

ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ГОРОДА МОСКВЫ  
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ  
«МОСКОВСКИЙ КОЛЛЕДЖ УПРАВЛЕНИЯ, ГОСТИНИЧНОГО БИЗНЕСА И  
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ «ЦАРИЦЫНО»

**ДИПЛОМНЫЙ ПРОЕКТ**

Тема: Разработка информационной системы для автоматизации управления персоналом на предприятиях с почасовой формой оплаты труда.

Пояснительная записка

ДП.09.02.07.41.16.2022.01 ПЗ

Руководитель проекта: Воронина С.Ю.    Дипломник: Мозырский А.Д.

Рецензент: Адоньев М.С.

**Допущен к защите на заседании**

**Государственной экзаменационной комиссии**

Учебно-методический отдел отделения УИТ

«01» июня 2022 г.

 Т.Н. Михайлова

Москва, 2022

**Согласовано**

**Утверждаю**

На заседании Кафедры  
информационных технологий

**УМО ОУИТ**

Протокол № 11

« 31 » 03 2022г.

от «30» марта 2022 г.

Заведующий кафедрой

 Т.Н. Михайлова

 Е.Ф. Писчасова

### **ЗАДАНИЕ НА ДИПЛОМНОЕ ПРОЕКТИРОВАНИЕ**

Студенту Мозырскому Андрею Дмитриевичу  
группы ИС4-1 специальности 09.02.07 Информационные системы и  
программирование

Дата выдачи задания: «19» апреля 2022 г.

Срок сдачи проекта: «27» мая 2022 г.

**Тема дипломного проекта:**

Разработка информационной системы для автоматизации управления  
персоналом на предприятиях с почасовой формой оплаты труда.

#### **Техническое задание**

1. **Исходные данные** информация по персонале
2. **Задачи:**
  - Разработать программный модуль для распределение плановых рабочих часов на сотрудников по отделам
  - Разработать программный модуль для формирования дополнительных соглашений.
  - Разработать алгоритм регистрации и авторизации пользователей;
  - Разработать алгоритм приема и увольнения сотрудников;
  - Разработать алгоритм расчета заработной платы сотрудников в соответствии с отработанными часами;
  - Провести тестирование информационной системы;

| Раздел пояснительной записки                 | Объем, % |
|--|----------|
| Введение                                     | 6        |
| 1. Теоретическая часть                       | 30       |
| 1.1. Исследование предметной области         |          |
| 1.2. Выбор средств разработки ПО             |          |
| 2. Практическая часть                        | 62       |
| 2.1. Разработка базы данных                  |          |
| 2.2. Алгоритм работы Web Api                 |          |
| 2.3. Разработка пользовательского интерфейса |          |
| 2.4. Тестирование и отладка                  |          |
| Заключение                                   | 1        |
| Список используемой литературы               | 1        |
| Приложение 1                                 |          |


#### Графическая часть:

Мультимедиа презентация.


Дипломник:

 /Мозырский А.Д./

Руководитель проекта:

 /Воронина С.Ю./

Заведующий кафедрой:

 /Писчасова Е.Ф./

## Содержание

|   |            |
|---|------------|
| <b>Введение .....</b>   | <b>5</b>   |
| <b>Глава 1 Теоретическая часть.....</b>                             | <b>8</b>   |
| 1.1 Исследование предметной области.....                            | 8          |
| 1.2 Выбор СУБД.....   | 9          |
| 1.3 Выбор IDE.....  | 13         |
| 1.4 Выбор платформы.....  | 17         |
| 1.5 Выбор case-средств .....  | 19         |
| <b>Глава 2 Практическая часть .....</b>                             | <b>22</b>  |
| 1.1 Разработка базы данных .....                                    | 22         |
| 1.2 Структура информационной системы .....                          | 38         |
| 1.3 Разработка программных модулей.....                             | 43         |
| 2.4 Разработка пользовательского интерфейса.....                    | 46         |
| 2.5 Тестирование и отладка.....                                     | 50         |
| <b>Заключение.....</b>  | <b>54</b>  |
| <b>Список используемой литературы .....</b>                         | <b>56</b>  |
| <b>Приложение 1. Код SQL-инструкций для создания базы данных ..</