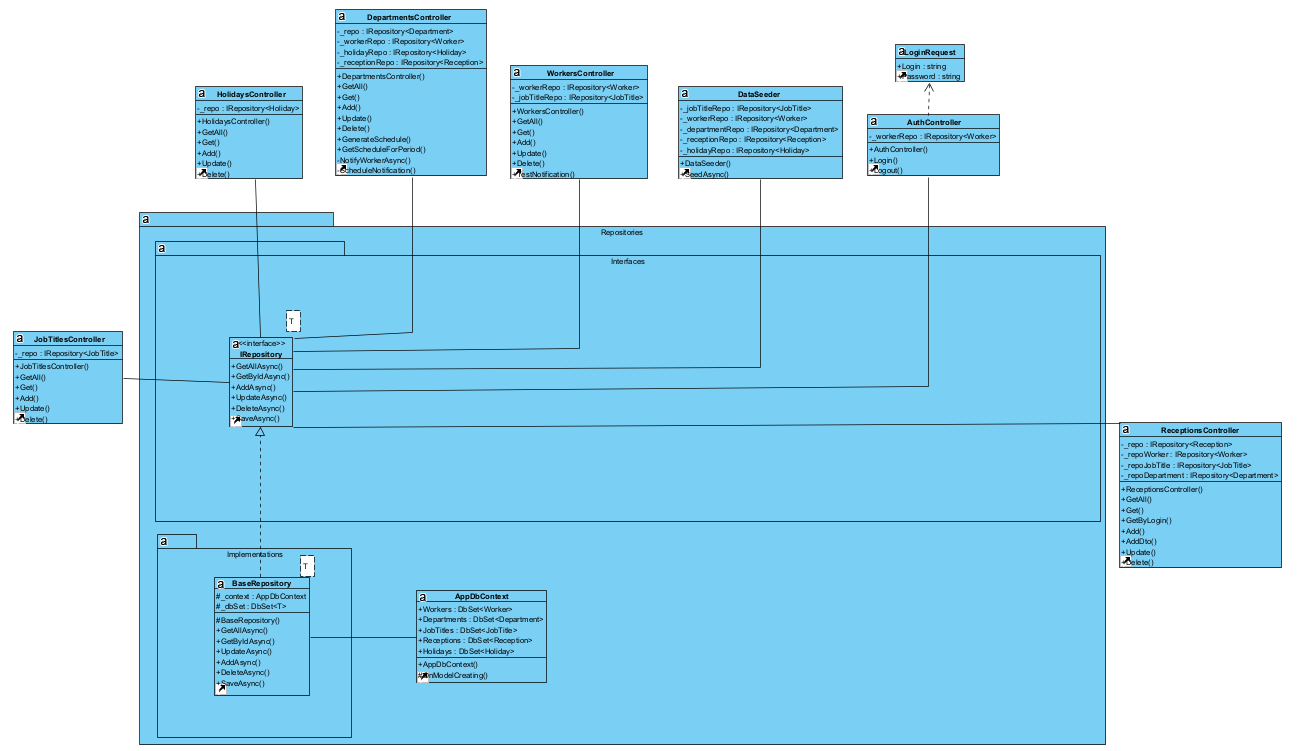


Рисунок 33 Сгенерированная диаграмма классов Repositories

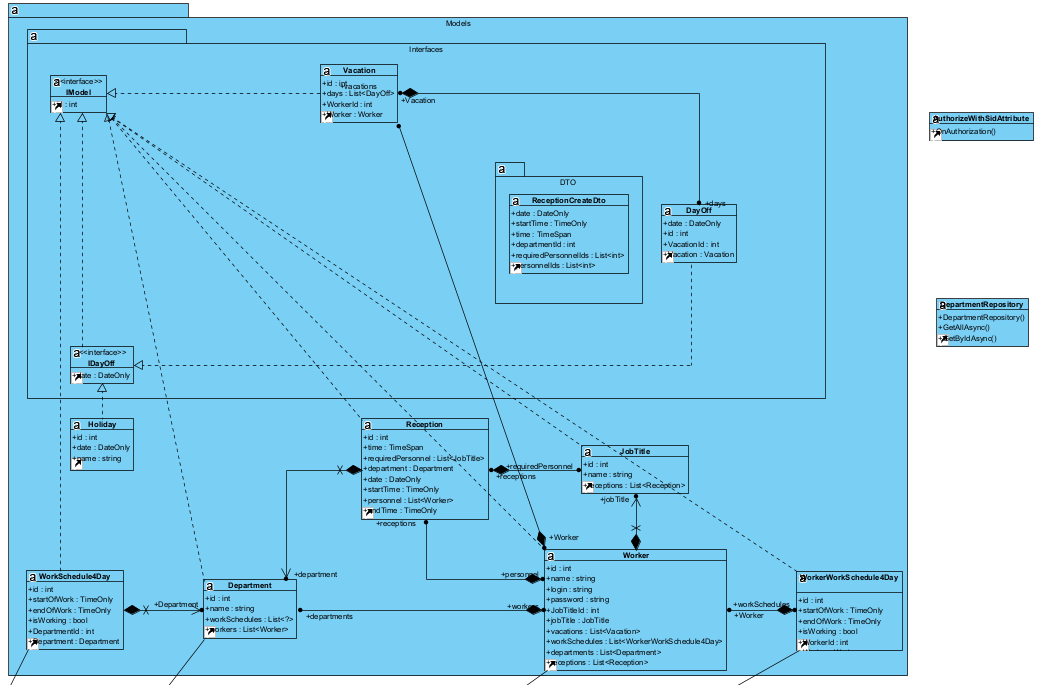


Рисунок 34 Сгенерированная диаграмма классов Models

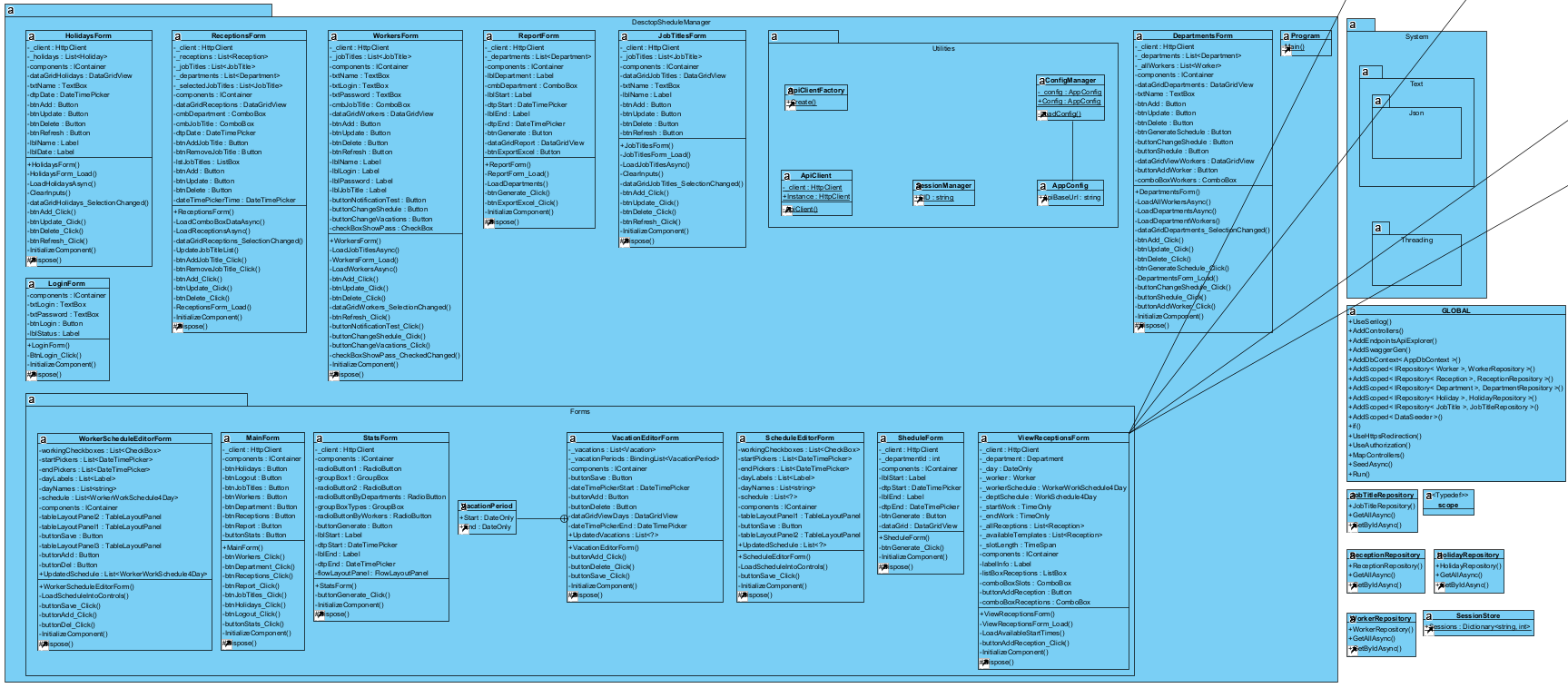


Рисунок 35 Сгенерированная диаграмма классов Desctop

Сквозная функциональность валидация и аутентификация реализована во всех классах, в виде проверки вводимых пользователем данных перед отправкой запроса на сервер и при отправке в бд.

Логирование/протоколирование реализовано во всех классах, из представленных на диаграммах выше, экранов взаимодействия с пользователем для отслеживания его действий, а также при отправке запроса и получения ответ с сервера. Данные протоколы и логи ведутся для упрощения отладки возможных ошибок.

Обработка ошибок так же реализована в классах взаимодействия с пользователем и работы с сервером для предотвращения ситуаций непредвиденного прекращения работы приложения.

# Выводы по реализации приложения на основании проекта

Во время реализации проекта конечный вид проекта претерпел изменения, т.к. в процессе разработки стали лучше понятно приоритетные функции, также реализовать некотрые функции оказалось проще по другому.

В связись с этим произошли серьезные изменениния в функционале – это удаление ролевых функций.

Изменения, которые произошли с проектом в ходе разработки можно назвать не существенными, так как не оказали влияние на общую архитектуру системы. Итоговая версия приложения отвечает всем основным требованиям, указанным в требованиях к проекту, и способна выполнять ключевые функции.

# Заключение

В результате выполнения проекта был создан полнофункциональный программный продукт, реализующий все заявленные цели и задачи в соответствии с исходным техническим заданием. Система продемонстрировала работоспособность, стабильность и соответствие требованиям, предъявленным на этапе проектирования.

В процессе разработки были приобретены практические навыки работы с инструментами моделирования, такими как **Visual Paradigm**, а также с платформой **Telegram API** для реализации бота. Особое внимание было уделено использованию **UML-диаграмм**, которые помогли структурировать архитектуру системы, наглядно отразить ключевые процессы и упростили последующую реализацию.

Проектирование программных систем стало важной частью работы. В ходе выполнения проекта были получены ценные знания в области анализа и предварительной подготовки архитектурных решений. Этот этап позволил:

* определить чёткую структуру приложения;
* грамотно разделить ответственность между компонентами;
* выстроить эффективную командную работу;
* заранее предусмотреть особенности реализации и возможные сложности;
* сформировать целостное видение итогового программного продукта.

Таким образом, выполнение данного проекта не только дало практический результат в виде готовой системы, но и способствовало развитию профессиональных навыков в области системного анализа, архитектурного проектирования и разработки программного обеспечения.

# Библиографический список

1. Крючкова, Е. Н. Объектно-ориентированное программирование: Архитектурное проектирование и паттерны программирования : учебно-методическое пособие / Е. Н. Крючкова, С. М. Старолетов. — Барнаул : АлтГТУ, 2020. — 180 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/292790>
2. Флегонтов, А. В. Моделирование информационных систем. Unified Modeling Language : учебное пособие / А. В. Флегонтов, И. Ю. Матюшичев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2022. — 112 с. — ISBN 978-5-8114-2907-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/206051
3. Корунова Н.В. Проектирование программного обеспечения: применение на практике / практикум по дисциплине «Проектирование информационных систем» / Н. В. Корунова. –Ульяновск : УлГТУ, 2019. – 69 с. - URL: […….](http://is.ulstu.ru/disc/pris). — Режим доступа: для авториз. пользователей.
4. Выполнение и оформление курсовых проектов (работ) [Текст]: методические указания / сост. Н. В. Корунова. - Ульяновск: УлГТУ, 2014. - 40 с. - Доступен также в Интернете <http://venec.ulstu.ru/lib/disk/2014/129.pdf>

….

# ПРИЛОЖЕНИЕ. Листинг кода

**Desctop+Api**

// ==== D:\piaps\Schedule\API\AuthorizeWithSidAttribute.cs ====

﻿using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

public static class SessionStore

{

// sid -> workerId

public static Dictionary<string, int> Sessions = new();

}

public class AuthorizeWithSidAttribute : Attribute, IAuthorizationFilter

{

public void OnAuthorization(AuthorizationFilterContext context)

{

var sid = context.HttpContext.Request.Headers["sid"].FirstOrDefault();

if (string.IsNullOrEmpty(sid) || !SessionStore.Sessions.ContainsKey(sid))

{

context.Result = new UnauthorizedObjectResult(new

{

error = "Unauthorized: Invalid or missing SID"

});

}

}

}

// ==== D:\piaps\Schedule\API\DataSeeder.cs ====

﻿using Models;

using Repositories.Interfaces;

public class DataSeeder

{

private readonly IRepository<JobTitle> \_jobTitleRepo;

private readonly IRepository<Worker> \_workerRepo;

private readonly IRepository<Department> \_departmentRepo;

private readonly IRepository<Reception> \_receptionRepo;

private readonly IRepository<Holiday> \_holidayRepo;

public DataSeeder(

IRepository<JobTitle> jobTitleRepo,

IRepository<Worker> workerRepo,

IRepository<Department> departmentRepo,

IRepository<Reception> receptionRepo,

IRepository<Holiday> holidayRepo)

{

\_jobTitleRepo = jobTitleRepo;

\_workerRepo = workerRepo;

\_departmentRepo = departmentRepo;

\_receptionRepo = receptionRepo;

\_holidayRepo = holidayRepo;

}

public async Task SeedAsync()

{

// Удаление в нужном порядке (зависимости!)

var receptions = await \_receptionRepo.GetAllAsync();

foreach (var r in receptions)

await \_receptionRepo.DeleteAsync(r);

await \_receptionRepo.SaveAsync();

var departments = await \_departmentRepo.GetAllAsync();

foreach (var d in departments)

await \_departmentRepo.DeleteAsync(d);

await \_departmentRepo.SaveAsync();

var workers = await \_workerRepo.GetAllAsync();

foreach (var w in workers)

await \_workerRepo.DeleteAsync(w);

await \_workerRepo.SaveAsync();

var jobTitles = await \_jobTitleRepo.GetAllAsync();

foreach (var jt in jobTitles)

await \_jobTitleRepo.DeleteAsync(jt);

await \_jobTitleRepo.SaveAsync();

var holidays = await \_holidayRepo.GetAllAsync();

foreach (var h in holidays)

await \_holidayRepo.DeleteAsync(h);

await \_holidayRepo.SaveAsync();

// 1. Должности

var doctor = new JobTitle { name = "Doctor" };

var nurse = new JobTitle { name = "Nurse" };

await \_jobTitleRepo.AddAsync(doctor);

await \_jobTitleRepo.AddAsync(nurse);

await \_jobTitleRepo.SaveAsync();

doctor = await \_jobTitleRepo.GetByIdAsync(doctor.id);

nurse = await \_jobTitleRepo.GetByIdAsync(nurse.id);

// 2. Сотрудники с отпусками

var vacation = new Vacation

{

days = new List<DayOff>

{

new DayOff { date = DateOnly.FromDateTime(DateTime.Today.AddDays(5)) }

}

};

var worker1 = new Worker

{

name = "Alice Smith",

login = "alice",

password = "alice",

jobTitle = doctor,

vacations = new List<Vacation> { vacation }

};

var worker2 = new Worker

{

name = "Bob Johnson",

login = "bob",

password = "secure456",

jobTitle = nurse,

vacations = new List<Vacation>()

};

await \_workerRepo.AddAsync(worker1);

await \_workerRepo.AddAsync(worker2);

await \_workerRepo.SaveAsync();

worker1 = await \_workerRepo.GetByIdAsync(worker1.id);

worker2 = await \_workerRepo.GetByIdAsync(worker2.id);

// 3. Расписание работы для сотрудников

// Расписание "2 через 2" для worker1 (Пн и Вт — рабочие)

var schedule2x2 = new List<WorkerWorkSchedule4Day>

{

new() { startOfWork = new TimeOnly(9, 0), endOfWork = new TimeOnly(17, 0), isWorking = true }, // Пн

new() { startOfWork = new TimeOnly(9, 0), endOfWork = new TimeOnly(17, 0), isWorking = true }, // Вт

new() { startOfWork = new TimeOnly(9, 0), endOfWork = new TimeOnly(17, 0), isWorking = false }, // Ср

new() { startOfWork = new TimeOnly(9, 0), endOfWork = new TimeOnly(17, 0), isWorking = false }, // Чт

};

worker1.workSchedules = schedule2x2;

// Расписание 5/2 для worker2 (Пн–Пт — рабочие, Сб и Вс — выходные)

var schedule5x2 = new List<WorkerWorkSchedule4Day>

{

new() { startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0), isWorking = true }, // Пн

new() { startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0), isWorking = true }, // Вт

new() { startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0), isWorking = true }, // Ср

new() { startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0), isWorking = true }, // Чт

new() { startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0), isWorking = true }, // Пт

new() { startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0), isWorking = false }, // Сб

new() { startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0), isWorking = false }, // Вс

};

worker2.workSchedules = schedule5x2;

// Обновляем работников с расписанием

await \_workerRepo.UpdateAsync(worker1);

await \_workerRepo.UpdateAsync(worker2);

await \_workerRepo.SaveAsync();

// 3. Отделение с графиком

var schedule = new List<WorkSchedule4Day>

{

new WorkSchedule4Day { isWorking = true, startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0) }, // Пн

new WorkSchedule4Day { isWorking = true, startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0) }, // Вт

new WorkSchedule4Day { isWorking = true, startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0) }, // Ср

new WorkSchedule4Day { isWorking = true, startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0) }, // Чт

new WorkSchedule4Day { isWorking = true, startOfWork = new TimeOnly(8, 0), endOfWork = new TimeOnly(16, 0) }, // Пт

new WorkSchedule4Day { isWorking = false },

new WorkSchedule4Day { isWorking = false }

};

var department = new Department

{

name = "Therapy",

workSchedules = schedule,

workers = new List<Worker> { worker1, worker2 }

};

await \_departmentRepo.AddAsync(department);

await \_departmentRepo.SaveAsync();

department = await \_departmentRepo.GetByIdAsync(department.id);

// 4. Приём

var reception = new Reception

{

time = TimeSpan.FromMinutes(30),

requiredPersonnel = new List<JobTitle> { doctor, nurse },

department = department

};

await \_receptionRepo.AddAsync(reception);

await \_receptionRepo.SaveAsync();

// 5. Праздник

var holiday = new Holiday

{

name = "New Year",

date = new DateOnly(DateTime.Today.Year, 1, 1)

};

await \_holidayRepo.AddAsync(holiday);

await \_holidayRepo.SaveAsync();

}

}

// ==== D:\piaps\Schedule\API\Program.cs ====

using Microsoft.EntityFrameworkCore;

using Models;

using Repositories.Interfaces;

using Repositories;

using Serilog;

var builder = WebApplication.CreateBuilder(args);

Log.Logger = new LoggerConfiguration()

.MinimumLevel.Information() //

.WriteTo.Console() //

.WriteTo.File("Logs/log-.txt", rollingInterval: RollingInterval.Day) // ( )

.CreateLogger();

builder.Host.UseSerilog();

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new() { Title = "Your API", Version = "v1" });

// sid

c.AddSecurityDefinition("sid", new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Name = "sid",

Type = Microsoft.OpenApi.Models.SecuritySchemeType.ApiKey,

In = Microsoft.OpenApi.Models.ParameterLocation.Header,

Description = "Session ID"

});

c.AddSecurityRequirement(new Microsoft.OpenApi.Models.OpenApiSecurityRequirement

{

{

new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Reference = new Microsoft.OpenApi.Models.OpenApiReference

{

Type = Microsoft.OpenApi.Models.ReferenceType.SecurityScheme,

Id = "sid"

}

},

Array.Empty<string>()

}

});

});

// PostgreSQL

builder.Services.AddDbContext<AppDbContext>(options =>

options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection")));

//

builder.Services.AddScoped<IRepository<Worker>, WorkerRepository>();

builder.Services.AddScoped<IRepository<Reception>, ReceptionRepository>();

builder.Services.AddScoped<IRepository<Department>, DepartmentRepository>();

builder.Services.AddScoped<IRepository<Holiday>, HolidayRepository>();

builder.Services.AddScoped<IRepository<JobTitle>, JobTitleRepository>();

builder.Services.AddScoped<DataSeeder>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

using (var scope = app.Services.CreateScope())

{

var seeder = scope.ServiceProvider.GetRequiredService<DataSeeder>();

await seeder.SeedAsync();

}

app.Run();

// ==== D:\piaps\Schedule\API\Controllers\AuthController.cs ====

﻿using Microsoft.AspNetCore.Mvc;

using Models;

using Repositories.Interfaces;

using System.Security.Cryptography;

[ApiController]

[Route("api/auth")]

public class AuthController : ControllerBase

{

private readonly IRepository<Worker> \_workerRepo;

public AuthController(IRepository<Worker> workerRepo)

{

\_workerRepo = workerRepo;

}

[HttpPost("login")]

public async Task<IActionResult> Login([FromBody] LoginRequest req)

{

var workers = await \_workerRepo.GetAllAsync();

var user = workers.FirstOrDefault(w => w.login == req.Login && w.password == req.Password);

if (user == null)

return Unauthorized(new { error = "Invalid login or password" });

// Генерация SID

var sid = Convert.ToHexString(RandomNumberGenerator.GetBytes(16));

SessionStore.Sessions[sid] = user.id;

return Ok(new { sid });

}

[HttpPost("logout")]

[AuthorizeWithSid]

public IActionResult Logout()

{

var sid = Request.Headers["sid"].ToString();

SessionStore.Sessions.Remove(sid);

return Ok(new { message = "Logged out" });

}

}

public class LoginRequest

{

public string Login { get; set; }

public string Password { get; set; }

}

// ==== D:\piaps\Schedule\API\Controllers\DepartmentsController.cs ====

﻿using Microsoft.AspNetCore.Mvc;

using Models;

using Repositories.Interfaces;

[ApiController]

[Route("api/department")]

public class DepartmentsController : ControllerBase

{

private readonly IRepository<Department> \_repo;

private readonly IRepository<Worker> \_workerRepo;

private readonly IRepository<Holiday> \_holidayRepo;

private readonly IRepository<Reception> \_receptionRepo;

public DepartmentsController(IRepository<Department> repo, IRepository<Worker> workerRepo, IRepository<Holiday> holidayRepo, IRepository<Reception> receptionRepo)

{

\_repo = repo;

\_workerRepo = workerRepo;

\_holidayRepo = holidayRepo;

\_receptionRepo = receptionRepo;

}

[HttpGet("get-all")]

[AuthorizeWithSid]

public async Task<IActionResult> GetAll() => Ok(await \_repo.GetAllAsync());

[HttpGet("get/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Get(int id)

{

var item = await \_repo.GetByIdAsync(id);

return item == null ? NotFound() : Ok(item);

}

[HttpPost("add")]

[AuthorizeWithSid]

public async Task<IActionResult> Add([FromBody] Department item)

{

if (item.workers != null)

{

for (int i = 0; i < item.workers.Count; i++)

{

var existingWorker = await \_workerRepo.GetByIdAsync(item.workers[i].id);

if (existingWorker != null)

{

item.workers[i] = existingWorker;

}

else

{

// Можно либо вернуть ошибку, либо создать новую должность

return BadRequest(new { error = "Invalid jobTitle id" });

}

}

}

await \_repo.AddAsync(item);

await \_repo.SaveAsync();

return Ok(item);

}

[HttpPut("update/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Update(int id, [FromBody] Department updated)

{

var existing = await \_repo.GetByIdAsync(id);

if (existing is null) return NotFound();

existing.name = updated.name;

existing.workSchedules = updated.workSchedules;

existing.workers = updated.workers;

if (existing.workers != null)

{

for (int i = 0; i < existing.workers.Count; i++)

{

var existingWorker = await \_workerRepo.GetByIdAsync(existing.workers[i].id);

if (existingWorker != null)

{

existing.workers[i] = existingWorker;

}

else

{

// Можно либо вернуть ошибку, либо создать новую должность

return BadRequest(new { error = "Invalid jobTitle id" });

}

}

}

await \_repo.UpdateAsync(existing);

await \_repo.SaveAsync();

return Ok(existing);

}

[HttpDelete("delete/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Delete(int id)

{

var existing = await \_repo.GetByIdAsync(id);

if (existing is null) return NotFound();

await \_repo.DeleteAsync(existing);

await \_repo.SaveAsync();

return Ok(new { message = "Department deleted" });

}

/// <summary>

/// жесть

/// </summary>

/// <returns></returns>

[HttpPost("generate-schedule/{departmentId}")]

[AuthorizeWithSid]

public async Task<IActionResult> GenerateSchedule(int departmentId)

{

var department = await \_repo.GetByIdAsync(departmentId);

if (department == null)

return NotFound(new { error = "Department not found" });

var receptions = (await \_receptionRepo.GetAllAsync())

.Where(r => r.department?.id == departmentId && !r.date.HasValue)

.ToList();

var holidays = await \_holidayRepo.GetAllAsync();

var allWorkers = department.workers;

List<Reception> scheduled = new();

foreach (var reception in receptions)

{

for (int offset = 0; offset < 7; offset++) // пробуем 7 дней вперёд

{

var targetDate = DateOnly.FromDateTime(DateTime.Now.Date.AddDays(offset));

// Праздник?

if (holidays.Any(h => h.date == targetDate))

continue;

var dayOfWeek = (int)targetDate.DayOfWeek;

if (dayOfWeek == 0) dayOfWeek = 7; // Sunday to 7

var schedule = department.workSchedules.ElementAtOrDefault(dayOfWeek - 1);

if (schedule == null || !schedule.isWorking)

continue;

List<Worker> workingWorkers = allWorkers.Where(

w => w.vacations == null || !w.vacations.Any(

v => v != null && v.days.Any(

d => d.Equals(targetDate)

)

)

).ToList();

// предположим что у нас отрезки времени это пол часа

for (TimeOnly receptionStartTime = schedule.startOfWork;

receptionStartTime < schedule.endOfWork;

receptionStartTime = receptionStartTime.Add(new TimeSpan(0, 30, 0)))

{

var receptionEndTime = receptionStartTime.Add(reception.time);

List<Worker> freeWorkers = new();

foreach (var worker in workingWorkers)

{

bool nany = true;

if (worker.receptions != null && worker.receptions.Any())

foreach (var r in worker.receptions)

{

if (r.date.Equals(targetDate)

&&

(

r.startTime <= receptionEndTime && r.startTime >= receptionStartTime

||

r.endTime <= receptionEndTime && r.endTime >= receptionStartTime

||

r.endTime <= receptionStartTime && r.startTime >= receptionStartTime

||

r.endTime <= receptionEndTime && r.startTime >= receptionEndTime

)

)

{

nany = false;

break;

}

}

if (nany)

freeWorkers.Add(worker);

}

//List<Worker> freeWorkers = workingWorkers.Where(

// w => !w.receptions.Any(

// r => r.date.Equals(targetDate) && (

// r.startTime <= receptionEndTime && r.startTime >= receptionStartTime

// ||

// r.endTime <= receptionEndTime && r.endTime >= receptionStartTime

// ||

// r.endTime <= receptionStartTime && r.startTime >= receptionStartTime

// ||

// r.endTime <= receptionEndTime && r.startTime >= receptionEndTime

// )

// )

// ).ToList();

foreach (JobTitle job in reception.requiredPersonnel)

{

Worker ourWorker = freeWorkers.FirstOrDefault(w => w.jobTitle == job);

if (ourWorker == null)

break;

reception.personnel.Add(ourWorker);

}

if (reception.personnel.Count == reception.requiredPersonnel.Count)

{

reception.date = targetDate;

reception.startTime = receptionStartTime;

await \_receptionRepo.UpdateAsync(reception);

await \_receptionRepo.SaveAsync();

// 🔔 Уведомляем всех работников

foreach (var worker in reception.personnel)

{

\_ = NotifyWorkerAsync(worker, reception, immediate: true);

ScheduleNotification(worker, reception, TimeSpan.FromMinutes(15));

ScheduleNotification(worker, reception, TimeSpan.FromMinutes(1));

}

receptions = (await \_receptionRepo.GetAllAsync())

.Where(r => r.department?.id == departmentId && !r.date.HasValue)

.ToList();

scheduled.Add(reception);

break;

}

reception.personnel.Clear();

}

if (reception.personnel.Count == reception.requiredPersonnel.Count)

{

break;

}

}

}

return Ok(scheduled);

}

private async Task NotifyWorkerAsync(Worker worker, Reception reception, bool immediate = false)

{

try

{

using var client = new HttpClient();

var text = immediate

? $"📅 Вам назначен приём: {reception.date:dd.MM.yyyy} в {reception.startTime:hh\\:mm}, отделение: {reception.department?.name}"

: $"⏰ Напоминание: Приём в {reception.startTime:hh\\:mm}.";

var payload = new

{

login = worker.login,

text

};

var response = await client.PostAsJsonAsync("http://localhost:8000/notify", payload);

response.EnsureSuccessStatusCode();

}

catch (Exception ex)

{

Console.WriteLine($"[NotifyWorkerAsync Error] {ex.Message}");

}

}

private void ScheduleNotification(Worker worker, Reception reception, TimeSpan beforeStart)

{

var now = DateTime.Now;

var targetTime = reception.date.Value.ToDateTime(reception.startTime.Value).Add(-beforeStart);

var delay = targetTime - now;

if (delay.TotalMilliseconds <= 0)

return;

Task.Delay(delay).ContinueWith(\_ => NotifyWorkerAsync(worker, reception));

}

[HttpGet("get-schedule-for-period")]

[AuthorizeWithSid]

public async Task<IActionResult> GetScheduleForPeriod(

[FromQuery] int departmentId,

[FromQuery] DateOnly start,

[FromQuery] DateOnly end)

{

var department = await \_repo.GetByIdAsync(departmentId);

if (department == null)

return NotFound(new { error = "Department not found" });

var receptions = await \_receptionRepo.GetAllAsync();

var filtered = receptions

.Where(r =>

r.department != null &&

r.department.id == departmentId &&

r.date.HasValue &&

r.date.Value >= start &&

r.date.Value <= end)

.ToList();

return Ok(filtered);

}

}

// ==== D:\piaps\Schedule\API\Controllers\HolidaysController.cs ====

﻿using Microsoft.AspNetCore.Mvc;

using Models;

using Repositories.Interfaces;

[ApiController]

[Route("api/holiday")]

public class HolidaysController : ControllerBase

{

private readonly IRepository<Holiday> \_repo;

public HolidaysController(IRepository<Holiday> repo)

{

\_repo = repo;

}

[HttpGet("get-all")]

[AuthorizeWithSid]

public async Task<IActionResult> GetAll() => Ok(await \_repo.GetAllAsync());

[HttpGet("get/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Get(int id)

{

var item = await \_repo.GetByIdAsync(id);

return item == null ? NotFound() : Ok(item);

}

[HttpPost("add")]

[AuthorizeWithSid]

public async Task<IActionResult> Add([FromBody] Holiday item)

{

await \_repo.AddAsync(item);

await \_repo.SaveAsync();

return Ok(item);

}

[HttpPut("update/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Update(int id, [FromBody] Holiday updated)

{

var existing = await \_repo.GetByIdAsync(id);

if (existing is null) return NotFound();

existing.name = updated.name;

existing.date = updated.date;

await \_repo.UpdateAsync(existing);

await \_repo.SaveAsync();

return Ok(existing);

}

[HttpDelete("delete/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Delete(int id)

{

var existing = await \_repo.GetByIdAsync(id);

if (existing is null) return NotFound();

await \_repo.DeleteAsync(existing);

await \_repo.SaveAsync();

return Ok(new { message = "Holiday deleted" });

}

}

// ==== D:\piaps\Schedule\API\Controllers\JobTitlesController.cs ====

﻿using Microsoft.AspNetCore.Mvc;

using Models;

using Repositories.Interfaces;

[ApiController]

[Route("api/job-title")]

public class JobTitlesController : ControllerBase

{

private readonly IRepository<JobTitle> \_repo;

public JobTitlesController(IRepository<JobTitle> repo)

{

\_repo = repo;

}

[HttpGet("get-all")]

[AuthorizeWithSid]

public async Task<IActionResult> GetAll() => Ok(await \_repo.GetAllAsync());

[HttpGet("get/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Get(int id)

{

var item = await \_repo.GetByIdAsync(id);

return item == null ? NotFound() : Ok(item);

}

[HttpPost("add")]

[AuthorizeWithSid]

public async Task<IActionResult> Add([FromBody] JobTitle item)

{

await \_repo.AddAsync(item);

await \_repo.SaveAsync();

return Ok(item);

}

[HttpPut("update/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Update(int id, [FromBody] JobTitle updated)

{

var existing = await \_repo.GetByIdAsync(id);

if (existing is null) return NotFound();

existing.name = updated.name;

await \_repo.UpdateAsync(existing);

await \_repo.SaveAsync();

return Ok(existing);

}

[HttpDelete("delete/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Delete(int id)

{

var existing = await \_repo.GetByIdAsync(id);

if (existing is null) return NotFound();

await \_repo.DeleteAsync(existing);

await \_repo.SaveAsync();

return Ok(new { message = "JobTitle deleted" });

}

}

// ==== D:\piaps\Schedule\API\Controllers\ReceptionsController.cs ====

﻿using Microsoft.AspNetCore.Mvc;

using Models;

using Models.DTO;

using Repositories.Interfaces;

[ApiController]

[Route("api/reception")]

public class ReceptionsController : ControllerBase

{

private readonly IRepository<Reception> \_repo;

private readonly IRepository<Worker> \_repoWorker;

private readonly IRepository<JobTitle> \_repoJobTitle;

private readonly IRepository<Department> \_repoDepartment;

public ReceptionsController(IRepository<Reception> repo, IRepository<Worker> repoWorker, IRepository<JobTitle> repoJobTitle, IRepository<Department> repoDepartment)

{

\_repo = repo;

\_repoWorker = repoWorker;

\_repoJobTitle = repoJobTitle;

\_repoDepartment = repoDepartment;

}

[HttpGet("get-all")]

[AuthorizeWithSid]

public async Task<IActionResult> GetAll() => Ok(await \_repo.GetAllAsync());

[HttpGet("get/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Get(int id)

{

var item = await \_repo.GetByIdAsync(id);

return item == null ? NotFound() : Ok(item);

}

[HttpGet("getByLogin/{login}")]

[AuthorizeWithSid]

public async Task<IActionResult> GetByLogin(string login)

{

var sid = HttpContext.Request.Headers["sid"].FirstOrDefault();

var workers = await \_repoWorker.GetAllAsync();

if (workers is null)

return NotFound();

var worker = workers.FirstOrDefault(w => w.login == login);

if (worker is null)

return NotFound();

if (worker.id != SessionStore.Sessions[sid])

return NotFound();

var receptions = await \_repo.GetAllAsync();

var filteredReceptions = receptions.Where(r => r.personnel != null && r.personnel.Contains(worker)).ToList();

return filteredReceptions is null ? NotFound() : Ok(filteredReceptions);

}

[HttpPost("add")]

[AuthorizeWithSid]

public async Task<IActionResult> Add([FromBody] Reception item)

{

if (item.personnel != null)

{

for (int i = 0; i < item.personnel.Count; i++)

{

var existingWorker = await \_repoWorker.GetByIdAsync(item.personnel[i].id);

if (existingWorker != null)

{

item.personnel[i] = existingWorker;

}

else

{

// Можно либо вернуть ошибку, либо создать новую должность

return BadRequest(new { error = "Invalid jobTitle id" });

}

}

}

if (item.requiredPersonnel != null)

{

for (int i = 0; i < item.requiredPersonnel.Count; i++)

{

var existingJob = await \_repoJobTitle.GetByIdAsync(item.requiredPersonnel[i].id);

if (existingJob != null)

{

item.requiredPersonnel[i] = existingJob;

}

else

{

// Можно либо вернуть ошибку, либо создать новую должность

return BadRequest(new { error = "Invalid jobTitle id" });

}

}

}

if (item.department != null)

{

var existingDepartment = await \_repoDepartment.GetByIdAsync(item.department.id);

if (existingDepartment != null)

{

item.department = existingDepartment;

}

else

{

// Можно либо вернуть ошибку, либо создать новую должность

return BadRequest(new { error = "Invalid jobTitle id" });

}

}

await \_repo.AddAsync(item);

await \_repo.SaveAsync();

return Ok(item);

}

[HttpPost("add-dto")]

[AuthorizeWithSid]

public async Task<IActionResult> AddDto([FromBody] ReceptionCreateDto dto)

{

// Получаем отделение

var department = await \_repoDepartment.GetByIdAsync(dto.departmentId);

if (department == null)

return BadRequest(new { error = "Invalid department id" });

// Получаем сотрудников

var personnel = new List<Worker>();

foreach (var workerId in dto.personnelIds)

{

var worker = await \_repoWorker.GetByIdAsync(workerId);

if (worker == null)

return BadRequest(new { error = $"Invalid worker id: {workerId}" });

personnel.Add(worker);

}

// Получаем требуемые должности

var requiredPersonnel = new List<JobTitle>();

foreach (var jobTitleId in dto.requiredPersonnelIds)

{

var jobTitle = await \_repoJobTitle.GetByIdAsync(jobTitleId);

if (jobTitle == null)

return BadRequest(new { error = $"Invalid job title id: {jobTitleId}" });

requiredPersonnel.Add(jobTitle);

}

// Формируем новый приём

var reception = new Reception

{

date = dto.date,

startTime = dto.startTime,

time = dto.time,

department = department,

personnel = personnel,

requiredPersonnel = requiredPersonnel

};

await \_repo.AddAsync(reception);

await \_repo.SaveAsync();

return Ok(reception);

}

[HttpPut("update/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Update(int id, [FromBody] Reception updated)

{

var existing = await \_repo.GetByIdAsync(id);

if (existing is null) return NotFound();

existing.time = updated.time;

existing.requiredPersonnel = updated.requiredPersonnel;

existing.personnel = updated.personnel;

existing.department = updated.department;

existing.date = updated.date;

existing.startTime = updated.startTime;

if (existing.personnel != null)

{

for (int i = 0; i < existing.personnel.Count; i++)

{

var existingWorker = await \_repoWorker.GetByIdAsync(existing.personnel[i].id);

if (existingWorker != null)

{

existing.personnel[i] = existingWorker;

}

else

{

// Можно либо вернуть ошибку, либо создать новую должность

return BadRequest(new { error = "Invalid jobTitle id" });

}

}

}

if (existing.requiredPersonnel != null)

{

for (int i = 0; i < existing.requiredPersonnel.Count; i++)

{

var existingJob = await \_repoJobTitle.GetByIdAsync(existing.requiredPersonnel[i].id);

if (existingJob != null)

{

existing.requiredPersonnel[i] = existingJob;

}

else

{

// Можно либо вернуть ошибку, либо создать новую должность

return BadRequest(new { error = "Invalid jobTitle id" });

}

}

}

if (existing.department != null)

{

var existingDepartment = await \_repoDepartment.GetByIdAsync(existing.department.id);

if (existingDepartment != null)

{

existing.department = existingDepartment;

}

else

{

// Можно либо вернуть ошибку, либо создать новую должность

return BadRequest(new { error = "Invalid jobTitle id" });

}

}

await \_repo.UpdateAsync(existing);

await \_repo.SaveAsync();

return Ok(existing);

}

[HttpDelete("delete/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Delete(int id)

{

var existing = await \_repo.GetByIdAsync(id);

if (existing is null) return NotFound();

await \_repo.DeleteAsync(existing);

await \_repo.SaveAsync();

return Ok(new { message = "Reception deleted" });

}

}

// ==== D:\piaps\Schedule\API\Controllers\WorkersController.cs ====

﻿using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Models;

using Repositories.Interfaces;

[ApiController]

[Route("api/worker")]

public class WorkersController : ControllerBase

{

private readonly IRepository<Worker> \_workerRepo;

private readonly IRepository<JobTitle> \_jobTitleRepo;

public WorkersController(IRepository<Worker> workerRepo, IRepository<JobTitle> jobTitleRepo)

{

\_workerRepo = workerRepo;

\_jobTitleRepo = jobTitleRepo;

}

[HttpGet("get-all")]

[AuthorizeWithSid]

public async Task<IActionResult> GetAll()

{

var workers = await \_workerRepo.GetAllAsync();

return Ok(workers);

}

[HttpGet("get/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Get(int id)

{

var worker = await \_workerRepo.GetByIdAsync(id);

return worker is null ? NotFound() : Ok(worker);

}

[HttpPost("add")]

[AuthorizeWithSid]

public async Task<IActionResult> Add([FromBody] Worker worker)

{

if (worker.jobTitle != null)

{

var existingJobTitle = await \_jobTitleRepo.GetByIdAsync(worker.jobTitle.id);

if (existingJobTitle != null)

{

worker.jobTitle = existingJobTitle;

}

else

{

// Можно либо вернуть ошибку, либо создать новую должность

return BadRequest(new { error = "Invalid jobTitle id" });

}

}

await \_workerRepo.AddAsync(worker);

await \_workerRepo.SaveAsync();

return Ok(worker);

}

[HttpPut("update/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Update(int id, [FromBody] Worker updated)

{

var existing = await \_workerRepo.GetByIdAsync(id);

if (existing is null) return NotFound();

existing.name = updated.name;

existing.login = updated.login;

existing.password = updated.password;

existing.jobTitle = updated.jobTitle;

existing.vacations = updated.vacations;

existing.workSchedules = updated.workSchedules;

if (existing.jobTitle != null)

{

var existingJobTitle = await \_jobTitleRepo.GetByIdAsync(existing.jobTitle.id);

if (existingJobTitle != null)

{

existing.jobTitle = existingJobTitle;

}

else

{

// Можно либо вернуть ошибку, либо создать новую должность

return BadRequest(new { error = "Invalid jobTitle id" });

}

}

await \_workerRepo.UpdateAsync(existing);

await \_workerRepo.SaveAsync();

return Ok(existing);

}

[HttpDelete("delete/{id}")]

[AuthorizeWithSid]

public async Task<IActionResult> Delete(int id)

{

var existing = await \_workerRepo.GetByIdAsync(id);

if (existing is null) return NotFound();

await \_workerRepo.DeleteAsync(existing);

await \_workerRepo.SaveAsync();

return Ok(new { message = "Worker deleted" });

}

[HttpGet("test-notification/{id}")]

public async Task<IActionResult> TestNotification(int id)

{

var worker = await \_workerRepo.GetByIdAsync(id);

if (worker == null)

return NotFound(new { error = "Worker not found" });

var httpClient = new HttpClient();

var requestBody = new

{

login = worker.login,

text = $"🔔 Тестовое уведомление для {worker.name}!"

};

var response = await httpClient.PostAsJsonAsync("http://localhost:8000/notify", requestBody);

if (response.IsSuccessStatusCode)

{

return Ok(new { message = "Notification sent." });

}

else

{

var content = await response.Content.ReadAsStringAsync();

return StatusCode((int)response.StatusCode, new { error = "Failed to send notification", details = content });

}

}

}

// ==== D:\piaps\Schedule\API\obj\Debug\net8.0\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\Schedule\API\obj\Debug\net8.0\API.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("API")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+504c82c7efe4e511801a3b9066c8d233e0a42f0d")]

[assembly: System.Reflection.AssemblyProductAttribute("API")]

[assembly: System.Reflection.AssemblyTitleAttribute("API")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\Schedule\API\obj\Debug\net8.0\API.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::Microsoft.AspNetCore.Builder;

global using global::Microsoft.AspNetCore.Hosting;

global using global::Microsoft.AspNetCore.Http;

global using global::Microsoft.AspNetCore.Routing;

global using global::Microsoft.Extensions.Configuration;

global using global::Microsoft.Extensions.DependencyInjection;

global using global::Microsoft.Extensions.Hosting;

global using global::Microsoft.Extensions.Logging;

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Net.Http.Json;

global using global::System.Threading;

global using global::System.Threading.Tasks;

// ==== D:\piaps\Schedule\API\obj\Debug\net8.0\API.MvcApplicationPartsAssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("Swashbuckle.AspNetCore.SwaggerGen")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\Schedule\API\obj\Release\net8.0\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\Schedule\API\obj\Release\net8.0\API.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("API")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

[assembly: System.Reflection.AssemblyProductAttribute("API")]

[assembly: System.Reflection.AssemblyTitleAttribute("API")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\Schedule\API\obj\Release\net8.0\API.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::Microsoft.AspNetCore.Builder;

global using global::Microsoft.AspNetCore.Hosting;

global using global::Microsoft.AspNetCore.Http;

global using global::Microsoft.AspNetCore.Routing;

global using global::Microsoft.Extensions.Configuration;

global using global::Microsoft.Extensions.DependencyInjection;

global using global::Microsoft.Extensions.Hosting;

global using global::Microsoft.Extensions.Logging;

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Net.Http.Json;

global using global::System.Threading;

global using global::System.Threading.Tasks;

// ==== D:\piaps\Schedule\API\obj\Release\net8.0\API.MvcApplicationPartsAssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: Microsoft.AspNetCore.Mvc.ApplicationParts.ApplicationPartAttribute("Swashbuckle.AspNetCore.SwaggerGen")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\Schedule\DesctopSheduleManager\Program.cs ====

namespace DesctopSheduleManager

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

// To customize application configuration such as set high DPI settings or default font,

// see https://aka.ms/applicationconfiguration.

ApplicationConfiguration.Initialize();

Application.Run(new LoginForm());

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\DepartmentsForm.cs ====

﻿using DesctopSheduleManager.Forms;

using DesctopSheduleManager.Utilities;

using DocumentFormat.OpenXml.Wordprocessing;

using Models;

using System.Net.Http.Json;

using System.Text.Json;

namespace DesctopSheduleManager

{

public partial class DepartmentsForm : Form

{

private readonly HttpClient \_client;

private List<Department> \_departments = new();

private List<Worker> \_allWorkers = new();

public DepartmentsForm()

{

InitializeComponent();

\_client = ApiClient.Instance;

}

private async Task LoadAllWorkersAsync()

{

try

{

\_allWorkers = await \_client.GetFromJsonAsync<List<Worker>>("api/worker/get-all") ?? new();

comboBoxWorkers.DataSource = \_allWorkers;

comboBoxWorkers.DisplayMember = "name";

comboBoxWorkers.ValueMember = "id";

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки персонала: {ex.Message}");

}

}

private async Task LoadDepartmentsAsync()

{

try

{

var departments = await \_client.GetFromJsonAsync<List<Department>>("api/department/get-all");

if (departments != null)

{

\_departments = departments;

dataGridDepartments.DataSource = \_departments.Select(d => new

{

d.id,

d.name,

WorkersCount = d.workers?.Count ?? 0

}).ToList();

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки данных: {ex.Message}");

}

}

private void LoadDepartmentWorkers()

{

if (dataGridDepartments.CurrentRow == null) return;

var selectedDepartment = \_departments[dataGridDepartments.CurrentRow.Index];

if (selectedDepartment == null) return;

var workers = selectedDepartment.workers ?? new List<Worker>();

dataGridViewWorkers.DataSource = workers.Select(w => new

{

w.id,

w.name,

Должность = w.jobTitle?.name

}).ToList();

}

private void dataGridDepartments\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridDepartments.CurrentRow == null) return;

txtName.Text = dataGridDepartments.CurrentRow.Cells["name"].Value?.ToString();

LoadDepartmentWorkers(); // <-- обновляем работников

}

private async void btnAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(txtName.Text))

{

MessageBox.Show("Введите название отделения.");

return;

}

var newDepartment = new Department { name = txtName.Text };

try

{

var response = await \_client.PostAsJsonAsync("api/department/add", newDepartment);

if (response.IsSuccessStatusCode)

{

await LoadDepartmentsAsync();

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка добавления: {response.StatusCode}\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения к API: {ex.Message}");

}

}

private async void btnUpdate\_Click(object sender, EventArgs e)

{

if (dataGridDepartments.CurrentRow == null) return;

if (string.IsNullOrWhiteSpace(txtName.Text))

{

MessageBox.Show("Введите название отделения.");

return;

}

var id = (int)dataGridDepartments.CurrentRow.Cells["id"].Value;

var updatedDepartment = new Department

{

id = id,

name = txtName.Text

};

try

{

var response = await \_client.PutAsJsonAsync($"api/department/update/{id}", updatedDepartment);

if (response.IsSuccessStatusCode)

{

await LoadDepartmentsAsync();

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка обновления: {response.StatusCode}\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения к API: {ex.Message}");

}

}

private async void btnDelete\_Click(object sender, EventArgs e)

{

if (dataGridDepartments.CurrentRow == null) return;

var id = (int)dataGridDepartments.CurrentRow.Cells["id"].Value;

var result = MessageBox.Show("Удалить выбранный департамент?", "Подтверждение", MessageBoxButtons.YesNo);

if (result != DialogResult.Yes) return;

try

{

var response = await \_client.DeleteAsync($"api/department/delete/{id}");

if (response.IsSuccessStatusCode)

{

await LoadDepartmentsAsync();

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка удаления: {response.StatusCode}\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения к API: {ex.Message}");

}

}

private async void btnGenerateSchedule\_Click(object sender, EventArgs e)

{

if (dataGridDepartments.CurrentRow == null)

{

MessageBox.Show("Выберите отделение.");

return;

}

var id = (int)dataGridDepartments.CurrentRow.Cells["id"].Value;

var result = MessageBox.Show($"Сгенерировать расписание для отделения ID {id}?", "Подтверждение", MessageBoxButtons.YesNo);

if (result != DialogResult.Yes) return;

try

{

var response = await \_client.PostAsync($"api/department/generate-schedule/{id}", null);

if (response.IsSuccessStatusCode)

{

MessageBox.Show("Расписание успешно сгенерировано.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка генерации: {response.StatusCode}\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения к API: {ex.Message}");

}

}

private async void DepartmentsForm\_Load(object sender, EventArgs e)

{

await LoadDepartmentsAsync();

await LoadAllWorkersAsync();

}

private async void buttonChangeShedule\_Click(object sender, EventArgs e)

{

if (dataGridDepartments.CurrentRow == null)

{

MessageBox.Show("Выберите отделение.");

return;

}

var selectedDepartment = \_departments[dataGridDepartments.CurrentRow.Index];

if (selectedDepartment == null)

{

MessageBox.Show("Не удалось получить отделение.");

return;

}

var editorForm = new ScheduleEditorForm(selectedDepartment.workSchedules);

if (editorForm.ShowDialog() == DialogResult.OK)

{

var updatedSchedule = editorForm.UpdatedSchedule;

foreach (var ws in updatedSchedule)

{

if (ws != null)

ws.DepartmentId = selectedDepartment.id;

}

selectedDepartment.workSchedules = updatedSchedule;

try

{

var response = await \_client.PutAsJsonAsync(

$"api/department/update/{selectedDepartment.id}", selectedDepartment);

if (response.IsSuccessStatusCode)

{

MessageBox.Show("Расписание обновлено.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка обновления: {response.StatusCode}\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения к API:\n{ex.Message}");

}

}

}

private void buttonShedule\_Click(object sender, EventArgs e)

{

if (dataGridDepartments.CurrentRow == null)

{

MessageBox.Show("Выберите отделение.");

return;

}

var selectedDepartment = \_departments[dataGridDepartments.CurrentRow.Index];

if (selectedDepartment == null)

{

MessageBox.Show("Не удалось получить отделение.");

return;

}

var sheduleForm = new SheduleForm(selectedDepartment.id);

sheduleForm.ShowDialog();

}

private async void buttonAddWorker\_Click(object sender, EventArgs e)

{

if (dataGridDepartments.CurrentRow == null)

{

MessageBox.Show("Выберите отделение.");

return;

}

var selectedDepartment = \_departments[dataGridDepartments.CurrentRow.Index];

if (selectedDepartment == null)

{

MessageBox.Show("Не удалось получить отделение.");

return;

}

var selectedWorker = comboBoxWorkers.SelectedItem as Worker;

if (selectedWorker == null)

{

MessageBox.Show("Выберите работника для добавления.");

return;

}

// Проверим, не добавлен ли уже

if (selectedDepartment.workers?.Any(w => w.id == selectedWorker.id) == true)

{

MessageBox.Show("Работник уже добавлен в это отделение.");

return;

}

selectedDepartment.workers ??= new List<Worker>();

selectedDepartment.workers.Add(selectedWorker);

try

{

var response = await \_client.PutAsJsonAsync(

$"api/department/update/{selectedDepartment.id}", selectedDepartment);

if (response.IsSuccessStatusCode)

{

MessageBox.Show("Работник добавлен в отделение.");

await LoadDepartmentsAsync(); // перезагружаем отделения с обновленными данными

LoadDepartmentWorkers(); // обновляем отображение

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка обновления: {response.StatusCode}\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения к API: {ex.Message}");

}

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\DepartmentsForm.Designer.cs ====

﻿

namespace DesctopSheduleManager

{

partial class DepartmentsForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

private DataGridView dataGridDepartments;

private TextBox txtName;

private Button btnAdd;

private Button btnUpdate;

private Button btnDelete;

private Button btnGenerateSchedule;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

dataGridDepartments = new DataGridView();

txtName = new TextBox();

btnAdd = new Button();

btnUpdate = new Button();

btnDelete = new Button();

btnGenerateSchedule = new Button();

buttonChangeShedule = new Button();

buttonShedule = new Button();

dataGridViewWorkers = new DataGridView();

buttonAddWorker = new Button();

comboBoxWorkers = new ComboBox();

((System.ComponentModel.ISupportInitialize)dataGridDepartments).BeginInit();

((System.ComponentModel.ISupportInitialize)dataGridViewWorkers).BeginInit();

SuspendLayout();

//

// dataGridDepartments

//

dataGridDepartments.AllowUserToAddRows = false;

dataGridDepartments.AllowUserToDeleteRows = false;

dataGridDepartments.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

dataGridDepartments.Location = new Point(12, 12);

dataGridDepartments.Name = "dataGridDepartments";

dataGridDepartments.ReadOnly = true;

dataGridDepartments.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

dataGridDepartments.Size = new Size(759, 200);

dataGridDepartments.TabIndex = 0;

dataGridDepartments.SelectionChanged += dataGridDepartments\_SelectionChanged;

//

// txtName

//

txtName.Location = new Point(12, 230);

txtName.Name = "txtName";

txtName.Size = new Size(200, 23);

txtName.TabIndex = 1;

//

// btnAdd

//

btnAdd.Location = new Point(220, 230);

btnAdd.Name = "btnAdd";

btnAdd.Size = new Size(75, 23);

btnAdd.TabIndex = 2;

btnAdd.Text = "Добавить";

btnAdd.UseVisualStyleBackColor = true;

btnAdd.Click += btnAdd\_Click;

//

// btnUpdate

//

btnUpdate.Location = new Point(300, 230);

btnUpdate.Name = "btnUpdate";

btnUpdate.Size = new Size(75, 23);

btnUpdate.TabIndex = 3;

btnUpdate.Text = "Обновить";

btnUpdate.UseVisualStyleBackColor = true;

btnUpdate.Click += btnUpdate\_Click;

//

// btnDelete

//

btnDelete.Location = new Point(380, 230);

btnDelete.Name = "btnDelete";

btnDelete.Size = new Size(75, 23);

btnDelete.TabIndex = 4;

btnDelete.Text = "Удалить";

btnDelete.UseVisualStyleBackColor = true;

btnDelete.Click += btnDelete\_Click;

//

// btnGenerateSchedule

//

btnGenerateSchedule.Location = new Point(554, 230);

btnGenerateSchedule.Name = "btnGenerateSchedule";

btnGenerateSchedule.Size = new Size(108, 58);

btnGenerateSchedule.TabIndex = 5;

btnGenerateSchedule.Text = "Распределить задачи автоматически";

btnGenerateSchedule.UseVisualStyleBackColor = true;

btnGenerateSchedule.Click += btnGenerateSchedule\_Click;

//

// buttonChangeShedule

//

buttonChangeShedule.Location = new Point(461, 230);

buttonChangeShedule.Name = "buttonChangeShedule";

buttonChangeShedule.Size = new Size(87, 44);

buttonChangeShedule.TabIndex = 6;

buttonChangeShedule.Text = "изменить расписание";

buttonChangeShedule.UseVisualStyleBackColor = true;

buttonChangeShedule.Click += buttonChangeShedule\_Click;

//

// buttonShedule

//

buttonShedule.Location = new Point(668, 230);

buttonShedule.Name = "buttonShedule";

buttonShedule.Size = new Size(103, 58);

buttonShedule.TabIndex = 7;

buttonShedule.Text = "посмотреть расписание за период";

buttonShedule.UseVisualStyleBackColor = true;

buttonShedule.Click += buttonShedule\_Click;

//

// dataGridViewWorkers

//

dataGridViewWorkers.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

dataGridViewWorkers.Location = new Point(12, 287);

dataGridViewWorkers.Name = "dataGridViewWorkers";

dataGridViewWorkers.Size = new Size(200, 126);

dataGridViewWorkers.TabIndex = 8;

//

// buttonAddWorker

//

buttonAddWorker.Location = new Point(220, 316);

buttonAddWorker.Name = "buttonAddWorker";

buttonAddWorker.Size = new Size(157, 23);

buttonAddWorker.TabIndex = 9;

buttonAddWorker.Text = "добавить работника";

buttonAddWorker.UseVisualStyleBackColor = true;

buttonAddWorker.Click += buttonAddWorker\_Click;

//

// comboBoxWorkers

//

comboBoxWorkers.FormattingEnabled = true;

comboBoxWorkers.Location = new Point(220, 287);

comboBoxWorkers.Name = "comboBoxWorkers";

comboBoxWorkers.Size = new Size(157, 23);

comboBoxWorkers.TabIndex = 10;

//

// DepartmentsForm

//

ClientSize = new Size(783, 425);

Controls.Add(comboBoxWorkers);

Controls.Add(buttonAddWorker);

Controls.Add(dataGridViewWorkers);

Controls.Add(buttonShedule);

Controls.Add(buttonChangeShedule);

Controls.Add(dataGridDepartments);

Controls.Add(txtName);

Controls.Add(btnAdd);

Controls.Add(btnUpdate);

Controls.Add(btnDelete);

Controls.Add(btnGenerateSchedule);

Name = "DepartmentsForm";

Text = "Департаменты";

Load += DepartmentsForm\_Load;

((System.ComponentModel.ISupportInitialize)dataGridDepartments).EndInit();

((System.ComponentModel.ISupportInitialize)dataGridViewWorkers).EndInit();

ResumeLayout(false);

PerformLayout();

}

#endregion

private Button buttonChangeShedule;

private Button buttonShedule;

private DataGridView dataGridViewWorkers;

private Button buttonAddWorker;

private ComboBox comboBoxWorkers;

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\HolidaysForm.cs ====

﻿using DesctopSheduleManager.Utilities;

using Models;

using System.Data;

using System.Net.Http.Json;

namespace DesctopSheduleManager

{

public partial class HolidaysForm : Form

{

private readonly HttpClient \_client;

private List<Holiday> \_holidays = new();

public HolidaysForm()

{

InitializeComponent();

\_client = ApiClient.Instance;

}

private async void HolidaysForm\_Load(object sender, EventArgs e)

{

await LoadHolidaysAsync();

}

private async Task LoadHolidaysAsync()

{

try

{

\_holidays = await \_client.GetFromJsonAsync<List<Holiday>>("api/holiday/get-all") ?? new();

dataGridHolidays.DataSource = \_holidays.Select(h => new

{

h.id,

h.name,

date = h.date.ToString("yyyy-MM-dd")

}).ToList();

ClearInputs();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки праздников: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ClearInputs()

{

txtName.Clear();

dtpDate.Value = DateTime.Today;

}

private void dataGridHolidays\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridHolidays.CurrentRow == null || dataGridHolidays.CurrentRow.Index < 0) return;

int index = dataGridHolidays.CurrentRow.Index;

if (index >= \_holidays.Count) return;

var selected = \_holidays[index];

txtName.Text = selected.name;

dtpDate.Value = selected.date.ToDateTime(TimeOnly.MinValue);

}

private async void btnAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(txtName.Text))

{

MessageBox.Show("Название праздника не может быть пустым.", "Валидация", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

var holiday = new Holiday

{

name = txtName.Text.Trim(),

date = DateOnly.FromDateTime(dtpDate.Value)

};

try

{

var response = await \_client.PostAsJsonAsync("api/holiday/add", holiday);

if (response.IsSuccessStatusCode)

{

await LoadHolidaysAsync();

MessageBox.Show("Праздник успешно добавлен.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при добавлении: {error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения: {ex.Message}", "Сетевая ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private async void btnUpdate\_Click(object sender, EventArgs e)

{

if (dataGridHolidays.CurrentRow == null)

{

MessageBox.Show("Выберите праздник для обновления.");

return;

}

int index = dataGridHolidays.CurrentRow.Index;

if (index >= \_holidays.Count) return;

if (string.IsNullOrWhiteSpace(txtName.Text))

{

MessageBox.Show("Название праздника не может быть пустым.", "Валидация", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

var id = \_holidays[index].id;

var updatedHoliday = new Holiday

{

id = id,

name = txtName.Text.Trim(),

date = DateOnly.FromDateTime(dtpDate.Value)

};

try

{

var response = await \_client.PutAsJsonAsync($"api/holiday/update/{id}", updatedHoliday);

if (response.IsSuccessStatusCode)

{

await LoadHolidaysAsync();

MessageBox.Show("Праздник обновлён.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при обновлении: {error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения: {ex.Message}", "Сетевая ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private async void btnDelete\_Click(object sender, EventArgs e)

{

if (dataGridHolidays.CurrentRow == null)

{

MessageBox.Show("Выберите праздник для удаления.");

return;

}

int index = dataGridHolidays.CurrentRow.Index;

if (index >= \_holidays.Count) return;

var id = \_holidays[index].id;

var confirm = MessageBox.Show("Удалить выбранный праздник?", "Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (confirm != DialogResult.Yes) return;

try

{

var response = await \_client.DeleteAsync($"api/holiday/delete/{id}");

if (response.IsSuccessStatusCode)

{

await LoadHolidaysAsync();

MessageBox.Show("Праздник удалён.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при удалении: {error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения: {ex.Message}", "Сетевая ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private async void btnRefresh\_Click(object sender, EventArgs e)

{

await LoadHolidaysAsync();

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\HolidaysForm.Designer.cs ====

﻿namespace DesctopSheduleManager

{

partial class HolidaysForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

private System.Windows.Forms.DataGridView dataGridHolidays;

private System.Windows.Forms.TextBox txtName;

private System.Windows.Forms.DateTimePicker dtpDate;

private System.Windows.Forms.Button btnAdd;

private System.Windows.Forms.Button btnUpdate;

private System.Windows.Forms.Button btnDelete;

private System.Windows.Forms.Button btnRefresh;

private System.Windows.Forms.Label lblName;

private System.Windows.Forms.Label lblDate;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.dataGridHolidays = new System.Windows.Forms.DataGridView();

this.txtName = new System.Windows.Forms.TextBox();

this.dtpDate = new System.Windows.Forms.DateTimePicker();

this.btnAdd = new System.Windows.Forms.Button();

this.btnUpdate = new System.Windows.Forms.Button();

this.btnDelete = new System.Windows.Forms.Button();

this.btnRefresh = new System.Windows.Forms.Button();

this.lblName = new System.Windows.Forms.Label();

this.lblDate = new System.Windows.Forms.Label();

((System.ComponentModel.ISupportInitialize)(this.dataGridHolidays)).BeginInit();

this.SuspendLayout();

// dataGridHolidays

this.dataGridHolidays.ColumnHeadersHeightSizeMode = System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;

this.dataGridHolidays.Location = new System.Drawing.Point(12, 12);

this.dataGridHolidays.Name = "dataGridHolidays";

this.dataGridHolidays.RowTemplate.Height = 25;

this.dataGridHolidays.Size = new System.Drawing.Size(460, 200);

this.dataGridHolidays.TabIndex = 0;

this.dataGridHolidays.SelectionChanged += new System.EventHandler(this.dataGridHolidays\_SelectionChanged);

// lblName

this.lblName.AutoSize = true;

this.lblName.Location = new System.Drawing.Point(12, 225);

this.lblName.Name = "lblName";

this.lblName.Size = new System.Drawing.Size(101, 15);

this.lblName.Text = "Название праздника";

// txtName

this.txtName.Location = new System.Drawing.Point(12, 243);

this.txtName.Name = "txtName";

this.txtName.Size = new System.Drawing.Size(250, 23);

this.txtName.TabIndex = 1;

// lblDate

this.lblDate.AutoSize = true;

this.lblDate.Location = new System.Drawing.Point(280, 225);

this.lblDate.Name = "lblDate";

this.lblDate.Size = new System.Drawing.Size(34, 15);

this.lblDate.Text = "Дата";

// dtpDate

this.dtpDate.Location = new System.Drawing.Point(280, 243);

this.dtpDate.Name = "dtpDate";

this.dtpDate.Size = new System.Drawing.Size(192, 23);

this.dtpDate.TabIndex = 2;

// btnAdd

this.btnAdd.Location = new System.Drawing.Point(12, 280);

this.btnAdd.Name = "btnAdd";

this.btnAdd.Size = new System.Drawing.Size(75, 30);

this.btnAdd.Text = "Добавить";

this.btnAdd.UseVisualStyleBackColor = true;

this.btnAdd.Click += new System.EventHandler(this.btnAdd\_Click);

// btnUpdate

this.btnUpdate.Location = new System.Drawing.Point(93, 280);

this.btnUpdate.Name = "btnUpdate";

this.btnUpdate.Size = new System.Drawing.Size(75, 30);

this.btnUpdate.Text = "Изменить";

this.btnUpdate.UseVisualStyleBackColor = true;

this.btnUpdate.Click += new System.EventHandler(this.btnUpdate\_Click);

// btnDelete

this.btnDelete.Location = new System.Drawing.Point(174, 280);

this.btnDelete.Name = "btnDelete";

this.btnDelete.Size = new System.Drawing.Size(75, 30);

this.btnDelete.Text = "Удалить";

this.btnDelete.UseVisualStyleBackColor = true;

this.btnDelete.Click += new System.EventHandler(this.btnDelete\_Click);

// btnRefresh

this.btnRefresh.Location = new System.Drawing.Point(397, 280);

this.btnRefresh.Name = "btnRefresh";

this.btnRefresh.Size = new System.Drawing.Size(75, 30);

this.btnRefresh.Text = "Обновить";

this.btnRefresh.UseVisualStyleBackColor = true;

this.btnRefresh.Click += new System.EventHandler(this.btnRefresh\_Click);

// HolidaysForm

this.ClientSize = new System.Drawing.Size(484, 331);

this.Controls.Add(this.btnRefresh);

this.Controls.Add(this.btnDelete);

this.Controls.Add(this.btnUpdate);

this.Controls.Add(this.btnAdd);

this.Controls.Add(this.dtpDate);

this.Controls.Add(this.txtName);

this.Controls.Add(this.lblDate);

this.Controls.Add(this.lblName);

this.Controls.Add(this.dataGridHolidays);

this.Name = "HolidaysForm";

this.Text = "Управление праздниками";

this.Load += new System.EventHandler(this.HolidaysForm\_Load);

((System.ComponentModel.ISupportInitialize)(this.dataGridHolidays)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\JobTitlesForm.cs ====

﻿using DesctopSheduleManager.Utilities;

using Models;

using System.Net.Http.Json;

namespace DesctopSheduleManager

{

public partial class JobTitlesForm : Form

{

private readonly HttpClient \_client;

private List<JobTitle> \_jobTitles = new();

public JobTitlesForm()

{

InitializeComponent();

\_client = ApiClient.Instance;

}

private async void JobTitlesForm\_Load(object sender, EventArgs e)

{

await LoadJobTitlesAsync();

}

private async Task LoadJobTitlesAsync()

{

try

{

var result = await \_client.GetFromJsonAsync<List<JobTitle>>("api/job-title/get-all");

if (result != null)

\_jobTitles = result;

else

\_jobTitles = new();

dataGridJobTitles.DataSource = \_jobTitles.Select(j => new { j.id, j.name }).ToList();

ClearInputs();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки данных: {ex.Message}");

}

}

private void ClearInputs()

{

txtName.Clear();

}

private void dataGridJobTitles\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridJobTitles.CurrentRow == null || dataGridJobTitles.CurrentRow.Index < 0) return;

int index = dataGridJobTitles.CurrentRow.Index;

if (index >= \_jobTitles.Count) return;

var selected = \_jobTitles[index];

txtName.Text = selected.name;

}

private async void btnAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(txtName.Text))

{

MessageBox.Show("Название должности не может быть пустым.");

return;

}

var jobTitle = new JobTitle { name = txtName.Text.Trim() };

try

{

var response = await \_client.PostAsJsonAsync("api/job-title/add", jobTitle);

if (response.IsSuccessStatusCode)

{

await LoadJobTitlesAsync();

MessageBox.Show("Должность добавлена.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при добавлении: {error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения: {ex.Message}");

}

}

private async void btnUpdate\_Click(object sender, EventArgs e)

{

if (dataGridJobTitles.CurrentRow == null)

{

MessageBox.Show("Выберите должность для обновления.");

return;

}

int index = dataGridJobTitles.CurrentRow.Index;

if (index >= \_jobTitles.Count) return;

if (string.IsNullOrWhiteSpace(txtName.Text))

{

MessageBox.Show("Название должности не может быть пустым.");

return;

}

var id = \_jobTitles[index].id;

var updatedJobTitle = new JobTitle

{

id = id,

name = txtName.Text.Trim()

};

try

{

var response = await \_client.PutAsJsonAsync($"api/job-title/update/{id}", updatedJobTitle);

if (response.IsSuccessStatusCode)

{

await LoadJobTitlesAsync();

MessageBox.Show("Должность обновлена.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при обновлении: {error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения: {ex.Message}");

}

}

private async void btnDelete\_Click(object sender, EventArgs e)

{

if (dataGridJobTitles.CurrentRow == null)

{

MessageBox.Show("Выберите должность для удаления.");

return;

}

int index = dataGridJobTitles.CurrentRow.Index;

if (index >= \_jobTitles.Count) return;

var id = \_jobTitles[index].id;

var result = MessageBox.Show("Удалить выбранную должность?", "Подтверждение", MessageBoxButtons.YesNo);

if (result != DialogResult.Yes) return;

try

{

var response = await \_client.DeleteAsync($"api/job-title/delete/{id}");

if (response.IsSuccessStatusCode)

{

await LoadJobTitlesAsync();

MessageBox.Show("Должность удалена.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при удалении: {error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения: {ex.Message}");

}

}

private async void btnRefresh\_Click(object sender, EventArgs e)

{

await LoadJobTitlesAsync();

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\JobTitlesForm.Designer.cs ====

﻿namespace DesctopSheduleManager

{

partial class JobTitlesForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

private DataGridView dataGridJobTitles;

private TextBox txtName;

private Label lblName;

private Button btnAdd;

private Button btnUpdate;

private Button btnDelete;

private Button btnRefresh;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.dataGridJobTitles = new DataGridView();

this.txtName = new TextBox();

this.lblName = new Label();

this.btnAdd = new Button();

this.btnUpdate = new Button();

this.btnDelete = new Button();

this.btnRefresh = new Button();

((System.ComponentModel.ISupportInitialize)(this.dataGridJobTitles)).BeginInit();

this.SuspendLayout();

// dataGridJobTitles

this.dataGridJobTitles.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

this.dataGridJobTitles.Location = new Point(12, 12);

this.dataGridJobTitles.Name = "dataGridJobTitles";

this.dataGridJobTitles.RowTemplate.Height = 25;

this.dataGridJobTitles.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

this.dataGridJobTitles.Size = new Size(360, 200);

this.dataGridJobTitles.TabIndex = 0;

this.dataGridJobTitles.SelectionChanged += new EventHandler(this.dataGridJobTitles\_SelectionChanged);

// lblName

this.lblName.AutoSize = true;

this.lblName.Location = new Point(12, 230);

this.lblName.Name = "lblName";

this.lblName.Size = new Size(95, 15);

this.lblName.TabIndex = 1;

this.lblName.Text = "Название должн.";

// txtName

this.txtName.Location = new Point(12, 250);

this.txtName.Name = "txtName";

this.txtName.Size = new Size(360, 23);

this.txtName.TabIndex = 2;

// btnAdd

this.btnAdd.Location = new Point(12, 290);

this.btnAdd.Name = "btnAdd";

this.btnAdd.Size = new Size(75, 30);

this.btnAdd.TabIndex = 3;

this.btnAdd.Text = "Добавить";

this.btnAdd.Click += new EventHandler(this.btnAdd\_Click);

// btnUpdate

this.btnUpdate.Location = new Point(102, 290);

this.btnUpdate.Name = "btnUpdate";

this.btnUpdate.Size = new Size(75, 30);

this.btnUpdate.TabIndex = 4;

this.btnUpdate.Text = "Обновить";

this.btnUpdate.Click += new EventHandler(this.btnUpdate\_Click);

// btnDelete

this.btnDelete.Location = new Point(192, 290);

this.btnDelete.Name = "btnDelete";

this.btnDelete.Size = new Size(75, 30);

this.btnDelete.TabIndex = 5;

this.btnDelete.Text = "Удалить";

this.btnDelete.Click += new EventHandler(this.btnDelete\_Click);

// btnRefresh

this.btnRefresh.Location = new Point(282, 290);

this.btnRefresh.Name = "btnRefresh";

this.btnRefresh.Size = new Size(90, 30);

this.btnRefresh.TabIndex = 6;

this.btnRefresh.Text = "Обновить список";

this.btnRefresh.Click += new EventHandler(this.btnRefresh\_Click);

// JobTitlesForm

this.ClientSize = new Size(384, 341);

this.Controls.Add(this.dataGridJobTitles);

this.Controls.Add(this.lblName);

this.Controls.Add(this.txtName);

this.Controls.Add(this.btnAdd);

this.Controls.Add(this.btnUpdate);

this.Controls.Add(this.btnDelete);

this.Controls.Add(this.btnRefresh);

this.Name = "JobTitlesForm";

this.Text = "Управление должностями";

this.Load += new EventHandler(this.JobTitlesForm\_Load);

((System.ComponentModel.ISupportInitialize)(this.dataGridJobTitles)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\LoginForm.cs ====

﻿using System.Text.Json;

using System.Text;

using DesctopSheduleManager.Forms;

using DesctopSheduleManager.Utilities;

namespace DesctopSheduleManager

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

}

private async void BtnLogin\_Click(object sender, EventArgs e)

{

string login = txtLogin.Text.Trim();

string password = txtPassword.Text;

if (string.IsNullOrWhiteSpace(login) || string.IsNullOrWhiteSpace(password))

{

lblStatus.Text = "Введите логин и пароль";

return;

}

var request = new

{

Login = login,

Password = password

};

var client = ApiClient.Instance;

var content = new StringContent(JsonSerializer.Serialize(request), Encoding.UTF8, "application/json");

try

{

var response = await client.PostAsync("api/auth/login", content);

if (response.IsSuccessStatusCode)

{

var responseContent = await response.Content.ReadAsStringAsync();

var json = JsonDocument.Parse(responseContent);

var sid = json.RootElement.GetProperty("sid").GetString();

SessionManager.SID = sid;

// перейти к главной форме или сообщить об успехе

MessageBox.Show("Вход выполнен успешно!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Hide();

var mainForm = new MainForm(); // например, главная форма

mainForm.Show();

}

else

{

lblStatus.Text = "Неверный логин или пароль";

}

}

catch (Exception ex)

{

lblStatus.Text = $"Ошибка: {ex.Message}";

}

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\LoginForm.Designer.cs ====

﻿namespace DesctopSheduleManager

{

partial class LoginForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

private TextBox txtLogin;

private TextBox txtPassword;

private Button btnLogin;

private Label lblStatus;

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.txtLogin = new TextBox();

this.txtPassword = new TextBox();

this.btnLogin = new Button();

this.lblStatus = new Label();

this.SuspendLayout();

// txtLogin

this.txtLogin.Location = new Point(30, 30);

this.txtLogin.Size = new Size(200, 23);

this.txtLogin.PlaceholderText = "Login";

// txtPassword

this.txtPassword.Location = new Point(30, 70);

this.txtPassword.Size = new Size(200, 23);

this.txtPassword.PasswordChar = '\*';

this.txtPassword.PlaceholderText = "Password";

// btnLogin

this.btnLogin.Location = new Point(30, 110);

this.btnLogin.Size = new Size(200, 30);

this.btnLogin.Text = "Login";

this.btnLogin.Click += new EventHandler(this.BtnLogin\_Click);

// lblStatus

this.lblStatus.Location = new Point(30, 150);

this.lblStatus.Size = new Size(200, 40);

this.lblStatus.ForeColor = Color.Red;

// LoginForm

this.ClientSize = new Size(280, 220);

this.Controls.Add(this.txtLogin);

this.Controls.Add(this.txtPassword);

this.Controls.Add(this.btnLogin);

this.Controls.Add(this.lblStatus);

this.Text = "Login";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\MainForm.cs ====

﻿using DesctopSheduleManager.Utilities;

namespace DesctopSheduleManager.Forms

{

public partial class MainForm : Form

{

private readonly HttpClient \_client;

public MainForm()

{

InitializeComponent();

\_client = ApiClient.Instance;

if (!string.IsNullOrEmpty(SessionManager.SID))

{

\_client.DefaultRequestHeaders.Add("sid", SessionManager.SID);

}

}

private void btnWorkers\_Click(object sender, EventArgs e)

{

var form = new WorkersForm();

form.ShowDialog();

}

private void btnDepartment\_Click(object sender, EventArgs e)

{

var form = new DepartmentsForm();

form.ShowDialog();

}

private void btnReceptions\_Click(object sender, EventArgs e)

{

var form = new ReceptionsForm();

form.ShowDialog();

}

private void btnReport\_Click(object sender, EventArgs e)

{

var form = new ReportForm();

form.ShowDialog();

}

private void btnJobTitles\_Click(object sender, EventArgs e)

{

var form = new JobTitlesForm();

form.ShowDialog();

}

private void btnHolidays\_Click(object sender, EventArgs e)

{

var form = new HolidaysForm();

form.ShowDialog();

}

private async void btnLogout\_Click(object sender, EventArgs e)

{

var confirm = MessageBox.Show("Вы уверены, что хотите выйти?", "Подтверждение", MessageBoxButtons.YesNo);

if (confirm != DialogResult.Yes) return;

try

{

var request = new HttpRequestMessage(HttpMethod.Post, "api/auth/logout");

request.Headers.Add("sid", SessionManager.SID);

await \_client.SendAsync(request);

}

catch

{

// Можно добавить логирование

}

Application.Exit();

}

private void buttonStats\_Click(object sender, EventArgs e)

{

var form = new StatsForm();

form.ShowDialog();

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\MainForm.Designer.cs ====

﻿namespace DesctopSheduleManager.Forms

{

partial class MainForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

private Button btnHolidays;

private Button btnLogout;

private Button btnJobTitles;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

btnHolidays = new Button();

btnLogout = new Button();

btnJobTitles = new Button();

btnWorkers = new Button();

btnDepartment = new Button();

btnReceptions = new Button();

btnReport = new Button();

buttonStats = new Button();

SuspendLayout();

//

// btnHolidays

//

btnHolidays.Location = new Point(50, 30);

btnHolidays.Name = "btnHolidays";

btnHolidays.Size = new Size(180, 40);

btnHolidays.TabIndex = 2;

btnHolidays.Text = "Управление праздниками";

btnHolidays.UseVisualStyleBackColor = true;

btnHolidays.Click += btnHolidays\_Click;

//

// btnLogout

//

btnLogout.Location = new Point(50, 389);

btnLogout.Name = "btnLogout";

btnLogout.Size = new Size(180, 40);

btnLogout.TabIndex = 1;

btnLogout.Text = "Выйти";

btnLogout.UseVisualStyleBackColor = true;

btnLogout.Click += btnLogout\_Click;

//

// btnJobTitles

//

btnJobTitles.Location = new Point(50, 76);

btnJobTitles.Name = "btnJobTitles";

btnJobTitles.Size = new Size(180, 40);

btnJobTitles.TabIndex = 0;

btnJobTitles.Text = "Должности";

btnJobTitles.UseVisualStyleBackColor = true;

btnJobTitles.Click += btnJobTitles\_Click;

//

// btnWorkers

//

btnWorkers.Location = new Point(50, 122);

btnWorkers.Name = "btnWorkers";

btnWorkers.Size = new Size(180, 40);

btnWorkers.TabIndex = 3;

btnWorkers.Text = "Работники";

btnWorkers.UseVisualStyleBackColor = true;

btnWorkers.Click += btnWorkers\_Click;

//

// btnDepartment

//

btnDepartment.Location = new Point(50, 168);

btnDepartment.Name = "btnDepartment";

btnDepartment.Size = new Size(180, 40);

btnDepartment.TabIndex = 4;

btnDepartment.Text = "Департаменты";

btnDepartment.UseVisualStyleBackColor = true;

btnDepartment.Click += btnDepartment\_Click;

//

// btnReceptions

//

btnReceptions.Location = new Point(50, 214);

btnReceptions.Name = "btnReceptions";

btnReceptions.Size = new Size(180, 40);

btnReceptions.TabIndex = 5;

btnReceptions.Text = "Записи";

btnReceptions.UseVisualStyleBackColor = true;

btnReceptions.Click += btnReceptions\_Click;

//

// btnReport

//

btnReport.Location = new Point(50, 260);

btnReport.Name = "btnReport";

btnReport.Size = new Size(180, 40);

btnReport.TabIndex = 6;

btnReport.Text = "Отчёты";

btnReport.UseVisualStyleBackColor = true;

btnReport.Click += btnReport\_Click;

//

// buttonStats

//

buttonStats.Location = new Point(50, 306);

buttonStats.Name = "buttonStats";

buttonStats.Size = new Size(180, 40);

buttonStats.TabIndex = 7;

buttonStats.Text = "статистика";

buttonStats.UseVisualStyleBackColor = true;

buttonStats.Click += buttonStats\_Click;

//

// MainForm

//

ClientSize = new Size(286, 441);

Controls.Add(buttonStats);

Controls.Add(btnReport);

Controls.Add(btnReceptions);

Controls.Add(btnDepartment);

Controls.Add(btnWorkers);

Controls.Add(btnJobTitles);

Controls.Add(btnLogout);

Controls.Add(btnHolidays);

Name = "MainForm";

Text = "Главное меню";

ResumeLayout(false);

}

#endregion

private Button btnWorkers;

private Button btnDepartment;

private Button btnReceptions;

private Button btnReport;

private Button buttonStats;

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\ReceptionsForm.cs ====

﻿using DesctopSheduleManager.Utilities;

using Microsoft.VisualBasic;

using Models;

using System.Net.Http.Json;

namespace DesctopSheduleManager

{

public partial class ReceptionsForm : Form

{

private readonly HttpClient \_client;

private List<Reception> \_receptions = new List<Reception>();

private List<JobTitle> \_jobTitles = new List<JobTitle>();

private List<Department> \_departments = new List<Department>();

private List<JobTitle> \_selectedJobTitles = new List<JobTitle>(); // Для выбранных должностей

public ReceptionsForm()

{

InitializeComponent();

\_client = ApiClient.Instance; // Используем ваш ApiClient

}

// Загружаем данные для комбобоксов (Персонал и Отделение)

private async Task LoadComboBoxDataAsync()

{

\_jobTitles = await \_client.GetFromJsonAsync<List<JobTitle>>("api/job-title/get-all");

\_departments = await \_client.GetFromJsonAsync<List<Department>>("api/department/get-all");

cmbJobTitle.DataSource = \_jobTitles;

cmbJobTitle.DisplayMember = "name"; // Отображаем имя должности

cmbJobTitle.ValueMember = "id"; // ID для отправки на сервер

cmbDepartment.DataSource = \_departments;

cmbDepartment.DisplayMember = "name"; // Отображаем имя департамента

cmbDepartment.ValueMember = "id"; // ID для отправки на сервер

}

// Загружаем список всех приёмов

private async Task LoadReceptionsAsync()

{

var receptions = await \_client.GetFromJsonAsync<List<Reception>>("api/reception/get-all");

if (receptions != null)

{

\_receptions = receptions;

dataGridReceptions.DataSource = \_receptions.Select(r => new

{

r.id,

ВремяПриёма = r.time.ToString(@"hh\:mm"),

Отделение = r.department?.name,

ТребуемыйПерсонал = r.requiredPersonnel != null ? string.Join(", ", r.requiredPersonnel.Select(p => p.name)) : "",

Дата = r.date?.ToString("yyyy-MM-dd") ?? "",

Начало = r.startTime?.ToString() ?? "",

Конец = r.endTime?.ToString() ?? "",

НазначенныйПерсонал = r.personnel != null ? string.Join(", ", r.personnel.Select(p => p.name)) : ""

}).ToList();

}

}

// Загружаем выбранный приём для редактирования

private void dataGridReceptions\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridReceptions.CurrentRow == null) return;

var selectedRow = dataGridReceptions.CurrentRow;

var selectedReception = \_receptions.FirstOrDefault(r => r.id == (int)selectedRow.Cells["id"].Value);

if (selectedReception == null) return;

// Заполняем данные

dateTimePickerTime.Value = DateTime.Today.Add(selectedReception.time);

cmbDepartment.SelectedValue = selectedReception.department?.id;

// Заполняем список должностей

\_selectedJobTitles = selectedReception.requiredPersonnel.ToList();

UpdateJobTitleList(); // Обновляем отображение должностей

}

// Обновляем список должностей в списке

private void UpdateJobTitleList()

{

lstJobTitles.Items.Clear();

foreach (var jobTitle in \_selectedJobTitles)

{

lstJobTitles.Items.Add(jobTitle.name);

}

}

// Добавляем выбранную должность в список requiredPersonnel

private void btnAddJobTitle\_Click(object sender, EventArgs e)

{

var selectedJobTitle = cmbJobTitle.SelectedItem as JobTitle;

if (selectedJobTitle != null && !\_selectedJobTitles.Contains(selectedJobTitle))

{

\_selectedJobTitles.Add(selectedJobTitle);

UpdateJobTitleList();

}

}

// Удаляем выбранную должность из списка requiredPersonnel

private void btnRemoveJobTitle\_Click(object sender, EventArgs e)

{

var selectedJobTitle = lstJobTitles.SelectedItem as string;

if (selectedJobTitle != null)

{

var jobTitleToRemove = \_selectedJobTitles.FirstOrDefault(jt => jt.name == selectedJobTitle);

if (jobTitleToRemove != null)

{

\_selectedJobTitles.Remove(jobTitleToRemove);

UpdateJobTitleList();

}

}

}

// Добавляем новый приём

private async void btnAdd\_Click(object sender, EventArgs e)

{

try

{

if (cmbDepartment.SelectedItem == null)

{

MessageBox.Show("Пожалуйста, выберите отделение.");

return;

}

if (\_selectedJobTitles.Count == 0)

{

MessageBox.Show("Пожалуйста, добавьте хотя бы одну требуемую должность.");

return;

}

TimeSpan time = dateTimePickerTime.Value.TimeOfDay;

if (time == TimeSpan.Zero)

{

MessageBox.Show("Пожалуйста, выберите корректное время.");

return;

}

var newReception = new Reception

{

time = time,

department = new Department { id = (int)cmbDepartment.SelectedValue },

requiredPersonnel = \_selectedJobTitles

};

var response = await \_client.PostAsJsonAsync("api/reception/add", newReception);

if (response.IsSuccessStatusCode)

{

MessageBox.Show("Прием успешно добавлен.");

await LoadReceptionsAsync();

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при добавлении: {response.ReasonPhrase}\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при добавлении: {ex.Message}");

}

}

private async void btnUpdate\_Click(object sender, EventArgs e)

{

try

{

if (dataGridReceptions.CurrentRow == null)

{

MessageBox.Show("Выберите прием для обновления.");

return;

}

var id = (int)dataGridReceptions.CurrentRow.Cells["id"].Value;

if (cmbDepartment.SelectedItem == null)

{

MessageBox.Show("Пожалуйста, выберите отделение.");

return;

}

if (\_selectedJobTitles.Count == 0)

{

MessageBox.Show("Пожалуйста, добавьте хотя бы одну требуемую должность.");

return;

}

TimeSpan time = dateTimePickerTime.Value.TimeOfDay;

if (time == TimeSpan.Zero)

{

MessageBox.Show("Пожалуйста, выберите корректное время.");

return;

}

var updatedReception = new Reception

{

id = id,

time = time,

department = new Department { id = (int)cmbDepartment.SelectedValue },

requiredPersonnel = \_selectedJobTitles,

date = DateOnly.FromDateTime(dtpDate.Value)

};

var response = await \_client.PutAsJsonAsync($"api/reception/update/{id}", updatedReception);

if (response.IsSuccessStatusCode)

{

MessageBox.Show("Прием успешно обновлен.");

await LoadReceptionsAsync();

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при обновлении: {response.ReasonPhrase}\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при обновлении: {ex.Message}");

}

}

private async void btnDelete\_Click(object sender, EventArgs e)

{

try

{

if (dataGridReceptions.CurrentRow == null)

{

MessageBox.Show("Выберите прием для удаления.");

return;

}

var id = (int)dataGridReceptions.CurrentRow.Cells["id"].Value;

var result = MessageBox.Show("Удалить выбранный приём?", "Подтверждение", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

{

var response = await \_client.DeleteAsync($"api/reception/delete/{id}");

if (response.IsSuccessStatusCode)

{

await LoadReceptionsAsync();

MessageBox.Show("Прием успешно удален.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при удалении: {response.ReasonPhrase}\n{error}");

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при удалении: {ex.Message}");

}

}

// Обработчик события загрузки формы

private async void ReceptionsForm\_Load(object sender, EventArgs e)

{

await LoadComboBoxDataAsync(); // Загружаем данные для комбобоксов

await LoadReceptionsAsync(); // Загружаем список приёмов

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\ReceptionsForm.Designer.cs ====

﻿

namespace DesctopSheduleManager

{

partial class ReceptionsForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

private DataGridView dataGridReceptions;

private ComboBox cmbDepartment;

private ComboBox cmbJobTitle;

private DateTimePicker dtpDate;

private Button btnAddJobTitle;

private Button btnRemoveJobTitle;

private ListBox lstJobTitles;

private Button btnAdd;

private Button btnUpdate;

private Button btnDelete;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

dataGridReceptions = new DataGridView();

cmbDepartment = new ComboBox();

cmbJobTitle = new ComboBox();

dtpDate = new DateTimePicker();

btnAddJobTitle = new Button();

btnRemoveJobTitle = new Button();

lstJobTitles = new ListBox();

btnAdd = new Button();

btnUpdate = new Button();

btnDelete = new Button();

dateTimePickerTime = new DateTimePicker();

((System.ComponentModel.ISupportInitialize)dataGridReceptions).BeginInit();

SuspendLayout();

//

// dataGridReceptions

//

dataGridReceptions.AllowUserToAddRows = false;

dataGridReceptions.AllowUserToDeleteRows = false;

dataGridReceptions.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

dataGridReceptions.Location = new Point(12, 12);

dataGridReceptions.Name = "dataGridReceptions";

dataGridReceptions.ReadOnly = true;

dataGridReceptions.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

dataGridReceptions.Size = new Size(500, 200);

dataGridReceptions.TabIndex = 10;

dataGridReceptions.SelectionChanged += dataGridReceptions\_SelectionChanged;

//

// cmbDepartment

//

cmbDepartment.DropDownStyle = ComboBoxStyle.DropDownList;

cmbDepartment.FormattingEnabled = true;

cmbDepartment.Location = new Point(12, 270);

cmbDepartment.Name = "cmbDepartment";

cmbDepartment.Size = new Size(200, 23);

cmbDepartment.TabIndex = 2;

//

// cmbJobTitle

//

cmbJobTitle.DropDownStyle = ComboBoxStyle.DropDownList;

cmbJobTitle.FormattingEnabled = true;

cmbJobTitle.Location = new Point(12, 310);

cmbJobTitle.Name = "cmbJobTitle";

cmbJobTitle.Size = new Size(200, 23);

cmbJobTitle.TabIndex = 3;

//

// dtpDate

//

dtpDate.Location = new Point(0, 0);

dtpDate.Name = "dtpDate";

dtpDate.Size = new Size(200, 23);

dtpDate.TabIndex = 0;

//

// btnAddJobTitle

//

btnAddJobTitle.Location = new Point(220, 310);

btnAddJobTitle.Name = "btnAddJobTitle";

btnAddJobTitle.Size = new Size(75, 23);

btnAddJobTitle.TabIndex = 4;

btnAddJobTitle.Text = "Добавить";

btnAddJobTitle.UseVisualStyleBackColor = true;

btnAddJobTitle.Click += btnAddJobTitle\_Click;

//

// btnRemoveJobTitle

//

btnRemoveJobTitle.Location = new Point(220, 350);

btnRemoveJobTitle.Name = "btnRemoveJobTitle";

btnRemoveJobTitle.Size = new Size(75, 23);

btnRemoveJobTitle.TabIndex = 6;

btnRemoveJobTitle.Text = "Удалить";

btnRemoveJobTitle.UseVisualStyleBackColor = true;

btnRemoveJobTitle.Click += btnRemoveJobTitle\_Click;

//

// lstJobTitles

//

lstJobTitles.FormattingEnabled = true;

lstJobTitles.ItemHeight = 15;

lstJobTitles.Location = new Point(12, 350);

lstJobTitles.Name = "lstJobTitles";

lstJobTitles.Size = new Size(200, 94);

lstJobTitles.TabIndex = 5;

//

// btnAdd

//

btnAdd.Location = new Point(12, 460);

btnAdd.Name = "btnAdd";

btnAdd.Size = new Size(75, 23);

btnAdd.TabIndex = 7;

btnAdd.Text = "Добавить";

btnAdd.UseVisualStyleBackColor = true;

btnAdd.Click += btnAdd\_Click;

//

// btnUpdate

//

btnUpdate.Location = new Point(100, 460);

btnUpdate.Name = "btnUpdate";

btnUpdate.Size = new Size(75, 23);

btnUpdate.TabIndex = 8;

btnUpdate.Text = "Обновить";

btnUpdate.UseVisualStyleBackColor = true;

btnUpdate.Click += btnUpdate\_Click;

//

// btnDelete

//

btnDelete.Location = new Point(190, 460);

btnDelete.Name = "btnDelete";

btnDelete.Size = new Size(75, 23);

btnDelete.TabIndex = 9;

btnDelete.Text = "Удалить";

btnDelete.UseVisualStyleBackColor = true;

btnDelete.Click += btnDelete\_Click;

//

// dateTimePickerTime

//

dateTimePickerTime.Format = DateTimePickerFormat.Time;

dateTimePickerTime.Location = new Point(12, 230);

dateTimePickerTime.Name = "dateTimePickerTime";

dateTimePickerTime.ShowUpDown = true;

dateTimePickerTime.Size = new Size(200, 23);

dateTimePickerTime.TabIndex = 11;

//

// ReceptionsForm

//

ClientSize = new Size(800, 500);

Controls.Add(dateTimePickerTime);

Controls.Add(btnDelete);

Controls.Add(btnUpdate);

Controls.Add(btnAdd);

Controls.Add(lstJobTitles);

Controls.Add(btnRemoveJobTitle);

Controls.Add(btnAddJobTitle);

Controls.Add(cmbJobTitle);

Controls.Add(cmbDepartment);

Controls.Add(dataGridReceptions);

Name = "ReceptionsForm";

Text = "Приёмы";

Load += ReceptionsForm\_Load;

((System.ComponentModel.ISupportInitialize)dataGridReceptions).EndInit();

ResumeLayout(false);

}

#endregion

private DateTimePicker dateTimePickerTime;

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\ReportForm.cs ====

﻿using ClosedXML.Excel;

using DesctopSheduleManager.Utilities;

using Models;

using System.Net.Http.Json;

namespace DesctopSheduleManager

{

public partial class ReportForm : Form

{

private readonly HttpClient \_client;

private List<Department> \_departments = new();

public ReportForm()

{

InitializeComponent();

\_client = ApiClient.Instance;

}

private async void ReportForm\_Load(object sender, EventArgs e)

{

await LoadDepartments();

}

private async Task LoadDepartments()

{

\_departments = await \_client.GetFromJsonAsync<List<Department>>("api/department/get-all");

cmbDepartment.Items.Clear();

foreach (var dept in \_departments)

cmbDepartment.Items.Add(dept);

cmbDepartment.DisplayMember = "name"; // отображается имя

}

private async void btnGenerate\_Click(object sender, EventArgs e)

{

if (cmbDepartment.SelectedItem is not Department selectedDept)

{

MessageBox.Show("Выберите департамент.");

return;

}

var start = DateOnly.FromDateTime(dtpStart.Value);

var end = DateOnly.FromDateTime(dtpEnd.Value);

try

{

var startStr = start.ToString("yyyy-MM-dd");

var endStr = end.ToString("yyyy-MM-dd");

var query = $"api/department/get-schedule-for-period?departmentId={selectedDept.id}&start={startStr}&end={endStr}";

var response = await ApiClient.Instance.GetAsync(query);

if (!response.IsSuccessStatusCode)

{

var errorContent = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка: {response.StatusCode}\n{errorContent}");

return;

}

var data = await response.Content.ReadFromJsonAsync<List<Reception>>();

if (data != null)

{

dataGridReport.DataSource = data.Select(r => new

{

r.id,

Дата = r.date?.ToString("yyyy-MM-dd"),

Начало = r.startTime?.ToString(),

Конец = r.endTime?.ToString(),

Отделение = r.department?.name,

ВремяПриёма = r.time.ToString(@"hh\:mm"),

Персонал = r.personnel != null ? string.Join(", ", r.personnel.Select(p => p.name)) : ""

}).ToList();

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка запроса:\n{ex.Message}");

}

}

private void btnExportExcel\_Click(object sender, EventArgs e)

{

// Проверяем, есть ли данные в DataGridView

if (dataGridReport.Rows.Count == 0)

{

MessageBox.Show("Нет данных для экспорта.");

return;

}

// Создание нового Excel документа

using (var workbook = new XLWorkbook())

{

var worksheet = workbook.Worksheets.Add("Отчёт");

// Заголовки для столбцов

worksheet.Cell(1, 1).Value = "ID";

worksheet.Cell(1, 2).Value = "Дата";

worksheet.Cell(1, 3).Value = "Начало";

worksheet.Cell(1, 4).Value = "Конец";

worksheet.Cell(1, 5).Value = "Отделение";

worksheet.Cell(1, 6).Value = "Время Приёма";

worksheet.Cell(1, 7).Value = "Персонал";

// Заполнение данных из DataGridView в Excel

int row = 2;

foreach (DataGridViewRow gridRow in dataGridReport.Rows)

{

if (gridRow.IsNewRow) continue; // Пропускаем строку добавления

worksheet.Cell(row, 1).Value = gridRow.Cells["id"].Value.ToString();

worksheet.Cell(row, 2).Value = gridRow.Cells["Дата"].Value.ToString();

worksheet.Cell(row, 3).Value = gridRow.Cells["Начало"].Value.ToString();

worksheet.Cell(row, 4).Value = gridRow.Cells["Конец"].Value.ToString();

worksheet.Cell(row, 5).Value = gridRow.Cells["Отделение"].Value.ToString();

worksheet.Cell(row, 6).Value = gridRow.Cells["ВремяПриёма"].Value.ToString();

worksheet.Cell(row, 7).Value = gridRow.Cells["Персонал"].Value.ToString();

row++;

}

worksheet.Columns().AdjustToContents();

// Сохранение файла

using (SaveFileDialog sfd = new SaveFileDialog()

{

Filter = "Excel Files|\*.xlsx",

FileName = "Отчёт.xlsx"

})

{

if (sfd.ShowDialog() == DialogResult.OK)

{

workbook.SaveAs(sfd.FileName);

MessageBox.Show("Файл успешно сохранён.");

}

}

}

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\ReportForm.Designer.cs ====

﻿

namespace DesctopSheduleManager

{

partial class ReportForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

private System.Windows.Forms.Label lblDepartment;

private System.Windows.Forms.ComboBox cmbDepartment;

private System.Windows.Forms.Label lblStart;

private System.Windows.Forms.DateTimePicker dtpStart;

private System.Windows.Forms.Label lblEnd;

private System.Windows.Forms.DateTimePicker dtpEnd;

private System.Windows.Forms.Button btnGenerate;

private System.Windows.Forms.DataGridView dataGridReport;

private Button btnExportExcel;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

lblDepartment = new Label();

cmbDepartment = new ComboBox();

lblStart = new Label();

dtpStart = new DateTimePicker();

lblEnd = new Label();

dtpEnd = new DateTimePicker();

btnGenerate = new Button();

dataGridReport = new DataGridView();

btnExportExcel = new Button();

((System.ComponentModel.ISupportInitialize)dataGridReport).BeginInit();

SuspendLayout();

//

// lblDepartment

//

lblDepartment.AutoSize = true;

lblDepartment.Location = new Point(20, 20);

lblDepartment.Name = "lblDepartment";

lblDepartment.Size = new Size(82, 15);

lblDepartment.TabIndex = 1;

lblDepartment.Text = "Департамент:";

//

// cmbDepartment

//

cmbDepartment.DropDownStyle = ComboBoxStyle.DropDownList;

cmbDepartment.FormattingEnabled = true;

cmbDepartment.Location = new Point(120, 16);

cmbDepartment.Name = "cmbDepartment";

cmbDepartment.Size = new Size(200, 23);

cmbDepartment.TabIndex = 2;

//

// lblStart

//

lblStart.AutoSize = true;

lblStart.Location = new Point(20, 60);

lblStart.Name = "lblStart";

lblStart.Size = new Size(77, 15);

lblStart.TabIndex = 3;

lblStart.Text = "Дата начала:";

//

// dtpStart

//

dtpStart.Location = new Point(120, 56);

dtpStart.Name = "dtpStart";

dtpStart.Size = new Size(200, 23);

dtpStart.TabIndex = 4;

//

// lblEnd

//

lblEnd.AutoSize = true;

lblEnd.Location = new Point(20, 100);

lblEnd.Name = "lblEnd";

lblEnd.Size = new Size(98, 15);

lblEnd.TabIndex = 5;

lblEnd.Text = "Дата окончания:";

//

// dtpEnd

//

dtpEnd.Location = new Point(120, 96);

dtpEnd.Name = "dtpEnd";

dtpEnd.Size = new Size(200, 23);

dtpEnd.TabIndex = 6;

//

// btnGenerate

//

btnGenerate.Location = new Point(350, 56);

btnGenerate.Name = "btnGenerate";

btnGenerate.Size = new Size(120, 30);

btnGenerate.TabIndex = 7;

btnGenerate.Text = "Сформировать";

btnGenerate.UseVisualStyleBackColor = true;

btnGenerate.Click += btnGenerate\_Click;

//

// dataGridReport

//

dataGridReport.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

dataGridReport.Location = new Point(20, 140);

dataGridReport.Name = "dataGridReport";

dataGridReport.Size = new Size(650, 300);

dataGridReport.TabIndex = 8;

//

// btnExportExcel

//

btnExportExcel.Location = new Point(550, 56);

btnExportExcel.Name = "btnExportExcel";

btnExportExcel.Size = new Size(120, 30);

btnExportExcel.TabIndex = 0;

btnExportExcel.Text = "Экспорт в Excel";

btnExportExcel.Click += btnExportExcel\_Click;

//

// ReportForm

//

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(700, 460);

Controls.Add(btnExportExcel);

Controls.Add(lblDepartment);

Controls.Add(cmbDepartment);

Controls.Add(lblStart);

Controls.Add(dtpStart);

Controls.Add(lblEnd);

Controls.Add(dtpEnd);

Controls.Add(btnGenerate);

Controls.Add(dataGridReport);

Name = "ReportForm";

Text = "Отчёт по приёмам";

Load += ReportForm\_Load;

((System.ComponentModel.ISupportInitialize)dataGridReport).EndInit();

ResumeLayout(false);

PerformLayout();

}

#endregion

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\ScheduleEditorForm.cs ====

﻿using Models;

using System.Data;

namespace DesctopSheduleManager.Forms

{

public partial class ScheduleEditorForm : Form

{

private readonly List<CheckBox> workingCheckboxes = new();

private readonly List<DateTimePicker> startPickers = new();

private readonly List<DateTimePicker> endPickers = new();

private readonly List<Label> dayLabels = new();

private readonly List<string> dayNames = new() { "Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Вс" };

private readonly List<WorkSchedule4Day?> schedule;

public List<WorkSchedule4Day?> UpdatedSchedule => schedule;

public ScheduleEditorForm(List<WorkSchedule4Day?>? existingSchedule)

{

InitializeComponent();

// 🧠 Инициализируем пустое расписание, если null или < 7

schedule = new List<WorkSchedule4Day?>();

for (int i = 0; i < 7; i++)

{

if (existingSchedule != null && i < existingSchedule.Count && existingSchedule[i] != null)

{

var ws = existingSchedule[i]!;

schedule.Add(new WorkSchedule4Day

{

isWorking = ws.isWorking,

startOfWork = ws.startOfWork,

endOfWork = ws.endOfWork

});

}

else

{

schedule.Add(new WorkSchedule4Day

{

isWorking = false,

startOfWork = TimeOnly.MinValue,

endOfWork = TimeOnly.MinValue

});

}

}

LoadScheduleIntoControls();

}

private void LoadScheduleIntoControls()

{

try

{

tableLayoutPanel1.RowCount = 7;

tableLayoutPanel1.ColumnCount = 4;

tableLayoutPanel1.Controls.Clear();

tableLayoutPanel1.AutoSize = true;

for (int i = 0; i < 7; i++)

{

var label = new Label { Text = dayNames[i], AutoSize = true, Anchor = AnchorStyles.Left };

var checkbox = new CheckBox { Checked = schedule[i]?.isWorking ?? false };

var start = new DateTimePicker { Format = DateTimePickerFormat.Time, ShowUpDown = true };

var end = new DateTimePicker { Format = DateTimePickerFormat.Time, ShowUpDown = true };

if (schedule[i] != null && schedule[i]!.isWorking)

{

start.Value = DateTime.Today.Add(schedule[i]!.startOfWork.ToTimeSpan());

end.Value = DateTime.Today.Add(schedule[i]!.endOfWork.ToTimeSpan());

}

dayLabels.Add(label);

workingCheckboxes.Add(checkbox);

startPickers.Add(start);

endPickers.Add(end);

tableLayoutPanel1.Controls.Add(label, 0, i);

tableLayoutPanel1.Controls.Add(checkbox, 1, i);

tableLayoutPanel1.Controls.Add(start, 2, i);

tableLayoutPanel1.Controls.Add(end, 3, i);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке расписания: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void buttonSave\_Click(object sender, EventArgs e)

{

try

{

for (int i = 0; i < 7; i++)

{

if (workingCheckboxes[i].Checked)

{

var start = TimeOnly.FromDateTime(startPickers[i].Value);

var end = TimeOnly.FromDateTime(endPickers[i].Value);

// ❗ Валидация времени

if (start >= end)

{

MessageBox.Show($"Ошибка в дне \"{dayNames[i]}\": начало позже окончания.", "Ошибка валидации", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

schedule[i] = new WorkSchedule4Day

{

isWorking = true,

startOfWork = start,

endOfWork = end

};

}

else

{

schedule[i] = new WorkSchedule4Day

{

isWorking = false,

startOfWork = TimeOnly.MinValue,

endOfWork = TimeOnly.MinValue

};

}

}

DialogResult = DialogResult.OK;

Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении расписания: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\ScheduleEditorForm.Designer.cs ====

﻿namespace DesctopSheduleManager.Forms

{

partial class ScheduleEditorForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

tableLayoutPanel1 = new TableLayoutPanel();

buttonSave = new Button();

tableLayoutPanel2 = new TableLayoutPanel();

tableLayoutPanel2.SuspendLayout();

SuspendLayout();

//

// tableLayoutPanel1

//

tableLayoutPanel1.AutoSize = true;

tableLayoutPanel1.ColumnCount = 4;

tableLayoutPanel1.ColumnStyles.Add(new ColumnStyle());

tableLayoutPanel1.ColumnStyles.Add(new ColumnStyle());

tableLayoutPanel1.ColumnStyles.Add(new ColumnStyle());

tableLayoutPanel1.ColumnStyles.Add(new ColumnStyle());

tableLayoutPanel1.Location = new Point(3, 3);

tableLayoutPanel1.Name = "tableLayoutPanel1";

tableLayoutPanel1.RowCount = 7;

tableLayoutPanel1.RowStyles.Add(new RowStyle());

tableLayoutPanel1.RowStyles.Add(new RowStyle());

tableLayoutPanel1.RowStyles.Add(new RowStyle());

tableLayoutPanel1.RowStyles.Add(new RowStyle());

tableLayoutPanel1.RowStyles.Add(new RowStyle());

tableLayoutPanel1.RowStyles.Add(new RowStyle());

tableLayoutPanel1.RowStyles.Add(new RowStyle());

tableLayoutPanel1.Size = new Size(0, 0);

tableLayoutPanel1.TabIndex = 0;

//

// buttonSave

//

buttonSave.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

buttonSave.Location = new Point(9, 400);

buttonSave.Name = "buttonSave";

buttonSave.Size = new Size(95, 23);

buttonSave.TabIndex = 1;

buttonSave.Text = "сохранить";

buttonSave.UseVisualStyleBackColor = true;

buttonSave.Click += buttonSave\_Click;

//

// tableLayoutPanel2

//

tableLayoutPanel2.Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right;

tableLayoutPanel2.ColumnCount = 2;

tableLayoutPanel2.ColumnStyles.Add(new ColumnStyle());

tableLayoutPanel2.ColumnStyles.Add(new ColumnStyle());

tableLayoutPanel2.Controls.Add(tableLayoutPanel1, 0, 0);

tableLayoutPanel2.Controls.Add(buttonSave, 1, 0);

tableLayoutPanel2.Location = new Point(12, 12);

tableLayoutPanel2.Name = "tableLayoutPanel2";

tableLayoutPanel2.RowCount = 1;

tableLayoutPanel2.RowStyles.Add(new RowStyle(SizeType.Percent, 100F));

tableLayoutPanel2.Size = new Size(776, 426);

tableLayoutPanel2.TabIndex = 0;

//

// ScheduleEditorForm

//

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(800, 450);

Controls.Add(tableLayoutPanel2);

Name = "ScheduleEditorForm";

Text = "ScheduleEditorForm";

tableLayoutPanel2.ResumeLayout(false);

tableLayoutPanel2.PerformLayout();

ResumeLayout(false);

}

#endregion

private TableLayoutPanel tableLayoutPanel1;

private Button buttonSave;

private TableLayoutPanel tableLayoutPanel2;

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\SheduleForm.cs ====

﻿using DesctopSheduleManager.Utilities;

using Models;

using System.Net.Http.Json;

namespace DesctopSheduleManager.Forms

{

public partial class SheduleForm : Form

{

// на вьюхе dataGrid - датагридвью dtpStart - дататаймпикер dtpEnd - дататаймпикер btnGenerate - кнопка

private readonly HttpClient \_client;

private int \_departmentId;

public SheduleForm(int departmentId)

{

\_departmentId = departmentId;

\_client = ApiClient.Instance; // Используем ваш ApiClient

InitializeComponent();

}

private async void btnGenerate\_Click(object sender, EventArgs e)

{

try

{

var start = DateOnly.FromDateTime(dtpStart.Value.Date);

var end = DateOnly.FromDateTime(dtpEnd.Value.Date);

Department department = await \_client.GetFromJsonAsync<Department>($"api/department/get/{\_departmentId}");

List<Worker> workers = department.workers;

List<WorkSchedule4Day> departmentSchedule = department.workSchedules;

List<Holiday> holidays = await \_client.GetFromJsonAsync<List<Holiday>>("api/holiday/get-all");

var allDays = Enumerable.Range(0, end.DayNumber - start.DayNumber + 1)

.Select(offset => start.AddDays(offset))

.ToList();

dataGrid.Columns.Clear();

dataGrid.Rows.Clear();

// Добавим колонки

dataGrid.Columns.Add("Name", "Сотрудник");

foreach (var day in allDays)

{

var col = new DataGridViewTextBoxColumn();

col.Name = day.ToString();

col.HeaderText = $"{day:dd.MM} ({day.DayOfWeek.ToString()[..2]})";

dataGrid.Columns.Add(col);

}

// Добавим строки с ячейками

foreach (var worker in workers)

{

worker.workSchedules.Sort((a, b) => a.id.CompareTo(b.id));

var rowIndex = dataGrid.Rows.Add();

var row = dataGrid.Rows[rowIndex];

row.Cells[0].Value = worker.name;

for (int i = 0; i < allDays.Count; i++)

{

var day = allDays[i];

var cell = row.Cells[i + 1];

bool isInteractive = false;

var weekday = (int)day.DayOfWeek;

if (weekday == 0) weekday = 7;

// Праздники

if (holidays.Any(h => h.date == day))

{

cell.Style.BackColor = Color.LightCoral;

cell.Value = "Праздник";

continue;

}

// Отдел не работает

var depSchedule = departmentSchedule.ElementAtOrDefault(weekday - 1);

if (depSchedule == null || !depSchedule.isWorking)

{

cell.Style.BackColor = Color.Red;

cell.Value = "Выходной";

continue;

}

// Отпуск

if (worker.vacations?.Any(v => v.days.Any(d => d.date == day)) == true)

{

cell.Style.BackColor = Color.Black;

cell.Style.ForeColor = Color.White;

cell.Value = "Отпуск";

continue;

}

// Личное расписание

if (worker.workSchedules?.Count > 0)

{

var scheduleIndex = day.DayNumber % worker.workSchedules.Count;

var ws = worker.workSchedules[scheduleIndex];

if (!ws.isWorking)

{

cell.Style.BackColor = Color.DarkRed;

cell.Value = "Не работает";

continue;

}

// Рабочий день

cell.Style.BackColor = Color.LightGreen;

isInteractive = true;

}

// Ячейка интерактивна только если день рабочий

cell.Tag = new { day, worker, interactive = isInteractive };

}

}

// Обработчик двойного клика

dataGrid.CellDoubleClick += (s, args) =>

{

if (args.RowIndex >= 0 && args.ColumnIndex > 0)

{

var cell = dataGrid.Rows[args.RowIndex].Cells[args.ColumnIndex];

if (cell.Tag is not { } tagObj) return;

var tagType = tagObj.GetType();

var interactiveProp = tagType.GetProperty("interactive");

if (interactiveProp == null || !(bool)interactiveProp.GetValue(tagObj)) return;

var dayProp = tagType.GetProperty("day");

var workerProp = tagType.GetProperty("worker");

if (dayProp == null || workerProp == null) return;

var day = (DateOnly)dayProp.GetValue(tagObj);

var worker = (Worker)workerProp.GetValue(tagObj);

var workSchedule = worker.workSchedules.Count > 0

? worker.workSchedules[day.DayNumber % worker.workSchedules.Count]

: null;

var weekday = (int)day.DayOfWeek;

if (weekday == 0) weekday = 7;

var depSchedule = department.workSchedules[weekday - 1];

var view = new ViewReceptionsForm(department, day, worker, workSchedule, depSchedule);

view.ShowDialog();

}

};

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при генерации расписания: " + ex.Message);

}

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\SheduleForm.Designer.cs ====

﻿namespace DesctopSheduleManager.Forms

{

partial class SheduleForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

lblStart = new Label();

dtpStart = new DateTimePicker();

lblEnd = new Label();

dtpEnd = new DateTimePicker();

btnGenerate = new Button();

dataGrid = new DataGridView();

((System.ComponentModel.ISupportInitialize)dataGrid).BeginInit();

SuspendLayout();

//

// lblStart

//

lblStart.AutoSize = true;

lblStart.Location = new Point(75, 37);

lblStart.Name = "lblStart";

lblStart.Size = new Size(77, 15);

lblStart.TabIndex = 9;

lblStart.Text = "Дата начала:";

//

// dtpStart

//

dtpStart.Location = new Point(175, 33);

dtpStart.Name = "dtpStart";

dtpStart.Size = new Size(200, 23);

dtpStart.TabIndex = 10;

//

// lblEnd

//

lblEnd.AutoSize = true;

lblEnd.Location = new Point(75, 77);

lblEnd.Name = "lblEnd";

lblEnd.Size = new Size(98, 15);

lblEnd.TabIndex = 11;

lblEnd.Text = "Дата окончания:";

//

// dtpEnd

//

dtpEnd.Location = new Point(175, 73);

dtpEnd.Name = "dtpEnd";

dtpEnd.Size = new Size(200, 23);

dtpEnd.TabIndex = 12;

//

// btnGenerate

//

btnGenerate.Location = new Point(405, 33);

btnGenerate.Name = "btnGenerate";

btnGenerate.Size = new Size(120, 30);

btnGenerate.TabIndex = 13;

btnGenerate.Text = "Сформировать";

btnGenerate.UseVisualStyleBackColor = true;

btnGenerate.Click += btnGenerate\_Click;

//

// dataGrid

//

dataGrid.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

dataGrid.Location = new Point(75, 117);

dataGrid.Name = "dataGrid";

dataGrid.Size = new Size(650, 300);

dataGrid.TabIndex = 14;

//

// SheduleForm

//

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(800, 473);

Controls.Add(lblStart);

Controls.Add(dtpStart);

Controls.Add(lblEnd);

Controls.Add(dtpEnd);

Controls.Add(btnGenerate);

Controls.Add(dataGrid);

Name = "SheduleForm";

Text = "Shedule";

((System.ComponentModel.ISupportInitialize)dataGrid).EndInit();

ResumeLayout(false);

PerformLayout();

}

#endregion

private Label lblStart;

private DateTimePicker dtpStart;

private Label lblEnd;

private DateTimePicker dtpEnd;

private Button btnGenerate;

private DataGridView dataGrid;

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\StatsForm.cs ====

﻿using DesctopSheduleManager.Utilities;

using Models;

using System.Net.Http.Json;

using System.Windows.Forms.DataVisualization.Charting;

namespace DesctopSheduleManager.Forms

{

public partial class StatsForm : Form

{

private HttpClient \_client;

// на форме: groupBoxTypes:radioButtonByDepartments, radioButtonByWorkers; dtpStart; dtpEnd; buttonGenerate

public StatsForm()

{

\_client = ApiClient.Instance;

InitializeComponent();

}

private async void buttonGenerate\_Click(object sender, EventArgs e)

{

if (radioButtonByWorkers.Checked)

{

flowLayoutPanel.Controls.Clear();

flowLayoutPanel.AutoScroll = true;

var start = DateOnly.FromDateTime(dtpStart.Value.Date);

var end = DateOnly.FromDateTime(dtpEnd.Value.Date);

var allDays = Enumerable.Range(0, end.DayNumber - start.DayNumber + 1)

.Select(offset => start.AddDays(offset))

.ToList();

List<Reception> receptions = await \_client.GetFromJsonAsync<List<Reception>>("api/reception/get-all");

List<Holiday> holidays = await \_client.GetFromJsonAsync<List<Holiday>>("api/holiday/get-all");

List<Department> departments = await \_client.GetFromJsonAsync<List<Department>>("api/department/get-all");

List<Worker> workers = await \_client.GetFromJsonAsync<List<Worker>>("api/worker/get-all");

foreach (var worker in workers)

{

int workingDays = 0;

int offDays = 0;

int vacationDays = 0;

foreach (var day in allDays)

{

var weekday = (int)day.DayOfWeek;

if (weekday == 0) weekday = 7; // Sunday → 7

bool isHoliday = holidays.Any(h => h.date == day);

bool isDepartmentDayOff = departments

.Where(

d => d.workers != null &&

d.workers

.Select(w => w.id)

.Contains(worker.id)

)

.Any(

d => d.workSchedules

.ElementAtOrDefault(weekday - 1)?.isWorking == false

);

bool isVacation = worker.vacations?.Any(v => v.days.Any(d => d.date == day)) == true;

// Проверяем личное расписание

var ws = worker.workSchedules.Count > 0

? worker.workSchedules[day.DayNumber % worker.workSchedules.Count]

: null;

bool isPersonalDayOff = ws != null && !ws.isWorking;

if (isHoliday || isDepartmentDayOff || isVacation || isPersonalDayOff)

offDays++;

else

workingDays++;

if (isVacation) vacationDays++;

}

int receptionsCount = receptions.Count(r =>

r.personnel?.Any(p => p.id == worker.id) == true &&

r.date >= start && r.date <= end

);

// График для одного сотрудника

Chart chart = new Chart

{

Width = 400,

Height = 250,

BackColor = Color.White

};

Legend legend = new Legend

{

Docking = Docking.Right,

Alignment = StringAlignment.Far

};

chart.Legends.Add(legend);

ChartArea area = new ChartArea();

area.AxisX.Interval = 0;

area.AxisX.LabelStyle.Angle = -45;

chart.ChartAreas.Add(area);

Series series1 = new Series("Рабочие")

{

ChartType = SeriesChartType.Column

};

Series series2 = new Series("Нерабочие")

{

ChartType = SeriesChartType.Column

};

Series series3 = new Series("Отпуск")

{

ChartType = SeriesChartType.Column

};

Series series4 = new Series("Приёмы")

{

ChartType = SeriesChartType.Column

};

series1.Points.AddXY("", workingDays);

series2.Points.AddXY("", offDays);

series3.Points.AddXY("", vacationDays);

series4.Points.AddXY("", receptionsCount);

chart.Series.Add(series1);

chart.Series.Add(series2);

chart.Series.Add(series3);

chart.Series.Add(series4);

chart.Titles.Add(worker.name ?? "Без имени");

flowLayoutPanel.Controls.Add(chart);

}

}

else if (radioButtonByDepartments.Checked)

{

flowLayoutPanel.Controls.Clear();

flowLayoutPanel.AutoScroll = true;

var start = DateOnly.FromDateTime(dtpStart.Value.Date);

var end = DateOnly.FromDateTime(dtpEnd.Value.Date);

var allDays = Enumerable.Range(0, end.DayNumber - start.DayNumber + 1)

.Select(offset => start.AddDays(offset))

.ToList();

List<Reception> receptions = await \_client.GetFromJsonAsync<List<Reception>>("api/reception/get-all");

List<Holiday> holidays = await \_client.GetFromJsonAsync<List<Holiday>>("api/holiday/get-all");

List<Department> departments = await \_client.GetFromJsonAsync<List<Department>>("api/department/get-all");

foreach (var department in departments)

{

int workingDays = 0;

int offDays = 0;

int holidayDays = 0;

foreach (var day in allDays)

{

var weekday = (int)day.DayOfWeek;

if (weekday == 0) weekday = 7; // Sunday → 7

bool isHoliday = holidays.Any(h => h.date == day);

bool isDepDayOff = department.workSchedules.ElementAtOrDefault(weekday - 1)?.isWorking == false;

if (isHoliday)

holidayDays++;

if (isDepDayOff || isHoliday)

offDays++;

else

workingDays++;

}

int receptionsCount = receptions.Count(r =>

r.department?.id == department.id &&

r.date >= start && r.date <= end

);

// График для одного отделения

Chart chart = new Chart

{

Width = 400,

Height = 250,

BackColor = Color.White

};

Legend legend = new Legend

{

Docking = Docking.Right,

Alignment = StringAlignment.Far

};

chart.Legends.Add(legend);

ChartArea area = new ChartArea();

area.AxisX.Interval = 0;

area.AxisX.LabelStyle.Angle = -45;

chart.ChartAreas.Add(area);

Series series1 = new Series("Рабочие дни") { ChartType = SeriesChartType.Column };

Series series2 = new Series("Нерабочие дни") { ChartType = SeriesChartType.Column };

Series series3 = new Series("Праздники") { ChartType = SeriesChartType.Column };

Series series4 = new Series("Приёмы") { ChartType = SeriesChartType.Column };

series1.Points.AddXY("", workingDays);

series2.Points.AddXY("", offDays);

series3.Points.AddXY("", holidayDays);

series4.Points.AddXY("", receptionsCount);

chart.Series.Add(series1);

chart.Series.Add(series2);

chart.Series.Add(series3);

chart.Series.Add(series4);

chart.Titles.Add(department.name ?? "Без названия");

flowLayoutPanel.Controls.Add(chart);

}

}

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\StatsForm.Designer.cs ====

﻿namespace DesctopSheduleManager.Forms

{

partial class StatsForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

radioButtonByDepartments = new RadioButton();

groupBoxTypes = new GroupBox();

radioButtonByWorkers = new RadioButton();

buttonGenerate = new Button();

lblStart = new Label();

dtpStart = new DateTimePicker();

lblEnd = new Label();

dtpEnd = new DateTimePicker();

flowLayoutPanel = new FlowLayoutPanel();

groupBoxTypes.SuspendLayout();

SuspendLayout();

//

// radioButtonByDepartments

//

radioButtonByDepartments.AutoSize = true;

radioButtonByDepartments.Location = new Point(6, 22);

radioButtonByDepartments.Name = "radioButtonByDepartments";

radioButtonByDepartments.Size = new Size(108, 19);

radioButtonByDepartments.TabIndex = 1;

radioButtonByDepartments.TabStop = true;

radioButtonByDepartments.Text = "по отделениям";

radioButtonByDepartments.UseVisualStyleBackColor = true;

//

// groupBoxTypes

//

groupBoxTypes.Controls.Add(radioButtonByWorkers);

groupBoxTypes.Controls.Add(radioButtonByDepartments);

groupBoxTypes.Location = new Point(12, 12);

groupBoxTypes.Name = "groupBoxTypes";

groupBoxTypes.Size = new Size(131, 100);

groupBoxTypes.TabIndex = 2;

groupBoxTypes.TabStop = false;

groupBoxTypes.Text = "типы";

//

// radioButtonByWorkers

//

radioButtonByWorkers.AutoSize = true;

radioButtonByWorkers.Location = new Point(6, 47);

radioButtonByWorkers.Name = "radioButtonByWorkers";

radioButtonByWorkers.Size = new Size(109, 19);

radioButtonByWorkers.TabIndex = 2;

radioButtonByWorkers.TabStop = true;

radioButtonByWorkers.Text = "по работникам";

radioButtonByWorkers.UseVisualStyleBackColor = true;

//

// buttonGenerate

//

buttonGenerate.Location = new Point(326, 89);

buttonGenerate.Name = "buttonGenerate";

buttonGenerate.Size = new Size(125, 23);

buttonGenerate.TabIndex = 3;

buttonGenerate.Text = "сгенерировать";

buttonGenerate.UseVisualStyleBackColor = true;

buttonGenerate.Click += buttonGenerate\_Click;

//

// lblStart

//

lblStart.AutoSize = true;

lblStart.Location = new Point(151, 16);

lblStart.Name = "lblStart";

lblStart.Size = new Size(77, 15);

lblStart.TabIndex = 7;

lblStart.Text = "Дата начала:";

//

// dtpStart

//

dtpStart.Location = new Point(251, 12);

dtpStart.Name = "dtpStart";

dtpStart.Size = new Size(200, 23);

dtpStart.TabIndex = 8;

//

// lblEnd

//

lblEnd.AutoSize = true;

lblEnd.Location = new Point(151, 56);

lblEnd.Name = "lblEnd";

lblEnd.Size = new Size(98, 15);

lblEnd.TabIndex = 9;

lblEnd.Text = "Дата окончания:";

//

// dtpEnd

//

dtpEnd.Location = new Point(251, 52);

dtpEnd.Name = "dtpEnd";

dtpEnd.Size = new Size(200, 23);

dtpEnd.TabIndex = 10;

//

// flowLayoutPanel

//

flowLayoutPanel.AutoScroll = true;

flowLayoutPanel.AutoSize = true;

flowLayoutPanel.Location = new Point(12, 118);

flowLayoutPanel.Name = "flowLayoutPanel";

flowLayoutPanel.Size = new Size(439, 320);

flowLayoutPanel.TabIndex = 11;

//

// StatsForm

//

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(472, 450);

Controls.Add(flowLayoutPanel);

Controls.Add(lblStart);

Controls.Add(dtpStart);

Controls.Add(lblEnd);

Controls.Add(dtpEnd);

Controls.Add(buttonGenerate);

Controls.Add(groupBoxTypes);

Name = "StatsForm";

Text = "StatsForm";

groupBoxTypes.ResumeLayout(false);

groupBoxTypes.PerformLayout();

ResumeLayout(false);

PerformLayout();

}

#endregion

private RadioButton radioButton1;

private GroupBox groupBox1;

private RadioButton radioButton2;

private RadioButton radioButtonByDepartments;

private GroupBox groupBoxTypes;

private RadioButton radioButtonByWorkers;

private Button buttonGenerate;

private Label lblStart;

private DateTimePicker dtpStart;

private Label lblEnd;

private DateTimePicker dtpEnd;

private FlowLayoutPanel flowLayoutPanel;

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\VacationEditorForm.cs ====

﻿using Models;

using System.ComponentModel;

using System.Windows.Forms;

namespace DesctopSheduleManager.Forms

{

public partial class VacationEditorForm : Form

{

public class VacationPeriod

{

public DateOnly Start { get; set; }

public DateOnly End { get; set; }

}

private List<Vacation> \_vacations;

private BindingList<VacationPeriod> \_vacationPeriods;

public List<Vacation?> UpdatedVacations => \_vacationPeriods

.Select(period =>

{

var days = Enumerable.Range(0, (period.End.DayNumber - period.Start.DayNumber) + 1)

.Select(offset => new DayOff { date = period.Start.AddDays(offset) })

.ToList();

return new Vacation { days = days };

}).ToList();

public VacationEditorForm(List<Vacation> existingVacations)

{

InitializeComponent();

// Преобразуем существующие отпуска в пары (Start, End)

var vacationPeriods = existingVacations?

.Select(v =>

{

var sortedDays = v.days.OrderBy(d => d.date).ToList();

if (sortedDays.Count == 0)

return null;

return new VacationPeriod

{

Start = sortedDays.First().date,

End = sortedDays.Last().date

};

})

.Where(p => p != null)

.ToList() ?? new();

\_vacationPeriods = new BindingList<VacationPeriod>(vacationPeriods);

dataGridViewDays.DataSource = \_vacationPeriods;

// Настрой заголовки

dataGridViewDays.Columns["Start"].HeaderText = "Начало отпуска";

dataGridViewDays.Columns["End"].HeaderText = "Конец отпуска";

}

private void buttonAdd\_Click(object sender, EventArgs e)

{

DateOnly start = DateOnly.FromDateTime(dateTimePickerStart.Value.Date);

DateOnly end = DateOnly.FromDateTime(dateTimePickerEnd.Value.Date);

if (end < start)

{

MessageBox.Show("Дата окончания не может быть раньше даты начала.");

return;

}

\_vacationPeriods.Add(new VacationPeriod { Start = start, End = end });

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

if (dataGridViewDays.CurrentRow?.DataBoundItem is VacationPeriod selectedPeriod)

{

\_vacationPeriods.Remove(selectedPeriod);

}

}

private void buttonSave\_Click(object sender, EventArgs e)

{

DialogResult = DialogResult.OK;

Close();

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\VacationEditorForm.Designer.cs ====

﻿namespace DesctopSheduleManager.Forms

{

partial class VacationEditorForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

buttonSave = new Button();

dateTimePickerStart = new DateTimePicker();

buttonAdd = new Button();

buttonDelete = new Button();

dataGridViewDays = new DataGridView();

dateTimePickerEnd = new DateTimePicker();

((System.ComponentModel.ISupportInitialize)dataGridViewDays).BeginInit();

SuspendLayout();

//

// buttonSave

//

buttonSave.Location = new Point(383, 139);

buttonSave.Name = "buttonSave";

buttonSave.Size = new Size(75, 23);

buttonSave.TabIndex = 0;

buttonSave.Text = "сохранить";

buttonSave.UseVisualStyleBackColor = true;

buttonSave.Click += buttonSave\_Click;

//

// dateTimePickerStart

//

dateTimePickerStart.Location = new Point(258, 12);

dateTimePickerStart.Name = "dateTimePickerStart";

dateTimePickerStart.Size = new Size(200, 23);

dateTimePickerStart.TabIndex = 1;

//

// buttonAdd

//

buttonAdd.Location = new Point(383, 70);

buttonAdd.Name = "buttonAdd";

buttonAdd.Size = new Size(75, 23);

buttonAdd.TabIndex = 2;

buttonAdd.Text = "Добавить";

buttonAdd.UseVisualStyleBackColor = true;

buttonAdd.Click += buttonAdd\_Click;

//

// buttonDelete

//

buttonDelete.Location = new Point(258, 70);

buttonDelete.Name = "buttonDelete";

buttonDelete.Size = new Size(75, 23);

buttonDelete.TabIndex = 3;

buttonDelete.Text = "Удалить";

buttonDelete.UseVisualStyleBackColor = true;

buttonDelete.Click += buttonDelete\_Click;

//

// dataGridViewDays

//

dataGridViewDays.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

dataGridViewDays.Location = new Point(12, 12);

dataGridViewDays.Name = "dataGridViewDays";

dataGridViewDays.Size = new Size(240, 150);

dataGridViewDays.TabIndex = 4;

//

// dateTimePickerEnd

//

dateTimePickerEnd.Location = new Point(258, 41);

dateTimePickerEnd.Name = "dateTimePickerEnd";

dateTimePickerEnd.Size = new Size(200, 23);

dateTimePickerEnd.TabIndex = 5;

//

// VacationEditorForm

//

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(470, 177);

Controls.Add(dateTimePickerEnd);

Controls.Add(dataGridViewDays);

Controls.Add(buttonDelete);

Controls.Add(buttonAdd);

Controls.Add(dateTimePickerStart);

Controls.Add(buttonSave);

Name = "VacationEditorForm";

Text = "VacationEditorForm";

((System.ComponentModel.ISupportInitialize)dataGridViewDays).EndInit();

ResumeLayout(false);

}

#endregion

private Button buttonSave;

private DateTimePicker dateTimePickerStart;

private Button buttonAdd;

private Button buttonDelete;

private DataGridView dataGridViewDays;

private DateTimePicker dateTimePickerEnd;

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\ViewReceptionsForm.cs ====

﻿using DesctopSheduleManager.Utilities;

using Models;

using Models.DTO;

using System.Net.Http.Json;

using static System.Runtime.InteropServices.Marshalling.IIUnknownCacheStrategy;

namespace DesctopSheduleManager.Forms

{

public partial class ViewReceptionsForm : Form

{

private readonly HttpClient \_client;

private Department \_department;

private DateOnly \_day;

private Worker \_worker;

private WorkerWorkSchedule4Day \_workerSchedule;

private WorkSchedule4Day \_deptSchedule;

private TimeOnly \_startWork;

private TimeOnly \_endWork;

private List<Reception> \_allReceptions;

public ViewReceptionsForm(

Department department, DateOnly day, Worker worker,

WorkerWorkSchedule4Day workerWorkSchedule4Day, WorkSchedule4Day departmentSchedule)

{

InitializeComponent();

\_client = ApiClient.Instance;

\_department = department;

\_day = day;

\_worker = worker;

\_workerSchedule = workerWorkSchedule4Day;

\_deptSchedule = departmentSchedule;

// Общее рабочее время: пересечение интервалов

\_startWork = TimeOnly.FromTimeSpan(

TimeSpan.FromTicks(Math.Max(

\_workerSchedule.startOfWork.Ticks,

\_deptSchedule.startOfWork.Ticks

)));

\_endWork = TimeOnly.FromTimeSpan(

TimeSpan.FromTicks(Math.Min(

\_workerSchedule.endOfWork.Ticks,

\_deptSchedule.endOfWork.Ticks

)));

}

private List<Reception> \_availableTemplates;

private TimeSpan \_slotLength = TimeSpan.FromMinutes(30);

private async void ViewReceptionsForm\_Load(object sender, EventArgs e)

{

// 1. Загрузка всех приёмов (шаблонов)

var allReceptions = await \_client.GetFromJsonAsync<List<Reception>>("api/reception/get-all");

// 2. Выбираем только шаблоны, относящиеся к отделению и без назначенного времени

\_availableTemplates = allReceptions

.Where(r => r.department.id == \_department.id && r.startTime == null)

.ToList();

comboBoxReceptions.Items.Clear();

foreach (var r in \_availableTemplates)

{

comboBoxReceptions.Items.Add($"{r.time} мин - {string.Join(", ", r.requiredPersonnel.Select(j => j.name))}");

}

if (comboBoxReceptions.Items.Count > 0)

comboBoxReceptions.SelectedIndex = 0;

// 3. Загрузка занятых приёмов на этот день

var scheduledReceptions = allReceptions

.Where(r => r.date == \_day && r.startTime != null && r.personnel != null && r.personnel.Any(p => p.id == \_worker.id))

.OrderBy(r => r.startTime)

.ToList();

labelInfo.Text = $"Рабочее время: {\_startWork} - {\_endWork}";

listBoxReceptions.Items.Clear();

foreach (var r in scheduledReceptions)

{

listBoxReceptions.Items.Add($"{r.startTime} - {r.endTime}: {string.Join(", ", r.requiredPersonnel.Select(j => j.name))}");

}

LoadAvailableStartTimes(scheduledReceptions);

}

private void LoadAvailableStartTimes(List<Reception> existingReceptions)

{

comboBoxSlots.Items.Clear();

var start = \_startWork;

while (start.Add(\_slotLength) <= \_endWork)

{

// Проверка пересечений

bool conflict = existingReceptions.Any(r =>

r.startTime.HasValue &&

start >= r.startTime.Value &

start <= r.endTime

);

if (!conflict)

{

comboBoxSlots.Items.Add(start.ToString("HH:mm"));

}

start = start.Add(\_slotLength);

}

if (comboBoxSlots.Items.Count > 0)

comboBoxSlots.SelectedIndex = 0;

}

private async void buttonAddReception\_Click(object sender, EventArgs e)

{

if (comboBoxSlots.SelectedItem == null || comboBoxReceptions.SelectedIndex == -1)

return;

var selectedStart = TimeOnly.Parse(comboBoxSlots.SelectedItem.ToString());

var template = \_availableTemplates[comboBoxReceptions.SelectedIndex];

// Проверка: подходит ли врач по должности

bool canAssign = template.requiredPersonnel.Any(j => j.id == \_worker.JobTitleId);

if (!canAssign)

{

MessageBox.Show("У врача неподходящая должность для выбранного приёма.");

return;

}

Reception oldReception = await \_client.GetFromJsonAsync<Reception>($"api/reception/get/{template.id}");

// Назначаем врача

if (oldReception == null)

{

MessageBox.Show("не нашёлся приём.");

return;

}

if (oldReception.personnel == null)

oldReception.personnel = new();

oldReception.personnel.Add(\_worker);

oldReception.date = \_day;

oldReception.startTime = selectedStart;

var response = await \_client.PutAsJsonAsync($"api/reception/update/{oldReception.id}", oldReception);

if (response.IsSuccessStatusCode)

{

MessageBox.Show("Приём успешно назначен.");

ViewReceptionsForm\_Load(this, EventArgs.Empty); // Обновим форму

}

else

{

MessageBox.Show("Ошибка при назначении приёма."); // вот здесь без каких либо причин оказываюсь на сервере даже метод контроллера не вызывается

}

}

//private async void buttonAddReception\_Click(object sender, EventArgs e)

//{

// if (comboBoxSlots.SelectedItem == null || comboBoxReceptions.SelectedIndex == -1)

// return;

// var selectedStart = TimeOnly.Parse(comboBoxSlots.SelectedItem.ToString());

// var template = \_availableTemplates[comboBoxReceptions.SelectedIndex];

// // Проверка должности

// bool canAssign = template.requiredPersonnel.Any(j => j.id == \_worker.JobTitleId);

// if (!canAssign)

// {

// MessageBox.Show("У врача неподходящая должность для выбранного приёма.");

// return;

// }

// // 1. Удаляем шаблон

// var deleteResp = await \_client.DeleteAsync($"api/reception/delete/{template.id}");

// if (!deleteResp.IsSuccessStatusCode)

// {

// MessageBox.Show("Не удалось удалить шаблон приёма.");

// return;

// }

// // 2. Создаём назначенный приём

// var dto = new ReceptionCreateDto

// {

// date = \_day,

// startTime = selectedStart,

// time = template.time,

// departmentId = \_department.id,

// requiredPersonnelIds = template.requiredPersonnel.Select(j => j.id).ToList(),

// personnelIds = new List<int> { \_worker.id }

// };

// var createResp = await \_client.PostAsJsonAsync("api/reception/add-dto", dto);

// if (createResp.IsSuccessStatusCode) // newReception в этот момент null какого то хрена

// {

// MessageBox.Show("Приём успешно назначен.");

// ViewReceptionsForm\_Load(this, EventArgs.Empty); // Обновить список

// }

// else

// {

// MessageBox.Show("Ошибка при создании приёма.");

// }

//}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\ViewReceptionsForm.Designer.cs ====

﻿namespace DesctopSheduleManager.Forms

{

partial class ViewReceptionsForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

labelInfo = new Label();

listBoxReceptions = new ListBox();

comboBoxSlots = new ComboBox();

buttonAddReception = new Button();

comboBoxReceptions = new ComboBox();

SuspendLayout();

//

// labelInfo

//

labelInfo.AutoSize = true;

labelInfo.Location = new Point(12, 9);

labelInfo.Name = "labelInfo";

labelInfo.Size = new Size(38, 15);

labelInfo.TabIndex = 0;

labelInfo.Text = "label1";

//

// listBoxReceptions

//

listBoxReceptions.FormattingEnabled = true;

listBoxReceptions.ItemHeight = 15;

listBoxReceptions.Location = new Point(12, 27);

listBoxReceptions.Name = "listBoxReceptions";

listBoxReceptions.Size = new Size(313, 169);

listBoxReceptions.TabIndex = 1;

//

// comboBoxSlots

//

comboBoxSlots.FormattingEnabled = true;

comboBoxSlots.Location = new Point(12, 202);

comboBoxSlots.Name = "comboBoxSlots";

comboBoxSlots.Size = new Size(313, 23);

comboBoxSlots.TabIndex = 2;

//

// buttonAddReception

//

buttonAddReception.Location = new Point(192, 260);

buttonAddReception.Name = "buttonAddReception";

buttonAddReception.Size = new Size(133, 23);

buttonAddReception.TabIndex = 3;

buttonAddReception.Text = "добавить приём";

buttonAddReception.UseVisualStyleBackColor = true;

buttonAddReception.Click += buttonAddReception\_Click;

//

// comboBoxReceptions

//

comboBoxReceptions.FormattingEnabled = true;

comboBoxReceptions.Location = new Point(12, 231);

comboBoxReceptions.Name = "comboBoxReceptions";

comboBoxReceptions.Size = new Size(313, 23);

comboBoxReceptions.TabIndex = 4;

//

// ViewReceptionsForm

//

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(340, 298);

Controls.Add(comboBoxReceptions);

Controls.Add(buttonAddReception);

Controls.Add(comboBoxSlots);

Controls.Add(listBoxReceptions);

Controls.Add(labelInfo);

Name = "ViewReceptionsForm";

Text = "ViewReceptionsForm";

Load += ViewReceptionsForm\_Load;

ResumeLayout(false);

PerformLayout();

}

#endregion

private Label labelInfo;

private ListBox listBoxReceptions;

private ComboBox comboBoxSlots;

private Button buttonAddReception;

private ComboBox comboBoxReceptions;

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\WorkerScheduleEditorForm.cs ====

﻿using Models;

namespace DesctopSheduleManager.Forms

{

public partial class WorkerScheduleEditorForm : Form

{

private List<CheckBox> workingCheckboxes = new();

private List<DateTimePicker> startPickers = new();

private List<DateTimePicker> endPickers = new();

private List<Label> dayLabels = new();

private List<string> dayNames = new() {

"Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Вс",

};// должны циклически повторяться при добавлении нового дня

private List<WorkerWorkSchedule4Day> schedule;

public List<WorkerWorkSchedule4Day> UpdatedSchedule => schedule;

public WorkerScheduleEditorForm(List<WorkerWorkSchedule4Day> existingSchedule)

{

InitializeComponent();

schedule = existingSchedule.Select(ws => new WorkerWorkSchedule4Day

{

startOfWork = ws.startOfWork,

endOfWork = ws.endOfWork,

isWorking = ws.isWorking

}).ToList();

LoadScheduleIntoControls();

}

private void LoadScheduleIntoControls()

{

tableLayoutPanel1.RowCount = schedule.Count;

tableLayoutPanel1.ColumnCount = 4;

tableLayoutPanel1.Controls.Clear();

tableLayoutPanel1.AutoSize = true;

for (int i = 0; i < schedule.Count; i++)

{

var label = new Label

{

Text = dayNames[i % dayNames.Count],

AutoSize = true,

Anchor = AnchorStyles.Left

};

var checkbox = new CheckBox { Checked = schedule[i]?.isWorking ?? false };

var start = new DateTimePicker { Format = DateTimePickerFormat.Time, ShowUpDown = true };

var end = new DateTimePicker { Format = DateTimePickerFormat.Time, ShowUpDown = true };

if (schedule[i] != null)

{

start.Value = DateTime.Today.Add(schedule[i]!.startOfWork.ToTimeSpan());

end.Value = DateTime.Today.Add(schedule[i]!.endOfWork.ToTimeSpan());

}

// Добавляем в список для дальнейшего считывания

dayLabels.Add(label);

workingCheckboxes.Add(checkbox);

startPickers.Add(start);

endPickers.Add(end);

// Добавляем на панель

tableLayoutPanel1.Controls.Add(label, 0, i);

tableLayoutPanel1.Controls.Add(checkbox, 1, i);

tableLayoutPanel1.Controls.Add(start, 2, i);

tableLayoutPanel1.Controls.Add(end, 3, i);

}

}

private void buttonSave\_Click(object sender, EventArgs e)

{

for (int i = 0; i < schedule.Count; i++)

{

if (workingCheckboxes[i].Checked)

{

schedule[i] = new WorkerWorkSchedule4Day

{

isWorking = true,

startOfWork = TimeOnly.FromDateTime(startPickers[i].Value),

endOfWork = TimeOnly.FromDateTime(endPickers[i].Value)

};

}

else

{

schedule[i] = new WorkerWorkSchedule4Day

{

isWorking = false,

startOfWork = TimeOnly.MinValue,

endOfWork = TimeOnly.MinValue

};

}

}

DialogResult = DialogResult.OK;

Close();

}

private void buttonAdd\_Click(object sender, EventArgs e)

{

// Добавим новый день по умолчанию

schedule.Add(new WorkerWorkSchedule4Day

{

isWorking = false,

startOfWork = TimeOnly.MinValue,

endOfWork = TimeOnly.MinValue

});

// Перезагрузим UI

LoadScheduleIntoControls();

}

private void buttonDel\_Click(object sender, EventArgs e)

{

if (schedule.Count > 0)

{

// Удалим последний день

schedule.RemoveAt(schedule.Count - 1);

// Перезагрузим UI

LoadScheduleIntoControls();

}

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\WorkerScheduleEditorForm.Designer.cs ====

﻿namespace DesctopSheduleManager.Forms

{

partial class WorkerScheduleEditorForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

tableLayoutPanel2 = new TableLayoutPanel();

tableLayoutPanel1 = new TableLayoutPanel();

tableLayoutPanel3 = new TableLayoutPanel();

buttonSave = new Button();

buttonAdd = new Button();

buttonDel = new Button();

tableLayoutPanel2.SuspendLayout();

tableLayoutPanel3.SuspendLayout();

SuspendLayout();

//

// tableLayoutPanel2

//

tableLayoutPanel2.AutoSize = true;

tableLayoutPanel2.ColumnCount = 2;

tableLayoutPanel2.ColumnStyles.Add(new ColumnStyle());

tableLayoutPanel2.ColumnStyles.Add(new ColumnStyle());

tableLayoutPanel2.Controls.Add(tableLayoutPanel1, 0, 0);

tableLayoutPanel2.Controls.Add(tableLayoutPanel3, 1, 0);

tableLayoutPanel2.Location = new Point(12, 12);

tableLayoutPanel2.Name = "tableLayoutPanel2";

tableLayoutPanel2.RowCount = 1;

tableLayoutPanel2.RowStyles.Add(new RowStyle());

tableLayoutPanel2.Size = new Size(432, 116);

tableLayoutPanel2.TabIndex = 0;

//

// tableLayoutPanel1

//

tableLayoutPanel1.ColumnCount = 2;

tableLayoutPanel1.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 50F));

tableLayoutPanel1.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 50F));

tableLayoutPanel1.Location = new Point(3, 3);

tableLayoutPanel1.Name = "tableLayoutPanel1";

tableLayoutPanel1.RowCount = 2;

tableLayoutPanel1.RowStyles.Add(new RowStyle(SizeType.Percent, 50F));

tableLayoutPanel1.RowStyles.Add(new RowStyle(SizeType.Percent, 50F));

tableLayoutPanel1.Size = new Size(200, 100);

tableLayoutPanel1.TabIndex = 0;

//

// tableLayoutPanel3

//

tableLayoutPanel3.AutoSize = true;

tableLayoutPanel3.ColumnCount = 2;

tableLayoutPanel3.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 50F));

tableLayoutPanel3.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 50F));

tableLayoutPanel3.Controls.Add(buttonSave, 0, 1);

tableLayoutPanel3.Controls.Add(buttonAdd, 0, 0);

tableLayoutPanel3.Controls.Add(buttonDel, 1, 0);

tableLayoutPanel3.Location = new Point(209, 3);

tableLayoutPanel3.Name = "tableLayoutPanel3";

tableLayoutPanel3.RowCount = 2;

tableLayoutPanel3.RowStyles.Add(new RowStyle(SizeType.Percent, 50F));

tableLayoutPanel3.RowStyles.Add(new RowStyle(SizeType.Percent, 50F));

tableLayoutPanel3.Size = new Size(220, 110);

tableLayoutPanel3.TabIndex = 1;

//

// buttonSave

//

buttonSave.Anchor = AnchorStyles.Bottom | AnchorStyles.Left;

buttonSave.Location = new Point(3, 84);

buttonSave.Name = "buttonSave";

buttonSave.Size = new Size(75, 23);

buttonSave.TabIndex = 1;

buttonSave.Text = "сохранить";

buttonSave.UseVisualStyleBackColor = true;

buttonSave.Click += buttonSave\_Click;

//

// buttonAdd

//

buttonAdd.Location = new Point(3, 3);

buttonAdd.Name = "buttonAdd";

buttonAdd.Size = new Size(100, 49);

buttonAdd.TabIndex = 2;

buttonAdd.Text = "добавить день в расписание";

buttonAdd.UseVisualStyleBackColor = true;

buttonAdd.Click += buttonAdd\_Click;

//

// buttonDel

//

buttonDel.Anchor = AnchorStyles.Top | AnchorStyles.Right;

buttonDel.Location = new Point(113, 3);

buttonDel.Name = "buttonDel";

buttonDel.Size = new Size(104, 49);

buttonDel.TabIndex = 3;

buttonDel.Text = "удалить день из расписания";

buttonDel.UseVisualStyleBackColor = true;

buttonDel.Click += buttonDel\_Click;

//

// WorkerScheduleEditorForm

//

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(927, 450);

Controls.Add(tableLayoutPanel2);

Name = "WorkerScheduleEditorForm";

Text = "WorkerScheduleEditorForm";

tableLayoutPanel2.ResumeLayout(false);

tableLayoutPanel2.PerformLayout();

tableLayoutPanel3.ResumeLayout(false);

ResumeLayout(false);

PerformLayout();

}

#endregion

private TableLayoutPanel tableLayoutPanel2;

private TableLayoutPanel tableLayoutPanel1;

private Button buttonSave;

private TableLayoutPanel tableLayoutPanel3;

private Button buttonAdd;

private Button buttonDel;

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\WorkersForm.cs ====

﻿using DesctopSheduleManager.Forms;

using DesctopSheduleManager.Utilities;

using Models;

using System.Net.Http.Json;

namespace DesctopSheduleManager

{

public partial class WorkersForm : Form

{

private readonly HttpClient \_client;

private List<JobTitle> \_jobTitles = new List<JobTitle>(); // Список должностей

public WorkersForm()

{

InitializeComponent();

\_client = ApiClient.Instance;

txtPassword.UseSystemPasswordChar = true;

}

// Загружаем список должностей

private async Task LoadJobTitlesAsync()

{

try

{

var jobTitles = await \_client.GetFromJsonAsync<List<JobTitle>>("api/job-title/get-all");

if (jobTitles != null)

{

\_jobTitles = jobTitles;

cmbJobTitle.DataSource = \_jobTitles;

cmbJobTitle.DisplayMember = "name";

cmbJobTitle.ValueMember = "id";

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки должностей:\n{ex.Message}");

}

}

// Вызываем загрузку данных при загрузке формы

private async void WorkersForm\_Load(object sender, EventArgs e)

{

await LoadJobTitlesAsync(); // Загружаем должности

await LoadWorkersAsync(); // Загружаем сотрудников

}

private async Task LoadWorkersAsync()

{

try

{

var workers = await \_client.GetFromJsonAsync<List<Worker>>("api/worker/get-all");

if (workers != null)

{

dataGridWorkers.DataSource = workers.Select(w => new

{

w.id,

w.name,

w.login,

JobTitle = w.jobTitle?.name

}).ToList();

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки сотрудников:\n{ex.Message}");

}

}

// Обработчик добавления нового сотрудника

private async void btnAdd\_Click(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(txtName.Text) ||

string.IsNullOrWhiteSpace(txtLogin.Text) ||

string.IsNullOrWhiteSpace(txtPassword.Text) ||

cmbJobTitle.SelectedItem == null)

{

MessageBox.Show("Пожалуйста, заполните все поля и выберите должность.");

return;

}

var selectedJobTitle = (JobTitle)cmbJobTitle.SelectedItem;

var newWorker = new Worker

{

name = txtName.Text.Trim(),

login = txtLogin.Text.Trim(),

password = txtPassword.Text,

jobTitle = selectedJobTitle,

workSchedules = new()

};

try

{

var response = await \_client.PostAsJsonAsync("api/worker/add", newWorker);

if (response.IsSuccessStatusCode)

{

await LoadWorkersAsync();

MessageBox.Show("Сотрудник успешно добавлен.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при добавлении:\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка:\n{ex.Message}");

}

}

// Обработчик обновления сотрудника

private async void btnUpdate\_Click(object sender, EventArgs e)

{

if (dataGridWorkers.CurrentRow == null)

{

MessageBox.Show("Выберите сотрудника для обновления.");

return;

}

if (string.IsNullOrWhiteSpace(txtName.Text) ||

string.IsNullOrWhiteSpace(txtLogin.Text) ||

string.IsNullOrWhiteSpace(txtPassword.Text) ||

cmbJobTitle.SelectedItem == null)

{

MessageBox.Show("Пожалуйста, заполните все поля и выберите должность.");

return;

}

var id = (int)dataGridWorkers.CurrentRow.Cells["id"].Value;

var selectedJobTitle = (JobTitle)cmbJobTitle.SelectedItem;

var updatedWorker = new Worker

{

id = id,

name = txtName.Text.Trim(),

login = txtLogin.Text.Trim(),

password = txtPassword.Text,

jobTitle = selectedJobTitle

};

try

{

var response = await \_client.PutAsJsonAsync($"api/worker/update/{id}", updatedWorker);

if (response.IsSuccessStatusCode)

{

await LoadWorkersAsync();

MessageBox.Show("Сотрудник успешно обновлён.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при обновлении:\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка:\n{ex.Message}");

}

}

// Обработчик удаления сотрудника

private async void btnDelete\_Click(object sender, EventArgs e)

{

if (dataGridWorkers.CurrentRow == null)

{

MessageBox.Show("Выберите сотрудника для удаления.");

return;

}

var id = (int)dataGridWorkers.CurrentRow.Cells["id"].Value;

if (MessageBox.Show("Удалить выбранного сотрудника?", "Подтверждение", MessageBoxButtons.YesNo) == DialogResult.Yes)

{

try

{

var response = await \_client.DeleteAsync($"api/worker/delete/{id}");

if (response.IsSuccessStatusCode)

{

await LoadWorkersAsync();

MessageBox.Show("Сотрудник удалён.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"Ошибка при удалении:\n{error}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка:\n{ex.Message}");

}

}

}

// Обработчик выбора строки в DataGridView

private void dataGridWorkers\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridWorkers.CurrentRow == null) return;

var selectedRow = dataGridWorkers.CurrentRow;

txtName.Text = selectedRow.Cells["name"].Value.ToString();

txtLogin.Text = selectedRow.Cells["login"].Value.ToString();

// Присваиваем ComboBox выбранную должность

var jobTitle = \_jobTitles.FirstOrDefault(j => j.name == selectedRow.Cells["JobTitle"].Value.ToString());

cmbJobTitle.SelectedItem = jobTitle;

}

private async void btnRefresh\_Click(object sender, EventArgs e)

{

await LoadWorkersAsync();

}

private async void buttonNotificationTest\_Click(object sender, EventArgs e)

{

if (dataGridWorkers.SelectedRows.Count == 0)

{

MessageBox.Show("Выберите работника.");

return;

}

// Предполагаем, что id хранится в первой колонке

int selectedWorkerId = Convert.ToInt32(dataGridWorkers.SelectedRows[0].Cells["id"].Value);

try

{

var response = await \_client.GetAsync(

$"api/worker/test-notification/{selectedWorkerId}");

if (response.IsSuccessStatusCode)

{

MessageBox.Show("✅ Уведомление успешно отправлено.");

}

else

{

var errorContent = await response.Content.ReadAsStringAsync();

MessageBox.Show($"❌ Ошибка при отправке уведомления:\n{errorContent}");

}

}

catch (Exception ex)

{

MessageBox.Show($"⚠️ Ошибка запроса:\n{ex.Message}");

}

}

private async void buttonChangeShedule\_Click(object sender, EventArgs e)

{

if (dataGridWorkers.CurrentRow == null)

{

MessageBox.Show("Выберите сотрудника.");

return;

}

int id = (int)dataGridWorkers.CurrentRow.Cells["id"].Value;

var worker = await \_client.GetFromJsonAsync<Worker>($"api/worker/get/{id}");

if (worker == null)

{

MessageBox.Show("Работник не найден.");

return;

}

var scheduleEditor = new WorkerScheduleEditorForm(worker.workSchedules); // создайте такую форму

if (scheduleEditor.ShowDialog() == DialogResult.OK)

{

worker.workSchedules = scheduleEditor.UpdatedSchedule;

var response = await \_client.PutAsJsonAsync($"api/worker/update/{id}", worker);

if (response.IsSuccessStatusCode)

{

MessageBox.Show("✅ Расписание обновлено.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"❌ Ошибка при обновлении:\n{error}");

}

}

}

private async void buttonChangeVacations\_Click(object sender, EventArgs e)

{

if (dataGridWorkers.CurrentRow == null)

{

MessageBox.Show("Выберите сотрудника.");

return;

}

int id = (int)dataGridWorkers.CurrentRow.Cells["id"].Value;

var worker = await \_client.GetFromJsonAsync<Worker>($"api/worker/get/{id}");

if (worker == null)

{

MessageBox.Show("Работник не найден.");

return;

}

var vacationEditor = new VacationEditorForm(worker.vacations); // создайте такую форму

if (vacationEditor.ShowDialog() == DialogResult.OK)

{

worker.vacations = vacationEditor.UpdatedVacations;

var response = await \_client.PutAsJsonAsync($"api/worker/update/{id}", worker);

if (response.IsSuccessStatusCode)

{

MessageBox.Show("✅ Отпуска обновлены.");

}

else

{

var error = await response.Content.ReadAsStringAsync();

MessageBox.Show($"❌ Ошибка при обновлении:\n{error}");

}

}

}

private void checkBoxShowPass\_CheckedChanged(object sender, EventArgs e)

{

txtPassword.UseSystemPasswordChar = !checkBoxShowPass.Checked;

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Forms\WorkersForm.Designer.cs ====

﻿namespace DesctopSheduleManager

{

partial class WorkersForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

private TextBox txtName;

private TextBox txtLogin;

private TextBox txtPassword;

private ComboBox cmbJobTitle;

private DataGridView dataGridWorkers;

private Button btnAdd;

private Button btnUpdate;

private Button btnDelete;

private Button btnRefresh;

private Label lblName;

private Label lblLogin;

private Label lblPassword;

private Label lblJobTitle;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

txtName = new TextBox();

txtLogin = new TextBox();

txtPassword = new TextBox();

cmbJobTitle = new ComboBox();

dataGridWorkers = new DataGridView();

btnAdd = new Button();

btnUpdate = new Button();

btnDelete = new Button();

btnRefresh = new Button();

lblName = new Label();

lblLogin = new Label();

lblPassword = new Label();

lblJobTitle = new Label();

buttonNotificationTest = new Button();

buttonChangeShedule = new Button();

buttonChangeVacations = new Button();

checkBoxShowPass = new CheckBox();

((System.ComponentModel.ISupportInitialize)dataGridWorkers).BeginInit();

SuspendLayout();

//

// txtName

//

txtName.Location = new Point(120, 20);

txtName.Name = "txtName";

txtName.Size = new Size(200, 23);

txtName.TabIndex = 0;

//

// txtLogin

//

txtLogin.Location = new Point(120, 60);

txtLogin.Name = "txtLogin";

txtLogin.Size = new Size(200, 23);

txtLogin.TabIndex = 1;

//

// txtPassword

//

txtPassword.Location = new Point(120, 100);

txtPassword.Name = "txtPassword";

txtPassword.Size = new Size(179, 23);

txtPassword.TabIndex = 2;

//

// cmbJobTitle

//

cmbJobTitle.DropDownStyle = ComboBoxStyle.DropDownList;

cmbJobTitle.Location = new Point(120, 140);

cmbJobTitle.Name = "cmbJobTitle";

cmbJobTitle.Size = new Size(200, 23);

cmbJobTitle.TabIndex = 3;

//

// dataGridWorkers

//

dataGridWorkers.Location = new Point(20, 200);

dataGridWorkers.Name = "dataGridWorkers";

dataGridWorkers.Size = new Size(500, 200);

dataGridWorkers.TabIndex = 4;

dataGridWorkers.SelectionChanged += dataGridWorkers\_SelectionChanged;

//

// btnAdd

//

btnAdd.Location = new Point(340, 20);

btnAdd.Name = "btnAdd";

btnAdd.Size = new Size(100, 30);

btnAdd.TabIndex = 5;

btnAdd.Text = "Добавить";

btnAdd.Click += btnAdd\_Click;

//

// btnUpdate

//

btnUpdate.Location = new Point(340, 60);

btnUpdate.Name = "btnUpdate";

btnUpdate.Size = new Size(100, 30);

btnUpdate.TabIndex = 6;

btnUpdate.Text = "Изменить";

btnUpdate.Click += btnUpdate\_Click;

//

// btnDelete

//

btnDelete.Location = new Point(340, 100);

btnDelete.Name = "btnDelete";

btnDelete.Size = new Size(100, 30);

btnDelete.TabIndex = 7;

btnDelete.Text = "Удалить";

btnDelete.Click += btnDelete\_Click;

//

// btnRefresh

//

btnRefresh.Location = new Point(340, 140);

btnRefresh.Name = "btnRefresh";

btnRefresh.Size = new Size(100, 30);

btnRefresh.TabIndex = 8;

btnRefresh.Text = "Обновить список";

btnRefresh.Click += btnRefresh\_Click;

//

// lblName

//

lblName.Location = new Point(20, 20);

lblName.Name = "lblName";

lblName.Size = new Size(100, 20);

lblName.TabIndex = 9;

lblName.Text = "Имя:";

//

// lblLogin

//

lblLogin.Location = new Point(20, 60);

lblLogin.Name = "lblLogin";

lblLogin.Size = new Size(100, 20);

lblLogin.TabIndex = 10;

lblLogin.Text = "Логин:";

//

// lblPassword

//

lblPassword.Location = new Point(20, 100);

lblPassword.Name = "lblPassword";

lblPassword.Size = new Size(100, 20);

lblPassword.TabIndex = 11;

lblPassword.Text = "Пароль:";

//

// lblJobTitle

//

lblJobTitle.Location = new Point(20, 140);

lblJobTitle.Name = "lblJobTitle";

lblJobTitle.Size = new Size(100, 20);

lblJobTitle.TabIndex = 12;

lblJobTitle.Text = "Должность:";

//

// buttonNotificationTest

//

buttonNotificationTest.Location = new Point(405, 415);

buttonNotificationTest.Name = "buttonNotificationTest";

buttonNotificationTest.Size = new Size(115, 23);

buttonNotificationTest.TabIndex = 13;

buttonNotificationTest.Text = "тест уведомления";

buttonNotificationTest.UseVisualStyleBackColor = true;

buttonNotificationTest.Click += buttonNotificationTest\_Click;

//

// buttonChangeShedule

//

buttonChangeShedule.Location = new Point(20, 415);

buttonChangeShedule.Name = "buttonChangeShedule";

buttonChangeShedule.Size = new Size(151, 23);

buttonChangeShedule.TabIndex = 14;

buttonChangeShedule.Text = "изменить расписание";

buttonChangeShedule.UseVisualStyleBackColor = true;

buttonChangeShedule.Click += buttonChangeShedule\_Click;

//

// buttonChangeVacations

//

buttonChangeVacations.Location = new Point(177, 415);

buttonChangeVacations.Name = "buttonChangeVacations";

buttonChangeVacations.Size = new Size(143, 23);

buttonChangeVacations.TabIndex = 15;

buttonChangeVacations.Text = "изменить отпуска";

buttonChangeVacations.UseVisualStyleBackColor = true;

buttonChangeVacations.Click += buttonChangeVacations\_Click;

//

// checkBoxShowPass

//

checkBoxShowPass.AutoSize = true;

checkBoxShowPass.Location = new Point(305, 104);

checkBoxShowPass.Name = "checkBoxShowPass";

checkBoxShowPass.Size = new Size(15, 14);

checkBoxShowPass.TabIndex = 16;

checkBoxShowPass.UseVisualStyleBackColor = true;

checkBoxShowPass.CheckedChanged += checkBoxShowPass\_CheckedChanged;

//

// WorkersForm

//

ClientSize = new Size(550, 450);

Controls.Add(checkBoxShowPass);

Controls.Add(buttonChangeVacations);

Controls.Add(buttonChangeShedule);

Controls.Add(buttonNotificationTest);

Controls.Add(txtName);

Controls.Add(txtLogin);

Controls.Add(txtPassword);

Controls.Add(cmbJobTitle);

Controls.Add(dataGridWorkers);

Controls.Add(btnAdd);

Controls.Add(btnUpdate);

Controls.Add(btnDelete);

Controls.Add(btnRefresh);

Controls.Add(lblName);

Controls.Add(lblLogin);

Controls.Add(lblPassword);

Controls.Add(lblJobTitle);

Name = "WorkersForm";

Text = "Управление сотрудниками";

Load += WorkersForm\_Load;

((System.ComponentModel.ISupportInitialize)dataGridWorkers).EndInit();

ResumeLayout(false);

PerformLayout();

}

#endregion

private Button buttonNotificationTest;

private Button buttonChangeShedule;

private Button buttonChangeVacations;

private CheckBox checkBoxShowPass;

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\obj\Debug\net8.0-windows\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\Schedule\DesctopSheduleManager\obj\Debug\net8.0-windows\DesctopSheduleManager.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("DesctopSheduleManager")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+504c82c7efe4e511801a3b9066c8d233e0a42f0d")]

[assembly: System.Reflection.AssemblyProductAttribute("DesctopSheduleManager")]

[assembly: System.Reflection.AssemblyTitleAttribute("DesctopSheduleManager")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

[assembly: System.Runtime.Versioning.TargetPlatformAttribute("Windows7.0")]

[assembly: System.Runtime.Versioning.SupportedOSPlatformAttribute("Windows7.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\Schedule\DesctopSheduleManager\obj\Debug\net8.0-windows\DesctopSheduleManager.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::System;

global using global::System.Collections.Generic;

global using global::System.Drawing;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

global using global::System.Windows.Forms;

// ==== D:\piaps\Schedule\DesctopSheduleManager\obj\Release\net8.0-windows\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\Schedule\DesctopSheduleManager\obj\Release\net8.0-windows\DesctopSheduleManager.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("DesctopSheduleManager")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0")]

[assembly: System.Reflection.AssemblyProductAttribute("DesctopSheduleManager")]

[assembly: System.Reflection.AssemblyTitleAttribute("DesctopSheduleManager")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

[assembly: System.Runtime.Versioning.TargetPlatformAttribute("Windows7.0")]

[assembly: System.Runtime.Versioning.SupportedOSPlatformAttribute("Windows7.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\Schedule\DesctopSheduleManager\obj\Release\net8.0-windows\DesctopSheduleManager.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::System;

global using global::System.Collections.Generic;

global using global::System.Drawing;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

global using global::System.Windows.Forms;

// ==== D:\piaps\Schedule\DesctopSheduleManager\Utilities\ApiClient.cs ====

﻿namespace DesctopSheduleManager.Utilities

{

public static class ApiClient

{

private static readonly HttpClient \_client;

static ApiClient()

{

\_client = new HttpClient

{

BaseAddress = new Uri(ConfigManager.Config.ApiBaseUrl)

};

}

public static HttpClient Instance

{

get

{

// Обновляем заголовок sid перед каждым использованием

if (\_client.DefaultRequestHeaders.Contains("sid"))

\_client.DefaultRequestHeaders.Remove("sid");

if (!string.IsNullOrEmpty(SessionManager.SID))

\_client.DefaultRequestHeaders.Add("sid", SessionManager.SID);

return \_client;

}

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Utilities\ApiClientFactory.cs ====

﻿namespace DesctopSheduleManager.Utilities

{

public static class ApiClientFactory

{

public static HttpClient Create()

{

var client = new HttpClient

{

BaseAddress = new Uri(ConfigManager.Config.ApiBaseUrl)

};

if (!string.IsNullOrEmpty(SessionManager.SID))

{

client.DefaultRequestHeaders.Add("sid", SessionManager.SID);

}

return client;

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Utilities\AppConfig.cs ====

﻿namespace DesctopSheduleManager.Utilities

{

public class AppConfig

{

public string ApiBaseUrl { get; set; }

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Utilities\ConfigManager.cs ====

﻿using System.Text.Json;

namespace DesctopSheduleManager.Utilities

{

public static class ConfigManager

{

private static AppConfig \_config;

public static AppConfig Config

{

get

{

if (\_config == null)

LoadConfig();

return \_config;

}

}

private static void LoadConfig()

{

var configText = File.ReadAllText("config.json");

\_config = JsonSerializer.Deserialize<AppConfig>(configText);

}

}

}

// ==== D:\piaps\Schedule\DesctopSheduleManager\Utilities\SessionManager.cs ====

﻿namespace DesctopSheduleManager.Utilities

{

public static class SessionManager

{

public static string SID { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\DayOff.cs ====

﻿

using Models.Interfaces;

using System.Text.Json.Serialization;

namespace Models

{

public class DayOff : IDayOff

{

public DateOnly date { get; set; }

public int id { get; set; }

public int VacationId { get; set; }

[JsonIgnore] public Vacation? Vacation { get; set;}

}

}

// ==== D:\piaps\Schedule\Models\Department.cs ====

﻿using Models.Interfaces;

namespace Models

{

public class Department : IModel

{

public int id { get; set; }

public string? name { get; set; }

public List<WorkSchedule4Day?> workSchedules { get; set; } = new List<WorkSchedule4Day?>(7);

public List<Worker>? workers { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\Holiday.cs ====

﻿using Models.Interfaces;

namespace Models

{

public class Holiday : IDayOff

{

public int id { get; set; }

public DateOnly date { get; set; }

public string name { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\JobTitle.cs ====

﻿using Models.Interfaces;

using System.Text.Json.Serialization;

namespace Models

{

public class JobTitle : IModel

{

public int id { get; set; }

public string? name { get; set; }

[JsonIgnore] public List<Reception>? receptions { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\Reception.cs ====

﻿using Models.Interfaces;

namespace Models

{

public class Reception : IModel

{

public int id { get; set; }

/// <summary>

/// время на приём

/// </summary>

public TimeSpan time { get; set; }

/// <summary>

/// персонал необходимый для проведения

/// </summary>

public List<JobTitle> requiredPersonnel { get; set; }

/// <summary>

/// отделение приёма

/// </summary>

public Department department { get; set; }

//----------------------- после составления расписания

/// <summary>

/// день приёма

/// </summary>

public DateOnly? date { get; set; }

/// <summary>

/// время начала приёма

/// </summary>

public TimeOnly? startTime { get; set; }

/// <summary>

/// время конца приёма

/// </summary>

public TimeOnly? endTime => startTime.HasValue ? startTime.Value.Add(time) : null;

/// <summary>

/// работники приписанные к приёму

/// </summary>

public List<Worker>? personnel { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\Vacation.cs ====

﻿using Models.Interfaces;

using System.Text.Json.Serialization;

namespace Models

{

public class Vacation : IModel

{

public int id { get; set; }

public List<DayOff> days { get; set; }

public int WorkerId { get; set; }

[JsonIgnore] public Worker? Worker { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\Worker.cs ====

﻿using Models.Interfaces;

using System.Text.Json.Serialization;

namespace Models

{

public class Worker : IModel

{

public int id { get; set; }

public string? name { get; set; }

public string? login { get; set; }

public string? password { get; set; }

public int? JobTitleId { get; set; }

public JobTitle? jobTitle { get; set; }

public List<Vacation>? vacations { get; set; }

public List<WorkerWorkSchedule4Day> workSchedules { get; set; }

[JsonIgnore] public List<Department>? departments { get; set; }

[JsonIgnore] public List<Reception>? receptions { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\WorkerWorkSchedule4Day.cs ====

﻿using Models.Interfaces;

using System.Text.Json.Serialization;

namespace Models

{

public class WorkerWorkSchedule4Day : IModel

{

public int id { get; set; }

public TimeOnly startOfWork { get; set; }

public TimeOnly endOfWork { get; set; }

public bool isWorking { get; set; }

public int WorkerId { get; set;}

[JsonIgnore] public Worker? Worker { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\WorkSchedule4Day.cs ====

﻿using Models.Interfaces;

using System.Text.Json.Serialization;

namespace Models

{

public class WorkSchedule4Day : IModel

{

public int id { get; set; }

public TimeOnly startOfWork { get; set; }

public TimeOnly endOfWork { get; set; }

public bool isWorking { get; set; }

public int DepartmentId { get; set; } // 🔥 Добавлено

[JsonIgnore] public Department? Department { get; set; } // 🔥 Добавлено

}

}

// ==== D:\piaps\Schedule\Models\DTO\ReceptionCreateDto.cs ====

﻿namespace Models.DTO

{

public class ReceptionCreateDto

{

public DateOnly date { get; set; }

public TimeOnly startTime { get; set; }

public TimeSpan time { get; set; }

public int departmentId { get; set; }

public List<int> requiredPersonnelIds { get; set; }

public List<int> personnelIds { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\Interfaces\IDayOff.cs ====

﻿namespace Models.Interfaces

{

public interface IDayOff : IModel

{

public DateOnly date { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\Interfaces\IModel.cs ====

﻿namespace Models.Interfaces

{

public interface IModel

{

public int id { get; set; }

}

}

// ==== D:\piaps\Schedule\Models\obj\Debug\net8.0\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\Schedule\Models\obj\Debug\net8.0\Models.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("Models")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+504c82c7efe4e511801a3b9066c8d233e0a42f0d")]

[assembly: System.Reflection.AssemblyProductAttribute("Models")]

[assembly: System.Reflection.AssemblyTitleAttribute("Models")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\Schedule\Models\obj\Debug\net8.0\Models.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

// ==== D:\piaps\Schedule\Models\obj\Release\net8.0\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\Schedule\Models\obj\Release\net8.0\Models.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("Models")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+5d726bf7f71429a9ab662739f3c7c68769eb4d34")]

[assembly: System.Reflection.AssemblyProductAttribute("Models")]

[assembly: System.Reflection.AssemblyTitleAttribute("Models")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\Schedule\Models\obj\Release\net8.0\Models.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

// ==== D:\piaps\Schedule\Repositories\AppDbContext.cs ====

﻿using Microsoft.EntityFrameworkCore;

using Models;

namespace Repositories

{

public class AppDbContext : DbContext

{

public DbSet<Worker> Workers { get; set; }

public DbSet<Department> Departments { get; set; }

public DbSet<JobTitle> JobTitles { get; set; }

public DbSet<Reception> Receptions { get; set; }

public DbSet<Holiday> Holidays { get; set; }

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

// Настройка каскадного удаления JobTitle → Worker

modelBuilder.Entity<Worker>()

.HasOne(w => w.jobTitle)

.WithMany() // или .WithMany(j => j.Workers), если в JobTitle есть список

.HasForeignKey("JobTitleId") // явный внешний ключ

.OnDelete(DeleteBehavior.Cascade);

}

}

}

// ==== D:\piaps\Schedule\Repositories\Implementations\BaseRepository.cs ====

﻿using Microsoft.EntityFrameworkCore;

using Repositories.Interfaces;

namespace Repositories.Implementations

{

public abstract class BaseRepository<T> : IRepository<T> where T : class

{

protected readonly AppDbContext \_context;

protected readonly DbSet<T> \_dbSet;

protected BaseRepository(AppDbContext context)

{

\_context = context;

\_dbSet = context.Set<T>();

}

public abstract Task<IEnumerable<T>> GetAllAsync();

public abstract Task<T> GetByIdAsync(int id);

public async Task AddAsync(T entity) => await \_dbSet.AddAsync(entity);

public Task UpdateAsync(T entity)

{

\_dbSet.Update(entity);

return Task.CompletedTask;

}

public Task DeleteAsync(T entity)

{

\_dbSet.Remove(entity);

return Task.CompletedTask;

}

public async Task SaveAsync() => await \_context.SaveChangesAsync();

}

}

// ==== D:\piaps\Schedule\Repositories\Implementations\DepartmentRepository.cs ====

﻿using Microsoft.EntityFrameworkCore;

using Models;

using Repositories.Implementations;

using Repositories;

public class DepartmentRepository : BaseRepository<Department>

{

public DepartmentRepository(AppDbContext context) : base(context) { }

public override async Task<IEnumerable<Department>> GetAllAsync()

{

return await \_dbSet

.Include(d => d.workSchedules)

.Include(d => d.workers)

.ThenInclude(w => w.jobTitle)

.Include(d => d.workers)

.ThenInclude(w => w.vacations)

.ThenInclude(v => v.days)

.Include(d => d.workers)

.ThenInclude(w => w.workSchedules)

.ToListAsync();

}

public override async Task<Department?> GetByIdAsync(int id)

{

return await \_dbSet

.Include(d => d.workSchedules)

.Include(d => d.workers)

.ThenInclude(w => w.jobTitle)

.Include(d => d.workers)

.ThenInclude(w => w.vacations)

.ThenInclude(v => v.days)

.Include(d => d.workers)

.ThenInclude(w => w.workSchedules)

.FirstOrDefaultAsync(d => d.id == id);

}

}

// ==== D:\piaps\Schedule\Repositories\Implementations\HolidayRepository.cs ====

﻿using Microsoft.EntityFrameworkCore;

using Models;

using Repositories.Implementations;

using Repositories;

public class HolidayRepository : BaseRepository<Holiday>

{

public HolidayRepository(AppDbContext context) : base(context) { }

public override async Task<IEnumerable<Holiday>> GetAllAsync() => await \_dbSet.ToListAsync();

public override async Task<Holiday?> GetByIdAsync(int id) => await \_dbSet.FirstOrDefaultAsync(h => h.id == id);

}

// ==== D:\piaps\Schedule\Repositories\Implementations\JobTitleRepository.cs ====

﻿using Microsoft.EntityFrameworkCore;

using Models;

using Repositories.Implementations;

using Repositories;

public class JobTitleRepository : BaseRepository<JobTitle>

{

public JobTitleRepository(AppDbContext context) : base(context) { }

public override async Task<IEnumerable<JobTitle>> GetAllAsync()

{

return await \_dbSet.Include(j => j.receptions).ToListAsync();

}

public override async Task<JobTitle?> GetByIdAsync(int id)

{

return await \_dbSet.Include(j => j.receptions).FirstOrDefaultAsync(j => j.id == id);

}

}

// ==== D:\piaps\Schedule\Repositories\Implementations\ReceptionRepository.cs ====

﻿using Microsoft.EntityFrameworkCore;

using Models;

using Repositories.Implementations;

using Repositories;

public class ReceptionRepository : BaseRepository<Reception>

{

public ReceptionRepository(AppDbContext context) : base(context) { }

public override async Task<IEnumerable<Reception>> GetAllAsync()

{

return await \_dbSet

.Include(r => r.requiredPersonnel)

.Include(r => r.department)

.Include(r => r.personnel)

.ThenInclude(p => p.jobTitle)

.ToListAsync();

}

public override async Task<Reception?> GetByIdAsync(int id)

{

return await \_dbSet

.Include(r => r.requiredPersonnel)

.Include(r => r.department)

.Include(r => r.personnel)

.ThenInclude(p => p.jobTitle)

.FirstOrDefaultAsync(r => r.id == id);

}

}

// ==== D:\piaps\Schedule\Repositories\Implementations\WorkerRepository.cs ====

﻿using Microsoft.EntityFrameworkCore;

using Models;

using Repositories.Implementations;

using Repositories;

public class WorkerRepository : BaseRepository<Worker>

{

public WorkerRepository(AppDbContext context) : base(context) { }

public override async Task<IEnumerable<Worker>> GetAllAsync()

{

return await \_dbSet

.Include(w => w.jobTitle)

.Include(w => w.workSchedules)

.Include(w => w.vacations)

.ThenInclude(v => v.days)

.Include(w => w.departments)

.ToListAsync();

}

public override async Task<Worker?> GetByIdAsync(int id)

{

return await \_dbSet

.Include(w => w.jobTitle)

.Include(w => w.workSchedules)

.Include(w => w.vacations)

.ThenInclude(v => v.days)

.Include(w => w.departments)

.FirstOrDefaultAsync(w => w.id == id);

}

}

// ==== D:\piaps\Schedule\Repositories\Interfaces\IRepository.cs ====

﻿namespace Repositories.Interfaces

{

public interface IRepository<T> where T : class

{

Task<IEnumerable<T>> GetAllAsync();

Task<T> GetByIdAsync(int id);

Task AddAsync(T entity);

Task UpdateAsync(T entity);

Task DeleteAsync(T entity);

Task SaveAsync();

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250417201448\_init.cs ====

﻿using System;

using Microsoft.EntityFrameworkCore.Migrations;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

#nullable disable

namespace Repositories.Migrations

{

/// <inheritdoc />

public partial class init : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Departments",

columns: table => new

{

id = table.Column<int>(type: "integer", nullable: false)

.Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

name = table.Column<string>(type: "text", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Departments", x => x.id);

});

migrationBuilder.CreateTable(

name: "Holidays",

columns: table => new

{

id = table.Column<int>(type: "integer", nullable: false)

.Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

date = table.Column<DateOnly>(type: "date", nullable: false),

name = table.Column<string>(type: "text", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Holidays", x => x.id);

});

migrationBuilder.CreateTable(

name: "JobTitles",

columns: table => new

{

id = table.Column<int>(type: "integer", nullable: false)

.Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

name = table.Column<string>(type: "text", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_JobTitles", x => x.id);

});

migrationBuilder.CreateTable(

name: "Receptions",

columns: table => new

{

id = table.Column<int>(type: "integer", nullable: false)

.Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

time = table.Column<TimeSpan>(type: "interval", nullable: false),

departmentid = table.Column<int>(type: "integer", nullable: false),

date = table.Column<DateOnly>(type: "date", nullable: true),

startTime = table.Column<TimeOnly>(type: "time without time zone", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Receptions", x => x.id);

table.ForeignKey(

name: "FK\_Receptions\_Departments\_departmentid",

column: x => x.departmentid,

principalTable: "Departments",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "WorkSchedule4Day",

columns: table => new

{

id = table.Column<int>(type: "integer", nullable: false)

.Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

startOfWork = table.Column<TimeOnly>(type: "time without time zone", nullable: false),

endOfWork = table.Column<TimeOnly>(type: "time without time zone", nullable: false),

isWorking = table.Column<bool>(type: "boolean", nullable: false),

Departmentid = table.Column<int>(type: "integer", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_WorkSchedule4Day", x => x.id);

table.ForeignKey(

name: "FK\_WorkSchedule4Day\_Departments\_Departmentid",

column: x => x.Departmentid,

principalTable: "Departments",

principalColumn: "id");

});

migrationBuilder.CreateTable(

name: "JobTitleReception",

columns: table => new

{

receptionsid = table.Column<int>(type: "integer", nullable: false),

requiredPersonnelid = table.Column<int>(type: "integer", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_JobTitleReception", x => new { x.receptionsid, x.requiredPersonnelid });

table.ForeignKey(

name: "FK\_JobTitleReception\_JobTitles\_requiredPersonnelid",

column: x => x.requiredPersonnelid,

principalTable: "JobTitles",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_JobTitleReception\_Receptions\_receptionsid",

column: x => x.receptionsid,

principalTable: "Receptions",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "Workers",

columns: table => new

{

id = table.Column<int>(type: "integer", nullable: false)

.Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

name = table.Column<string>(type: "text", nullable: false),

login = table.Column<string>(type: "text", nullable: false),

password = table.Column<string>(type: "text", nullable: false),

jobTitleid = table.Column<int>(type: "integer", nullable: false),

Departmentid = table.Column<int>(type: "integer", nullable: true),

Receptionid = table.Column<int>(type: "integer", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Workers", x => x.id);

table.ForeignKey(

name: "FK\_Workers\_Departments\_Departmentid",

column: x => x.Departmentid,

principalTable: "Departments",

principalColumn: "id");

table.ForeignKey(

name: "FK\_Workers\_JobTitles\_jobTitleid",

column: x => x.jobTitleid,

principalTable: "JobTitles",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_Workers\_Receptions\_Receptionid",

column: x => x.Receptionid,

principalTable: "Receptions",

principalColumn: "id");

});

migrationBuilder.CreateTable(

name: "Vacation",

columns: table => new

{

id = table.Column<int>(type: "integer", nullable: false)

.Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

Workerid = table.Column<int>(type: "integer", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Vacation", x => x.id);

table.ForeignKey(

name: "FK\_Vacation\_Workers\_Workerid",

column: x => x.Workerid,

principalTable: "Workers",

principalColumn: "id");

});

migrationBuilder.CreateTable(

name: "DayOff",

columns: table => new

{

id = table.Column<int>(type: "integer", nullable: false)

.Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

date = table.Column<DateOnly>(type: "date", nullable: false),

Vacationid = table.Column<int>(type: "integer", nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_DayOff", x => x.id);

table.ForeignKey(

name: "FK\_DayOff\_Vacation\_Vacationid",

column: x => x.Vacationid,

principalTable: "Vacation",

principalColumn: "id");

});

migrationBuilder.CreateIndex(

name: "IX\_DayOff\_Vacationid",

table: "DayOff",

column: "Vacationid");

migrationBuilder.CreateIndex(

name: "IX\_JobTitleReception\_requiredPersonnelid",

table: "JobTitleReception",

column: "requiredPersonnelid");

migrationBuilder.CreateIndex(

name: "IX\_Receptions\_departmentid",

table: "Receptions",

column: "departmentid");

migrationBuilder.CreateIndex(

name: "IX\_Vacation\_Workerid",

table: "Vacation",

column: "Workerid");

migrationBuilder.CreateIndex(

name: "IX\_WorkSchedule4Day\_Departmentid",

table: "WorkSchedule4Day",

column: "Departmentid");

migrationBuilder.CreateIndex(

name: "IX\_Workers\_Departmentid",

table: "Workers",

column: "Departmentid");

migrationBuilder.CreateIndex(

name: "IX\_Workers\_Receptionid",

table: "Workers",

column: "Receptionid");

migrationBuilder.CreateIndex(

name: "IX\_Workers\_jobTitleid",

table: "Workers",

column: "jobTitleid");

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "DayOff");

migrationBuilder.DropTable(

name: "Holidays");

migrationBuilder.DropTable(

name: "JobTitleReception");

migrationBuilder.DropTable(

name: "WorkSchedule4Day");

migrationBuilder.DropTable(

name: "Vacation");

migrationBuilder.DropTable(

name: "Workers");

migrationBuilder.DropTable(

name: "JobTitles");

migrationBuilder.DropTable(

name: "Receptions");

migrationBuilder.DropTable(

name: "Departments");

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250417201448\_init.Designer.cs ====

﻿// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

using Repositories;

#nullable disable

namespace Repositories.Migrations

{

[DbContext(typeof(AppDbContext))]

[Migration("20250417201448\_init")]

partial class init

{

/// <inheritdoc />

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "9.0.4")

.HasAnnotation("Relational:MaxIdentifierLength", 63);

NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

modelBuilder.Entity("JobTitleReception", b =>

{

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.Property<int>("requiredPersonnelid")

.HasColumnType("integer");

b.HasKey("receptionsid", "requiredPersonnelid");

b.HasIndex("requiredPersonnelid");

b.ToTable("JobTitleReception");

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("Vacationid")

.HasColumnType("integer");

b.Property<DateOnly>("date")

.HasColumnType("date");

b.HasKey("id");

b.HasIndex("Vacationid");

b.ToTable("DayOff");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Departments");

});

modelBuilder.Entity("Models.Holiday", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly>("date")

.HasColumnType("date");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Holidays");

});

modelBuilder.Entity("Models.JobTitle", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("JobTitles");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly?>("date")

.HasColumnType("date");

b.Property<int>("departmentid")

.HasColumnType("integer");

b.Property<TimeOnly?>("startTime")

.HasColumnType("time without time zone");

b.Property<TimeSpan>("time")

.HasColumnType("interval");

b.HasKey("id");

b.HasIndex("departmentid");

b.ToTable("Receptions");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("Workerid")

.HasColumnType("integer");

b.HasKey("id");

b.HasIndex("Workerid");

b.ToTable("Vacation");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("Departmentid")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("Departmentid");

b.ToTable("WorkSchedule4Day");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("Departmentid")

.HasColumnType("integer");

b.Property<int?>("Receptionid")

.HasColumnType("integer");

b.Property<int>("jobTitleid")

.HasColumnType("integer");

b.Property<string>("login")

.IsRequired()

.HasColumnType("text");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.Property<string>("password")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.HasIndex("Departmentid");

b.HasIndex("Receptionid");

b.HasIndex("jobTitleid");

b.ToTable("Workers");

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.JobTitle", null)

.WithMany()

.HasForeignKey("requiredPersonnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.HasOne("Models.Vacation", null)

.WithMany("days")

.HasForeignKey("Vacationid");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.HasOne("Models.Department", "department")

.WithMany()

.HasForeignKey("departmentid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("department");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.HasOne("Models.Worker", null)

.WithMany("vacations")

.HasForeignKey("Workerid");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.HasOne("Models.Department", null)

.WithMany("workSchedules")

.HasForeignKey("Departmentid");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.HasOne("Models.Department", null)

.WithMany("workers")

.HasForeignKey("Departmentid");

b.HasOne("Models.Reception", null)

.WithMany("personnel")

.HasForeignKey("Receptionid");

b.HasOne("Models.JobTitle", "jobTitle")

.WithMany()

.HasForeignKey("jobTitleid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("jobTitle");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Navigation("workSchedules");

b.Navigation("workers");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Navigation("personnel");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Navigation("days");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Navigation("vacations");

});

#pragma warning restore 612, 618

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250417210244\_init2.cs ====

﻿using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Repositories.Migrations

{

/// <inheritdoc />

public partial class init2 : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_DayOff\_Vacation\_Vacationid",

table: "DayOff");

migrationBuilder.DropForeignKey(

name: "FK\_Vacation\_Workers\_Workerid",

table: "Vacation");

migrationBuilder.DropForeignKey(

name: "FK\_WorkSchedule4Day\_Departments\_Departmentid",

table: "WorkSchedule4Day");

migrationBuilder.RenameColumn(

name: "Departmentid",

table: "WorkSchedule4Day",

newName: "DepartmentId");

migrationBuilder.RenameIndex(

name: "IX\_WorkSchedule4Day\_Departmentid",

table: "WorkSchedule4Day",

newName: "IX\_WorkSchedule4Day\_DepartmentId");

migrationBuilder.RenameColumn(

name: "Workerid",

table: "Vacation",

newName: "WorkerId");

migrationBuilder.RenameIndex(

name: "IX\_Vacation\_Workerid",

table: "Vacation",

newName: "IX\_Vacation\_WorkerId");

migrationBuilder.RenameColumn(

name: "Vacationid",

table: "DayOff",

newName: "VacationId");

migrationBuilder.RenameIndex(

name: "IX\_DayOff\_Vacationid",

table: "DayOff",

newName: "IX\_DayOff\_VacationId");

migrationBuilder.AlterColumn<int>(

name: "DepartmentId",

table: "WorkSchedule4Day",

type: "integer",

nullable: false,

defaultValue: 0,

oldClrType: typeof(int),

oldType: "integer",

oldNullable: true);

migrationBuilder.AlterColumn<int>(

name: "WorkerId",

table: "Vacation",

type: "integer",

nullable: false,

defaultValue: 0,

oldClrType: typeof(int),

oldType: "integer",

oldNullable: true);

migrationBuilder.AlterColumn<int>(

name: "VacationId",

table: "DayOff",

type: "integer",

nullable: false,

defaultValue: 0,

oldClrType: typeof(int),

oldType: "integer",

oldNullable: true);

migrationBuilder.AddForeignKey(

name: "FK\_DayOff\_Vacation\_VacationId",

table: "DayOff",

column: "VacationId",

principalTable: "Vacation",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

migrationBuilder.AddForeignKey(

name: "FK\_Vacation\_Workers\_WorkerId",

table: "Vacation",

column: "WorkerId",

principalTable: "Workers",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

migrationBuilder.AddForeignKey(

name: "FK\_WorkSchedule4Day\_Departments\_DepartmentId",

table: "WorkSchedule4Day",

column: "DepartmentId",

principalTable: "Departments",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_DayOff\_Vacation\_VacationId",

table: "DayOff");

migrationBuilder.DropForeignKey(

name: "FK\_Vacation\_Workers\_WorkerId",

table: "Vacation");

migrationBuilder.DropForeignKey(

name: "FK\_WorkSchedule4Day\_Departments\_DepartmentId",

table: "WorkSchedule4Day");

migrationBuilder.RenameColumn(

name: "DepartmentId",

table: "WorkSchedule4Day",

newName: "Departmentid");

migrationBuilder.RenameIndex(

name: "IX\_WorkSchedule4Day\_DepartmentId",

table: "WorkSchedule4Day",

newName: "IX\_WorkSchedule4Day\_Departmentid");

migrationBuilder.RenameColumn(

name: "WorkerId",

table: "Vacation",

newName: "Workerid");

migrationBuilder.RenameIndex(

name: "IX\_Vacation\_WorkerId",

table: "Vacation",

newName: "IX\_Vacation\_Workerid");

migrationBuilder.RenameColumn(

name: "VacationId",

table: "DayOff",

newName: "Vacationid");

migrationBuilder.RenameIndex(

name: "IX\_DayOff\_VacationId",

table: "DayOff",

newName: "IX\_DayOff\_Vacationid");

migrationBuilder.AlterColumn<int>(

name: "Departmentid",

table: "WorkSchedule4Day",

type: "integer",

nullable: true,

oldClrType: typeof(int),

oldType: "integer");

migrationBuilder.AlterColumn<int>(

name: "Workerid",

table: "Vacation",

type: "integer",

nullable: true,

oldClrType: typeof(int),

oldType: "integer");

migrationBuilder.AlterColumn<int>(

name: "Vacationid",

table: "DayOff",

type: "integer",

nullable: true,

oldClrType: typeof(int),

oldType: "integer");

migrationBuilder.AddForeignKey(

name: "FK\_DayOff\_Vacation\_Vacationid",

table: "DayOff",

column: "Vacationid",

principalTable: "Vacation",

principalColumn: "id");

migrationBuilder.AddForeignKey(

name: "FK\_Vacation\_Workers\_Workerid",

table: "Vacation",

column: "Workerid",

principalTable: "Workers",

principalColumn: "id");

migrationBuilder.AddForeignKey(

name: "FK\_WorkSchedule4Day\_Departments\_Departmentid",

table: "WorkSchedule4Day",

column: "Departmentid",

principalTable: "Departments",

principalColumn: "id");

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250417210244\_init2.Designer.cs ====

﻿// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

using Repositories;

#nullable disable

namespace Repositories.Migrations

{

[DbContext(typeof(AppDbContext))]

[Migration("20250417210244\_init2")]

partial class init2

{

/// <inheritdoc />

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "9.0.4")

.HasAnnotation("Relational:MaxIdentifierLength", 63);

NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

modelBuilder.Entity("JobTitleReception", b =>

{

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.Property<int>("requiredPersonnelid")

.HasColumnType("integer");

b.HasKey("receptionsid", "requiredPersonnelid");

b.HasIndex("requiredPersonnelid");

b.ToTable("JobTitleReception");

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("VacationId")

.HasColumnType("integer");

b.Property<DateOnly>("date")

.HasColumnType("date");

b.HasKey("id");

b.HasIndex("VacationId");

b.ToTable("DayOff");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Departments");

});

modelBuilder.Entity("Models.Holiday", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly>("date")

.HasColumnType("date");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Holidays");

});

modelBuilder.Entity("Models.JobTitle", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("JobTitles");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly?>("date")

.HasColumnType("date");

b.Property<int>("departmentid")

.HasColumnType("integer");

b.Property<TimeOnly?>("startTime")

.HasColumnType("time without time zone");

b.Property<TimeSpan>("time")

.HasColumnType("interval");

b.HasKey("id");

b.HasIndex("departmentid");

b.ToTable("Receptions");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("Vacation");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("DepartmentId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("DepartmentId");

b.ToTable("WorkSchedule4Day");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("Departmentid")

.HasColumnType("integer");

b.Property<int?>("Receptionid")

.HasColumnType("integer");

b.Property<int>("jobTitleid")

.HasColumnType("integer");

b.Property<string>("login")

.IsRequired()

.HasColumnType("text");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.Property<string>("password")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.HasIndex("Departmentid");

b.HasIndex("Receptionid");

b.HasIndex("jobTitleid");

b.ToTable("Workers");

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.JobTitle", null)

.WithMany()

.HasForeignKey("requiredPersonnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.HasOne("Models.Vacation", "Vacation")

.WithMany("days")

.HasForeignKey("VacationId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Vacation");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.HasOne("Models.Department", "department")

.WithMany()

.HasForeignKey("departmentid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("department");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("vacations")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.HasOne("Models.Department", "Department")

.WithMany("workSchedules")

.HasForeignKey("DepartmentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Department");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.HasOne("Models.Department", null)

.WithMany("workers")

.HasForeignKey("Departmentid");

b.HasOne("Models.Reception", null)

.WithMany("personnel")

.HasForeignKey("Receptionid");

b.HasOne("Models.JobTitle", "jobTitle")

.WithMany()

.HasForeignKey("jobTitleid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("jobTitle");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Navigation("workSchedules");

b.Navigation("workers");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Navigation("personnel");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Navigation("days");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Navigation("vacations");

});

#pragma warning restore 612, 618

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250417210353\_init3.cs ====

﻿using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Repositories.Migrations

{

/// <inheritdoc />

public partial class init3 : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_Workers\_Departments\_Departmentid",

table: "Workers");

migrationBuilder.DropIndex(

name: "IX\_Workers\_Departmentid",

table: "Workers");

migrationBuilder.DropColumn(

name: "Departmentid",

table: "Workers");

migrationBuilder.CreateTable(

name: "DepartmentWorker",

columns: table => new

{

departmentsid = table.Column<int>(type: "integer", nullable: false),

workersid = table.Column<int>(type: "integer", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_DepartmentWorker", x => new { x.departmentsid, x.workersid });

table.ForeignKey(

name: "FK\_DepartmentWorker\_Departments\_departmentsid",

column: x => x.departmentsid,

principalTable: "Departments",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_DepartmentWorker\_Workers\_workersid",

column: x => x.workersid,

principalTable: "Workers",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_DepartmentWorker\_workersid",

table: "DepartmentWorker",

column: "workersid");

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "DepartmentWorker");

migrationBuilder.AddColumn<int>(

name: "Departmentid",

table: "Workers",

type: "integer",

nullable: true);

migrationBuilder.CreateIndex(

name: "IX\_Workers\_Departmentid",

table: "Workers",

column: "Departmentid");

migrationBuilder.AddForeignKey(

name: "FK\_Workers\_Departments\_Departmentid",

table: "Workers",

column: "Departmentid",

principalTable: "Departments",

principalColumn: "id");

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250417210353\_init3.Designer.cs ====

﻿// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

using Repositories;

#nullable disable

namespace Repositories.Migrations

{

[DbContext(typeof(AppDbContext))]

[Migration("20250417210353\_init3")]

partial class init3

{

/// <inheritdoc />

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "9.0.4")

.HasAnnotation("Relational:MaxIdentifierLength", 63);

NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

modelBuilder.Entity("DepartmentWorker", b =>

{

b.Property<int>("departmentsid")

.HasColumnType("integer");

b.Property<int>("workersid")

.HasColumnType("integer");

b.HasKey("departmentsid", "workersid");

b.HasIndex("workersid");

b.ToTable("DepartmentWorker");

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.Property<int>("requiredPersonnelid")

.HasColumnType("integer");

b.HasKey("receptionsid", "requiredPersonnelid");

b.HasIndex("requiredPersonnelid");

b.ToTable("JobTitleReception");

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("VacationId")

.HasColumnType("integer");

b.Property<DateOnly>("date")

.HasColumnType("date");

b.HasKey("id");

b.HasIndex("VacationId");

b.ToTable("DayOff");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Departments");

});

modelBuilder.Entity("Models.Holiday", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly>("date")

.HasColumnType("date");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Holidays");

});

modelBuilder.Entity("Models.JobTitle", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("JobTitles");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly?>("date")

.HasColumnType("date");

b.Property<int>("departmentid")

.HasColumnType("integer");

b.Property<TimeOnly?>("startTime")

.HasColumnType("time without time zone");

b.Property<TimeSpan>("time")

.HasColumnType("interval");

b.HasKey("id");

b.HasIndex("departmentid");

b.ToTable("Receptions");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("Vacation");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("DepartmentId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("DepartmentId");

b.ToTable("WorkSchedule4Day");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("Receptionid")

.HasColumnType("integer");

b.Property<int>("jobTitleid")

.HasColumnType("integer");

b.Property<string>("login")

.IsRequired()

.HasColumnType("text");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.Property<string>("password")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.HasIndex("Receptionid");

b.HasIndex("jobTitleid");

b.ToTable("Workers");

});

modelBuilder.Entity("DepartmentWorker", b =>

{

b.HasOne("Models.Department", null)

.WithMany()

.HasForeignKey("departmentsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("workersid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.JobTitle", null)

.WithMany()

.HasForeignKey("requiredPersonnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.HasOne("Models.Vacation", "Vacation")

.WithMany("days")

.HasForeignKey("VacationId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Vacation");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.HasOne("Models.Department", "department")

.WithMany()

.HasForeignKey("departmentid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("department");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("vacations")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.HasOne("Models.Department", "Department")

.WithMany("workSchedules")

.HasForeignKey("DepartmentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Department");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.HasOne("Models.Reception", null)

.WithMany("personnel")

.HasForeignKey("Receptionid");

b.HasOne("Models.JobTitle", "jobTitle")

.WithMany()

.HasForeignKey("jobTitleid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("jobTitle");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Navigation("workSchedules");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Navigation("personnel");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Navigation("days");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Navigation("vacations");

});

#pragma warning restore 612, 618

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250417220912\_init4.cs ====

﻿using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Repositories.Migrations

{

/// <inheritdoc />

public partial class init4 : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250417220912\_init4.Designer.cs ====

﻿// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

using Repositories;

#nullable disable

namespace Repositories.Migrations

{

[DbContext(typeof(AppDbContext))]

[Migration("20250417220912\_init4")]

partial class init4

{

/// <inheritdoc />

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "9.0.4")

.HasAnnotation("Relational:MaxIdentifierLength", 63);

NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

modelBuilder.Entity("DepartmentWorker", b =>

{

b.Property<int>("departmentsid")

.HasColumnType("integer");

b.Property<int>("workersid")

.HasColumnType("integer");

b.HasKey("departmentsid", "workersid");

b.HasIndex("workersid");

b.ToTable("DepartmentWorker");

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.Property<int>("requiredPersonnelid")

.HasColumnType("integer");

b.HasKey("receptionsid", "requiredPersonnelid");

b.HasIndex("requiredPersonnelid");

b.ToTable("JobTitleReception");

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("VacationId")

.HasColumnType("integer");

b.Property<DateOnly>("date")

.HasColumnType("date");

b.HasKey("id");

b.HasIndex("VacationId");

b.ToTable("DayOff");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Departments");

});

modelBuilder.Entity("Models.Holiday", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly>("date")

.HasColumnType("date");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Holidays");

});

modelBuilder.Entity("Models.JobTitle", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("JobTitles");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly?>("date")

.HasColumnType("date");

b.Property<int>("departmentid")

.HasColumnType("integer");

b.Property<TimeOnly?>("startTime")

.HasColumnType("time without time zone");

b.Property<TimeSpan>("time")

.HasColumnType("interval");

b.HasKey("id");

b.HasIndex("departmentid");

b.ToTable("Receptions");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("Vacation");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("DepartmentId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("DepartmentId");

b.ToTable("WorkSchedule4Day");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("Receptionid")

.HasColumnType("integer");

b.Property<int>("jobTitleid")

.HasColumnType("integer");

b.Property<string>("login")

.IsRequired()

.HasColumnType("text");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.Property<string>("password")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.HasIndex("Receptionid");

b.HasIndex("jobTitleid");

b.ToTable("Workers");

});

modelBuilder.Entity("DepartmentWorker", b =>

{

b.HasOne("Models.Department", null)

.WithMany()

.HasForeignKey("departmentsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("workersid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.JobTitle", null)

.WithMany()

.HasForeignKey("requiredPersonnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.HasOne("Models.Vacation", "Vacation")

.WithMany("days")

.HasForeignKey("VacationId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Vacation");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.HasOne("Models.Department", "department")

.WithMany()

.HasForeignKey("departmentid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("department");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("vacations")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.HasOne("Models.Department", "Department")

.WithMany("workSchedules")

.HasForeignKey("DepartmentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Department");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.HasOne("Models.Reception", null)

.WithMany("personnel")

.HasForeignKey("Receptionid");

b.HasOne("Models.JobTitle", "jobTitle")

.WithMany()

.HasForeignKey("jobTitleid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("jobTitle");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Navigation("workSchedules");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Navigation("personnel");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Navigation("days");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Navigation("vacations");

});

#pragma warning restore 612, 618

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250526124500\_Initial.cs ====

﻿using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Repositories.Migrations

{

/// <inheritdoc />

public partial class Initial : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_Workers\_JobTitles\_jobTitleid",

table: "Workers");

migrationBuilder.AlterColumn<string>(

name: "password",

table: "Workers",

type: "text",

nullable: true,

oldClrType: typeof(string),

oldType: "text");

migrationBuilder.AlterColumn<string>(

name: "name",

table: "Workers",

type: "text",

nullable: true,

oldClrType: typeof(string),

oldType: "text");

migrationBuilder.AlterColumn<string>(

name: "login",

table: "Workers",

type: "text",

nullable: true,

oldClrType: typeof(string),

oldType: "text");

migrationBuilder.AlterColumn<int>(

name: "jobTitleid",

table: "Workers",

type: "integer",

nullable: true,

oldClrType: typeof(int),

oldType: "integer");

migrationBuilder.AlterColumn<string>(

name: "name",

table: "JobTitles",

type: "text",

nullable: true,

oldClrType: typeof(string),

oldType: "text");

migrationBuilder.AlterColumn<string>(

name: "name",

table: "Departments",

type: "text",

nullable: true,

oldClrType: typeof(string),

oldType: "text");

migrationBuilder.AddForeignKey(

name: "FK\_Workers\_JobTitles\_jobTitleid",

table: "Workers",

column: "jobTitleid",

principalTable: "JobTitles",

principalColumn: "id");

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_Workers\_JobTitles\_jobTitleid",

table: "Workers");

migrationBuilder.AlterColumn<string>(

name: "password",

table: "Workers",

type: "text",

nullable: false,

defaultValue: "",

oldClrType: typeof(string),

oldType: "text",

oldNullable: true);

migrationBuilder.AlterColumn<string>(

name: "name",

table: "Workers",

type: "text",

nullable: false,

defaultValue: "",

oldClrType: typeof(string),

oldType: "text",

oldNullable: true);

migrationBuilder.AlterColumn<string>(

name: "login",

table: "Workers",

type: "text",

nullable: false,

defaultValue: "",

oldClrType: typeof(string),

oldType: "text",

oldNullable: true);

migrationBuilder.AlterColumn<int>(

name: "jobTitleid",

table: "Workers",

type: "integer",

nullable: false,

defaultValue: 0,

oldClrType: typeof(int),

oldType: "integer",

oldNullable: true);

migrationBuilder.AlterColumn<string>(

name: "name",

table: "JobTitles",

type: "text",

nullable: false,

defaultValue: "",

oldClrType: typeof(string),

oldType: "text",

oldNullable: true);

migrationBuilder.AlterColumn<string>(

name: "name",

table: "Departments",

type: "text",

nullable: false,

defaultValue: "",

oldClrType: typeof(string),

oldType: "text",

oldNullable: true);

migrationBuilder.AddForeignKey(

name: "FK\_Workers\_JobTitles\_jobTitleid",

table: "Workers",

column: "jobTitleid",

principalTable: "JobTitles",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250526124500\_Initial.Designer.cs ====

﻿// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

using Repositories;

#nullable disable

namespace Repositories.Migrations

{

[DbContext(typeof(AppDbContext))]

[Migration("20250526124500\_Initial")]

partial class Initial

{

/// <inheritdoc />

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "9.0.5")

.HasAnnotation("Relational:MaxIdentifierLength", 63);

NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

modelBuilder.Entity("DepartmentWorker", b =>

{

b.Property<int>("departmentsid")

.HasColumnType("integer");

b.Property<int>("workersid")

.HasColumnType("integer");

b.HasKey("departmentsid", "workersid");

b.HasIndex("workersid");

b.ToTable("DepartmentWorker");

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.Property<int>("requiredPersonnelid")

.HasColumnType("integer");

b.HasKey("receptionsid", "requiredPersonnelid");

b.HasIndex("requiredPersonnelid");

b.ToTable("JobTitleReception");

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("VacationId")

.HasColumnType("integer");

b.Property<DateOnly>("date")

.HasColumnType("date");

b.HasKey("id");

b.HasIndex("VacationId");

b.ToTable("DayOff");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Departments");

});

modelBuilder.Entity("Models.Holiday", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly>("date")

.HasColumnType("date");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Holidays");

});

modelBuilder.Entity("Models.JobTitle", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("JobTitles");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly?>("date")

.HasColumnType("date");

b.Property<int>("departmentid")

.HasColumnType("integer");

b.Property<TimeOnly?>("startTime")

.HasColumnType("time without time zone");

b.Property<TimeSpan>("time")

.HasColumnType("interval");

b.HasKey("id");

b.HasIndex("departmentid");

b.ToTable("Receptions");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("Vacation");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("DepartmentId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("DepartmentId");

b.ToTable("WorkSchedule4Day");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("Receptionid")

.HasColumnType("integer");

b.Property<int?>("jobTitleid")

.HasColumnType("integer");

b.Property<string>("login")

.HasColumnType("text");

b.Property<string>("name")

.HasColumnType("text");

b.Property<string>("password")

.HasColumnType("text");

b.HasKey("id");

b.HasIndex("Receptionid");

b.HasIndex("jobTitleid");

b.ToTable("Workers");

});

modelBuilder.Entity("DepartmentWorker", b =>

{

b.HasOne("Models.Department", null)

.WithMany()

.HasForeignKey("departmentsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("workersid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.JobTitle", null)

.WithMany()

.HasForeignKey("requiredPersonnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.HasOne("Models.Vacation", "Vacation")

.WithMany("days")

.HasForeignKey("VacationId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Vacation");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.HasOne("Models.Department", "department")

.WithMany()

.HasForeignKey("departmentid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("department");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("vacations")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.HasOne("Models.Department", "Department")

.WithMany("workSchedules")

.HasForeignKey("DepartmentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Department");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.HasOne("Models.Reception", null)

.WithMany("personnel")

.HasForeignKey("Receptionid");

b.HasOne("Models.JobTitle", "jobTitle")

.WithMany()

.HasForeignKey("jobTitleid");

b.Navigation("jobTitle");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Navigation("workSchedules");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Navigation("personnel");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Navigation("days");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Navigation("vacations");

});

#pragma warning restore 612, 618

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250526223546\_initital2.cs ====

﻿using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Repositories.Migrations

{

/// <inheritdoc />

public partial class initital2 : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_Workers\_Receptions\_Receptionid",

table: "Workers");

migrationBuilder.DropIndex(

name: "IX\_Workers\_Receptionid",

table: "Workers");

migrationBuilder.DropColumn(

name: "Receptionid",

table: "Workers");

migrationBuilder.CreateTable(

name: "ReceptionWorker",

columns: table => new

{

personnelid = table.Column<int>(type: "integer", nullable: false),

receptionsid = table.Column<int>(type: "integer", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_ReceptionWorker", x => new { x.personnelid, x.receptionsid });

table.ForeignKey(

name: "FK\_ReceptionWorker\_Receptions\_receptionsid",

column: x => x.receptionsid,

principalTable: "Receptions",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_ReceptionWorker\_Workers\_personnelid",

column: x => x.personnelid,

principalTable: "Workers",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_ReceptionWorker\_receptionsid",

table: "ReceptionWorker",

column: "receptionsid");

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "ReceptionWorker");

migrationBuilder.AddColumn<int>(

name: "Receptionid",

table: "Workers",

type: "integer",

nullable: true);

migrationBuilder.CreateIndex(

name: "IX\_Workers\_Receptionid",

table: "Workers",

column: "Receptionid");

migrationBuilder.AddForeignKey(

name: "FK\_Workers\_Receptions\_Receptionid",

table: "Workers",

column: "Receptionid",

principalTable: "Receptions",

principalColumn: "id");

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250526223546\_initital2.Designer.cs ====

﻿// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

using Repositories;

#nullable disable

namespace Repositories.Migrations

{

[DbContext(typeof(AppDbContext))]

[Migration("20250526223546\_initital2")]

partial class initital2

{

/// <inheritdoc />

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "9.0.5")

.HasAnnotation("Relational:MaxIdentifierLength", 63);

NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

modelBuilder.Entity("DepartmentWorker", b =>

{

b.Property<int>("departmentsid")

.HasColumnType("integer");

b.Property<int>("workersid")

.HasColumnType("integer");

b.HasKey("departmentsid", "workersid");

b.HasIndex("workersid");

b.ToTable("DepartmentWorker");

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.Property<int>("requiredPersonnelid")

.HasColumnType("integer");

b.HasKey("receptionsid", "requiredPersonnelid");

b.HasIndex("requiredPersonnelid");

b.ToTable("JobTitleReception");

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("VacationId")

.HasColumnType("integer");

b.Property<DateOnly>("date")

.HasColumnType("date");

b.HasKey("id");

b.HasIndex("VacationId");

b.ToTable("DayOff");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Departments");

});

modelBuilder.Entity("Models.Holiday", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly>("date")

.HasColumnType("date");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Holidays");

});

modelBuilder.Entity("Models.JobTitle", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("JobTitles");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly?>("date")

.HasColumnType("date");

b.Property<int>("departmentid")

.HasColumnType("integer");

b.Property<TimeOnly?>("startTime")

.HasColumnType("time without time zone");

b.Property<TimeSpan>("time")

.HasColumnType("interval");

b.HasKey("id");

b.HasIndex("departmentid");

b.ToTable("Receptions");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("Vacation");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("DepartmentId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("DepartmentId");

b.ToTable("WorkSchedule4Day");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("jobTitleid")

.HasColumnType("integer");

b.Property<string>("login")

.HasColumnType("text");

b.Property<string>("name")

.HasColumnType("text");

b.Property<string>("password")

.HasColumnType("text");

b.HasKey("id");

b.HasIndex("jobTitleid");

b.ToTable("Workers");

});

modelBuilder.Entity("ReceptionWorker", b =>

{

b.Property<int>("personnelid")

.HasColumnType("integer");

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.HasKey("personnelid", "receptionsid");

b.HasIndex("receptionsid");

b.ToTable("ReceptionWorker");

});

modelBuilder.Entity("DepartmentWorker", b =>

{

b.HasOne("Models.Department", null)

.WithMany()

.HasForeignKey("departmentsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("workersid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.JobTitle", null)

.WithMany()

.HasForeignKey("requiredPersonnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.HasOne("Models.Vacation", "Vacation")

.WithMany("days")

.HasForeignKey("VacationId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Vacation");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.HasOne("Models.Department", "department")

.WithMany()

.HasForeignKey("departmentid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("department");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("vacations")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.HasOne("Models.Department", "Department")

.WithMany("workSchedules")

.HasForeignKey("DepartmentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Department");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.HasOne("Models.JobTitle", "jobTitle")

.WithMany()

.HasForeignKey("jobTitleid");

b.Navigation("jobTitle");

});

modelBuilder.Entity("ReceptionWorker", b =>

{

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("personnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.Department", b =>

{

b.Navigation("workSchedules");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Navigation("days");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Navigation("vacations");

});

#pragma warning restore 612, 618

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250528191431\_initial3.cs ====

﻿using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Repositories.Migrations

{

/// <inheritdoc />

public partial class initial3 : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_Workers\_JobTitles\_jobTitleid",

table: "Workers");

migrationBuilder.RenameColumn(

name: "jobTitleid",

table: "Workers",

newName: "JobTitleId");

migrationBuilder.RenameIndex(

name: "IX\_Workers\_jobTitleid",

table: "Workers",

newName: "IX\_Workers\_JobTitleId");

migrationBuilder.AddForeignKey(

name: "FK\_Workers\_JobTitles\_JobTitleId",

table: "Workers",

column: "JobTitleId",

principalTable: "JobTitles",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_Workers\_JobTitles\_JobTitleId",

table: "Workers");

migrationBuilder.RenameColumn(

name: "JobTitleId",

table: "Workers",

newName: "jobTitleid");

migrationBuilder.RenameIndex(

name: "IX\_Workers\_JobTitleId",

table: "Workers",

newName: "IX\_Workers\_jobTitleid");

migrationBuilder.AddForeignKey(

name: "FK\_Workers\_JobTitles\_jobTitleid",

table: "Workers",

column: "jobTitleid",

principalTable: "JobTitles",

principalColumn: "id");

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250528191431\_initial3.Designer.cs ====

﻿// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

using Repositories;

#nullable disable

namespace Repositories.Migrations

{

[DbContext(typeof(AppDbContext))]

[Migration("20250528191431\_initial3")]

partial class initial3

{

/// <inheritdoc />

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "9.0.5")

.HasAnnotation("Relational:MaxIdentifierLength", 63);

NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

modelBuilder.Entity("DepartmentWorker", b =>

{

b.Property<int>("departmentsid")

.HasColumnType("integer");

b.Property<int>("workersid")

.HasColumnType("integer");

b.HasKey("departmentsid", "workersid");

b.HasIndex("workersid");

b.ToTable("DepartmentWorker");

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.Property<int>("requiredPersonnelid")

.HasColumnType("integer");

b.HasKey("receptionsid", "requiredPersonnelid");

b.HasIndex("requiredPersonnelid");

b.ToTable("JobTitleReception");

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("VacationId")

.HasColumnType("integer");

b.Property<DateOnly>("date")

.HasColumnType("date");

b.HasKey("id");

b.HasIndex("VacationId");

b.ToTable("DayOff");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Departments");

});

modelBuilder.Entity("Models.Holiday", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly>("date")

.HasColumnType("date");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Holidays");

});

modelBuilder.Entity("Models.JobTitle", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("JobTitles");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly?>("date")

.HasColumnType("date");

b.Property<int>("departmentid")

.HasColumnType("integer");

b.Property<TimeOnly?>("startTime")

.HasColumnType("time without time zone");

b.Property<TimeSpan>("time")

.HasColumnType("interval");

b.HasKey("id");

b.HasIndex("departmentid");

b.ToTable("Receptions");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("Vacation");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("DepartmentId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("DepartmentId");

b.ToTable("WorkSchedule4Day");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("JobTitleId")

.HasColumnType("integer");

b.Property<string>("login")

.HasColumnType("text");

b.Property<string>("name")

.HasColumnType("text");

b.Property<string>("password")

.HasColumnType("text");

b.HasKey("id");

b.HasIndex("JobTitleId");

b.ToTable("Workers");

});

modelBuilder.Entity("ReceptionWorker", b =>

{

b.Property<int>("personnelid")

.HasColumnType("integer");

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.HasKey("personnelid", "receptionsid");

b.HasIndex("receptionsid");

b.ToTable("ReceptionWorker");

});

modelBuilder.Entity("DepartmentWorker", b =>

{

b.HasOne("Models.Department", null)

.WithMany()

.HasForeignKey("departmentsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("workersid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.JobTitle", null)

.WithMany()

.HasForeignKey("requiredPersonnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.HasOne("Models.Vacation", "Vacation")

.WithMany("days")

.HasForeignKey("VacationId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Vacation");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.HasOne("Models.Department", "department")

.WithMany()

.HasForeignKey("departmentid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("department");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("vacations")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.HasOne("Models.Department", "Department")

.WithMany("workSchedules")

.HasForeignKey("DepartmentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Department");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.HasOne("Models.JobTitle", "jobTitle")

.WithMany()

.HasForeignKey("JobTitleId")

.OnDelete(DeleteBehavior.Cascade);

b.Navigation("jobTitle");

});

modelBuilder.Entity("ReceptionWorker", b =>

{

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("personnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.Department", b =>

{

b.Navigation("workSchedules");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Navigation("days");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Navigation("vacations");

});

#pragma warning restore 612, 618

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250604175308\_04-06-2025.cs ====

﻿using System;

using Microsoft.EntityFrameworkCore.Migrations;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

#nullable disable

namespace Repositories.Migrations

{

/// <inheritdoc />

public partial class \_04062025 : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "WorkerWorkSchedule4Day",

columns: table => new

{

id = table.Column<int>(type: "integer", nullable: false)

.Annotation("Npgsql:ValueGenerationStrategy", NpgsqlValueGenerationStrategy.IdentityByDefaultColumn),

startOfWork = table.Column<TimeOnly>(type: "time without time zone", nullable: false),

endOfWork = table.Column<TimeOnly>(type: "time without time zone", nullable: false),

isWorking = table.Column<bool>(type: "boolean", nullable: false),

WorkerId = table.Column<int>(type: "integer", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_WorkerWorkSchedule4Day", x => x.id);

table.ForeignKey(

name: "FK\_WorkerWorkSchedule4Day\_Workers\_WorkerId",

column: x => x.WorkerId,

principalTable: "Workers",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_WorkerWorkSchedule4Day\_WorkerId",

table: "WorkerWorkSchedule4Day",

column: "WorkerId");

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "WorkerWorkSchedule4Day");

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250604175308\_04-06-2025.Designer.cs ====

﻿// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

using Repositories;

#nullable disable

namespace Repositories.Migrations

{

[DbContext(typeof(AppDbContext))]

[Migration("20250604175308\_04-06-2025")]

partial class \_04062025

{

/// <inheritdoc />

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "9.0.5")

.HasAnnotation("Relational:MaxIdentifierLength", 63);

NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

modelBuilder.Entity("DepartmentWorker", b =>

{

b.Property<int>("departmentsid")

.HasColumnType("integer");

b.Property<int>("workersid")

.HasColumnType("integer");

b.HasKey("departmentsid", "workersid");

b.HasIndex("workersid");

b.ToTable("DepartmentWorker");

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.Property<int>("requiredPersonnelid")

.HasColumnType("integer");

b.HasKey("receptionsid", "requiredPersonnelid");

b.HasIndex("requiredPersonnelid");

b.ToTable("JobTitleReception");

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("VacationId")

.HasColumnType("integer");

b.Property<DateOnly>("date")

.HasColumnType("date");

b.HasKey("id");

b.HasIndex("VacationId");

b.ToTable("DayOff");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Departments");

});

modelBuilder.Entity("Models.Holiday", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly>("date")

.HasColumnType("date");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Holidays");

});

modelBuilder.Entity("Models.JobTitle", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("JobTitles");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly?>("date")

.HasColumnType("date");

b.Property<int>("departmentid")

.HasColumnType("integer");

b.Property<TimeOnly?>("startTime")

.HasColumnType("time without time zone");

b.Property<TimeSpan>("time")

.HasColumnType("interval");

b.HasKey("id");

b.HasIndex("departmentid");

b.ToTable("Receptions");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("Vacation");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("DepartmentId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("DepartmentId");

b.ToTable("WorkSchedule4Day");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("JobTitleId")

.HasColumnType("integer");

b.Property<string>("login")

.HasColumnType("text");

b.Property<string>("name")

.HasColumnType("text");

b.Property<string>("password")

.HasColumnType("text");

b.HasKey("id");

b.HasIndex("JobTitleId");

b.ToTable("Workers");

});

modelBuilder.Entity("Models.WorkerWorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("WorkerWorkSchedule4Day");

});

modelBuilder.Entity("ReceptionWorker", b =>

{

b.Property<int>("personnelid")

.HasColumnType("integer");

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.HasKey("personnelid", "receptionsid");

b.HasIndex("receptionsid");

b.ToTable("ReceptionWorker");

});

modelBuilder.Entity("DepartmentWorker", b =>

{

b.HasOne("Models.Department", null)

.WithMany()

.HasForeignKey("departmentsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("workersid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.JobTitle", null)

.WithMany()

.HasForeignKey("requiredPersonnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.HasOne("Models.Vacation", "Vacation")

.WithMany("days")

.HasForeignKey("VacationId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Vacation");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.HasOne("Models.Department", "department")

.WithMany()

.HasForeignKey("departmentid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("department");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("vacations")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.HasOne("Models.Department", "Department")

.WithMany("workSchedules")

.HasForeignKey("DepartmentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Department");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.HasOne("Models.JobTitle", "jobTitle")

.WithMany()

.HasForeignKey("JobTitleId")

.OnDelete(DeleteBehavior.Cascade);

b.Navigation("jobTitle");

});

modelBuilder.Entity("Models.WorkerWorkSchedule4Day", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("workSchedules")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("ReceptionWorker", b =>

{

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("personnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.Department", b =>

{

b.Navigation("workSchedules");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Navigation("days");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Navigation("vacations");

b.Navigation("workSchedules");

});

#pragma warning restore 612, 618

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250604175729\_04-06-2025-2.cs ====

﻿using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Repositories.Migrations

{

/// <inheritdoc />

public partial class \_040620252 : Migration

{

/// <inheritdoc />

protected override void Up(MigrationBuilder migrationBuilder)

{

}

/// <inheritdoc />

protected override void Down(MigrationBuilder migrationBuilder)

{

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\20250604175729\_04-06-2025-2.Designer.cs ====

﻿// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

using Repositories;

#nullable disable

namespace Repositories.Migrations

{

[DbContext(typeof(AppDbContext))]

[Migration("20250604175729\_04-06-2025-2")]

partial class \_040620252

{

/// <inheritdoc />

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "9.0.5")

.HasAnnotation("Relational:MaxIdentifierLength", 63);

NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

modelBuilder.Entity("DepartmentWorker", b =>

{

b.Property<int>("departmentsid")

.HasColumnType("integer");

b.Property<int>("workersid")

.HasColumnType("integer");

b.HasKey("departmentsid", "workersid");

b.HasIndex("workersid");

b.ToTable("DepartmentWorker");

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.Property<int>("requiredPersonnelid")

.HasColumnType("integer");

b.HasKey("receptionsid", "requiredPersonnelid");

b.HasIndex("requiredPersonnelid");

b.ToTable("JobTitleReception");

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("VacationId")

.HasColumnType("integer");

b.Property<DateOnly>("date")

.HasColumnType("date");

b.HasKey("id");

b.HasIndex("VacationId");

b.ToTable("DayOff");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Departments");

});

modelBuilder.Entity("Models.Holiday", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly>("date")

.HasColumnType("date");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Holidays");

});

modelBuilder.Entity("Models.JobTitle", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("JobTitles");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly?>("date")

.HasColumnType("date");

b.Property<int>("departmentid")

.HasColumnType("integer");

b.Property<TimeOnly?>("startTime")

.HasColumnType("time without time zone");

b.Property<TimeSpan>("time")

.HasColumnType("interval");

b.HasKey("id");

b.HasIndex("departmentid");

b.ToTable("Receptions");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("Vacation");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("DepartmentId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("DepartmentId");

b.ToTable("WorkSchedule4Day");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("JobTitleId")

.HasColumnType("integer");

b.Property<string>("login")

.HasColumnType("text");

b.Property<string>("name")

.HasColumnType("text");

b.Property<string>("password")

.HasColumnType("text");

b.HasKey("id");

b.HasIndex("JobTitleId");

b.ToTable("Workers");

});

modelBuilder.Entity("Models.WorkerWorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("WorkerWorkSchedule4Day");

});

modelBuilder.Entity("ReceptionWorker", b =>

{

b.Property<int>("personnelid")

.HasColumnType("integer");

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.HasKey("personnelid", "receptionsid");

b.HasIndex("receptionsid");

b.ToTable("ReceptionWorker");

});

modelBuilder.Entity("DepartmentWorker", b =>

{

b.HasOne("Models.Department", null)

.WithMany()

.HasForeignKey("departmentsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("workersid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.JobTitle", null)

.WithMany()

.HasForeignKey("requiredPersonnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.HasOne("Models.Vacation", "Vacation")

.WithMany("days")

.HasForeignKey("VacationId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Vacation");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.HasOne("Models.Department", "department")

.WithMany()

.HasForeignKey("departmentid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("department");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("vacations")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.HasOne("Models.Department", "Department")

.WithMany("workSchedules")

.HasForeignKey("DepartmentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Department");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.HasOne("Models.JobTitle", "jobTitle")

.WithMany()

.HasForeignKey("JobTitleId")

.OnDelete(DeleteBehavior.Cascade);

b.Navigation("jobTitle");

});

modelBuilder.Entity("Models.WorkerWorkSchedule4Day", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("workSchedules")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("ReceptionWorker", b =>

{

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("personnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.Department", b =>

{

b.Navigation("workSchedules");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Navigation("days");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Navigation("vacations");

b.Navigation("workSchedules");

});

#pragma warning restore 612, 618

}

}

}

// ==== D:\piaps\Schedule\Repositories\Migrations\AppDbContextModelSnapshot.cs ====

﻿// <auto-generated />

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

using Npgsql.EntityFrameworkCore.PostgreSQL.Metadata;

using Repositories;

#nullable disable

namespace Repositories.Migrations

{

[DbContext(typeof(AppDbContext))]

partial class AppDbContextModelSnapshot : ModelSnapshot

{

protected override void BuildModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "9.0.5")

.HasAnnotation("Relational:MaxIdentifierLength", 63);

NpgsqlModelBuilderExtensions.UseIdentityByDefaultColumns(modelBuilder);

modelBuilder.Entity("DepartmentWorker", b =>

{

b.Property<int>("departmentsid")

.HasColumnType("integer");

b.Property<int>("workersid")

.HasColumnType("integer");

b.HasKey("departmentsid", "workersid");

b.HasIndex("workersid");

b.ToTable("DepartmentWorker");

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.Property<int>("requiredPersonnelid")

.HasColumnType("integer");

b.HasKey("receptionsid", "requiredPersonnelid");

b.HasIndex("requiredPersonnelid");

b.ToTable("JobTitleReception");

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("VacationId")

.HasColumnType("integer");

b.Property<DateOnly>("date")

.HasColumnType("date");

b.HasKey("id");

b.HasIndex("VacationId");

b.ToTable("DayOff");

});

modelBuilder.Entity("Models.Department", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Departments");

});

modelBuilder.Entity("Models.Holiday", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly>("date")

.HasColumnType("date");

b.Property<string>("name")

.IsRequired()

.HasColumnType("text");

b.HasKey("id");

b.ToTable("Holidays");

});

modelBuilder.Entity("Models.JobTitle", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<string>("name")

.HasColumnType("text");

b.HasKey("id");

b.ToTable("JobTitles");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<DateOnly?>("date")

.HasColumnType("date");

b.Property<int>("departmentid")

.HasColumnType("integer");

b.Property<TimeOnly?>("startTime")

.HasColumnType("time without time zone");

b.Property<TimeSpan>("time")

.HasColumnType("interval");

b.HasKey("id");

b.HasIndex("departmentid");

b.ToTable("Receptions");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("Vacation");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("DepartmentId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("DepartmentId");

b.ToTable("WorkSchedule4Day");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int?>("JobTitleId")

.HasColumnType("integer");

b.Property<string>("login")

.HasColumnType("text");

b.Property<string>("name")

.HasColumnType("text");

b.Property<string>("password")

.HasColumnType("text");

b.HasKey("id");

b.HasIndex("JobTitleId");

b.ToTable("Workers");

});

modelBuilder.Entity("Models.WorkerWorkSchedule4Day", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("integer");

NpgsqlPropertyBuilderExtensions.UseIdentityByDefaultColumn(b.Property<int>("id"));

b.Property<int>("WorkerId")

.HasColumnType("integer");

b.Property<TimeOnly>("endOfWork")

.HasColumnType("time without time zone");

b.Property<bool>("isWorking")

.HasColumnType("boolean");

b.Property<TimeOnly>("startOfWork")

.HasColumnType("time without time zone");

b.HasKey("id");

b.HasIndex("WorkerId");

b.ToTable("WorkerWorkSchedule4Day");

});

modelBuilder.Entity("ReceptionWorker", b =>

{

b.Property<int>("personnelid")

.HasColumnType("integer");

b.Property<int>("receptionsid")

.HasColumnType("integer");

b.HasKey("personnelid", "receptionsid");

b.HasIndex("receptionsid");

b.ToTable("ReceptionWorker");

});

modelBuilder.Entity("DepartmentWorker", b =>

{

b.HasOne("Models.Department", null)

.WithMany()

.HasForeignKey("departmentsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("workersid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("JobTitleReception", b =>

{

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.JobTitle", null)

.WithMany()

.HasForeignKey("requiredPersonnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.DayOff", b =>

{

b.HasOne("Models.Vacation", "Vacation")

.WithMany("days")

.HasForeignKey("VacationId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Vacation");

});

modelBuilder.Entity("Models.Reception", b =>

{

b.HasOne("Models.Department", "department")

.WithMany()

.HasForeignKey("departmentid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("department");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("vacations")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("Models.WorkSchedule4Day", b =>

{

b.HasOne("Models.Department", "Department")

.WithMany("workSchedules")

.HasForeignKey("DepartmentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Department");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.HasOne("Models.JobTitle", "jobTitle")

.WithMany()

.HasForeignKey("JobTitleId")

.OnDelete(DeleteBehavior.Cascade);

b.Navigation("jobTitle");

});

modelBuilder.Entity("Models.WorkerWorkSchedule4Day", b =>

{

b.HasOne("Models.Worker", "Worker")

.WithMany("workSchedules")

.HasForeignKey("WorkerId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Worker");

});

modelBuilder.Entity("ReceptionWorker", b =>

{

b.HasOne("Models.Worker", null)

.WithMany()

.HasForeignKey("personnelid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Models.Reception", null)

.WithMany()

.HasForeignKey("receptionsid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Models.Department", b =>

{

b.Navigation("workSchedules");

});

modelBuilder.Entity("Models.Vacation", b =>

{

b.Navigation("days");

});

modelBuilder.Entity("Models.Worker", b =>

{

b.Navigation("vacations");

b.Navigation("workSchedules");

});

#pragma warning restore 612, 618

}

}

}

// ==== D:\piaps\Schedule\Repositories\obj\Debug\net8.0\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\Schedule\Repositories\obj\Debug\net8.0\Repositories.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("Repositories")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+504c82c7efe4e511801a3b9066c8d233e0a42f0d")]

[assembly: System.Reflection.AssemblyProductAttribute("Repositories")]

[assembly: System.Reflection.AssemblyTitleAttribute("Repositories")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\Schedule\Repositories\obj\Debug\net8.0\Repositories.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

// ==== D:\piaps\Schedule\Repositories\obj\Release\net8.0\.NETCoreApp,Version=v8.0.AssemblyAttributes.cs ====

// <autogenerated />

using System;

using System.Reflection;

[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0", FrameworkDisplayName = ".NET 8.0")]

// ==== D:\piaps\Schedule\Repositories\obj\Release\net8.0\Repositories.AssemblyInfo.cs ====

//------------------------------------------------------------------------------

// <auto-generated>

// Этот код создан программой.

// Исполняемая версия:4.0.30319.42000

//

// Изменения в этом файле могут привести к неправильной работе и будут потеряны в случае

// повторной генерации кода.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("Repositories")]

[assembly: System.Reflection.AssemblyConfigurationAttribute("Release")]

[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]

[assembly: System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+5d726bf7f71429a9ab662739f3c7c68769eb4d34")]

[assembly: System.Reflection.AssemblyProductAttribute("Repositories")]

[assembly: System.Reflection.AssemblyTitleAttribute("Repositories")]

[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Создано классом WriteCodeFragment MSBuild.

// ==== D:\piaps\Schedule\Repositories\obj\Release\net8.0\Repositories.GlobalUsings.g.cs ====

// <auto-generated/>

global using global::System;

global using global::System.Collections.Generic;

global using global::System.IO;

global using global::System.Linq;

global using global::System.Net.Http;

global using global::System.Threading;

global using global::System.Threading.Tasks;

**AndroidApp**

// ==== D:\piaps\mobile\_schedule\lib\main.dart ====

import 'package:flutter/material.dart';

import 'package:mobile\_schedule/screens/login\_screen.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Schedule App',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: LoginScreen(),

);

}

}

// ==== D:\piaps\mobile\_schedule\lib\models\auth\_model.dart ====

class LoginRequest {

String login;

String password;

LoginRequest({required this.login, required this.password});

Map<String, dynamic> toJson() {

return {

'Login': login,

'Password': password,

};

}

}

class LoginResponse {

String sid;

LoginResponse({required this.sid});

factory LoginResponse.fromJson(Map<String, dynamic> json) {

return LoginResponse(

sid: json['sid'],

);

}

}

// ==== D:\piaps\mobile\_schedule\lib\models\reception\_model.dart ====

class Reception {

int id;

String date;

String startTime;

String endTime;

String departmentName;

String time;

Reception({

required this.id,

required this.date,

required this.startTime,

required this.endTime,

required this.departmentName,

required this.time,

});

factory Reception.fromJson(Map<String, dynamic> json) {

return Reception(

id: json['id'],

date: json['date'],

startTime: json['startTime'],

endTime: json['endTime'],

departmentName: json['department']['name'],

time: json['time'],

);

}

}

// ==== D:\piaps\mobile\_schedule\lib\screens\login\_screen.dart ====

import 'package:flutter/material.dart';

import 'package:mobile\_schedule/screens/reception\_screen.dart';

import '../services/api\_service.dart';

class LoginScreen extends StatefulWidget {

@override

\_LoginScreenState createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

final \_loginController = TextEditingController();

final \_passwordController = TextEditingController();

final \_apiService = ApiService();

String? \_errorMessage;

void \_login() async {

try {

final loginResponse = await \_apiService.login(

\_loginController.text,

\_passwordController.text,

);

// Переход на экран расписания

Navigator.push(

context,

MaterialPageRoute(

builder: (context) => ReceptionScreen(sid: loginResponse.sid, login: \_loginController.text),

),

);

} catch (e) {

setState(() {

\_errorMessage = e.toString();

});

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Login')),

body: Padding(

padding: EdgeInsets.all(16.0),

child: Column(

children: <Widget>[

TextField(

controller: \_loginController,

decoration: InputDecoration(labelText: 'Login'),

),

TextField(

controller: \_passwordController,

decoration: InputDecoration(labelText: 'Password'),

obscureText: true,

),

SizedBox(height: 20),

ElevatedButton(

onPressed: \_login,

child: Text('Login'),

),

if (\_errorMessage != null) ...[

SizedBox(height: 20),

Text(\_errorMessage!, style: TextStyle(color: Colors.red)),

]

],

),

),

);

}

}

// ==== D:\piaps\mobile\_schedule\lib\screens\reception\_screen.dart ====

import 'package:flutter/material.dart';

import '../models/reception\_model.dart';

import '../services/api\_service.dart';

class ReceptionScreen extends StatefulWidget {

final String sid;

final String login;

ReceptionScreen({required this.sid, required this.login});

@override

\_ReceptionScreenState createState() => \_ReceptionScreenState();

}

class \_ReceptionScreenState extends State<ReceptionScreen> {

final \_apiService = ApiService();

List<Reception> \_receptions = [];

@override

void initState() {

super.initState();

\_loadReceptions();

}

void \_loadReceptions() async {

try {

final receptions = await \_apiService.getReceptions(widget.sid, widget.login);

setState(() {

\_receptions = receptions;

});

} catch (e) {

ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text('Failed to load receptions')));

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Schedule')),

body: ListView.builder(

itemCount: \_receptions.length,

itemBuilder: (context, index) {

final reception = \_receptions[index];

return ListTile(

title: Text(reception.date),

subtitle: Text('Start: ${reception.startTime} - End: ${reception.endTime}'),

trailing: Text(reception.departmentName),

);

},

),

);

}

}

// ==== D:\piaps\mobile\_schedule\lib\services\api\_service.dart ====

import 'dart:convert';

import 'package:http/http.dart' as http;

import '../models/auth\_model.dart';

import '../models/reception\_model.dart';

class ApiService {

static const String baseUrl = 'http://10.0.2.2:5270/api/';

Future<LoginResponse> login(String login, String password) async {

final url = Uri.parse('${baseUrl}auth/login');

final response = await http.post(

url,

headers: {'Content-Type': 'application/json'},

body: json.encode(LoginRequest(login: login, password: password).toJson()),

);

if (response.statusCode == 200) {

return LoginResponse.fromJson(json.decode(response.body));

} else {

throw Exception('Failed to login');

}

}

Future<List<Reception>> getReceptions(String sid, String login) async {

final url = Uri.parse('${baseUrl}reception/getByLogin/$login');

final response = await http.get(

url,

headers: {'sid': sid},

);

if (response.statusCode == 200) {

var data = json.decode(response.body) as List;

return data.map((reception) => Reception.fromJson(reception)).toList();

} else {

throw Exception('Failed to load receptions');

}

}

}

// ==== D:\piaps\mobile\_schedule\test\widget\_test.dart ====

// This is a basic Flutter widget test.

//

// To perform an interaction with a widget in your test, use the WidgetTester

// utility in the flutter\_test package. For example, you can send tap and scroll

// gestures. You can also use WidgetTester to find child widgets in the widget

// tree, read text, and verify that the values of widget properties are correct.

import 'package:flutter/material.dart';

import 'package:flutter\_test/flutter\_test.dart';

import 'package:mobile\_schedule/main.dart';

void main() {

testWidgets('Counter increments smoke test', (WidgetTester tester) async {

// Build our app and trigger a frame.

await tester.pumpWidget(const MyApp());

// Verify that our counter starts at 0.

expect(find.text('0'), findsOneWidget);

expect(find.text('1'), findsNothing);

// Tap the '+' icon and trigger a frame.

await tester.tap(find.byIcon(Icons.add));

await tester.pump();

// Verify that our counter has incremented.

expect(find.text('0'), findsNothing);

expect(find.text('1'), findsOneWidget);

});

}

**tgBot**

import asyncio

import uvicorn

from tgBot import bot, dp

from webhook import app as fastapi\_app

from fastapi import FastAPI

async def start\_fastapi():

config = uvicorn.Config(fastapi\_app, host="0.0.0.0", port=8000, log\_level="info")

server = uvicorn.Server(config)

await server.serve()

async def start\_bot():

await dp.start\_polling(bot)

async def main():

await asyncio.gather(start\_fastapi(), start\_bot())

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

asyncio.run(main())

from aiogram import Bot, Dispatcher, Router, types, F

from aiogram.fsm.context import FSMContext

from aiogram.fsm.state import State, StatesGroup

from aiogram.fsm.storage.memory import MemoryStorage

from aiogram.types import Message

import requests

API\_TOKEN = "7874153058:AAHc-c6NcpAFZ8SIv4sEX3jPLcAZftMkqcI"

bot = Bot(token=API\_TOKEN)

dp = Dispatcher(storage=MemoryStorage())

router = Router()

dp.include\_router(router)

# Telegram ID ↔ логин + SID

user\_sessions = {}

# Состояния для FSM авторизации

class AuthState(StatesGroup):

login = State()

password = State()

# Команда /start

@router.message(F.text == "/start")

async def cmd\_start(message: Message, state: FSMContext):

await state.set\_state(AuthState.login)

await message.answer("Введите логин:")

# Ввод логина

@router.message(AuthState.login)

async def enter\_login(message: Message, state: FSMContext):

await state.update\_data(login=message.text)

await state.set\_state(AuthState.password)

await message.answer("Введите пароль:")

# Ввод пароля и авторизация

@router.message(AuthState.password)

async def enter\_password(message: Message, state: FSMContext):

data = await state.get\_data()

login, password = data["login"], message.text

try:

response = requests.post("http://localhost:5270/api/auth/login", json={

"login": login,

"password": password

})

if response.status\_code == 200:

sid = response.json()["sid"]

user\_sessions[message.from\_user.id] = {"login": login, "sid": sid}

await message.answer("✅ Авторизация прошла успешно")

else:

await message.answer("❌ Неверный логин или пароль")

except requests.exceptions.RequestException as e:

await message.answer(f"Ошибка соединения с API: {e}")

await state.clear()

from fastapi import FastAPI, Request

from tgBot import bot, user\_sessions

app = FastAPI()

@app.post("/notify")

async def notify(request: Request):

data = await request.json()

login = data.get("login")

text = data.get("text")

for tg\_id, info in user\_sessions.items():

if info["login"] == login:

await bot.send\_message(chat\_id=tg\_id, text=text)

return {"status": "sent"}

return {"status": "user not found"}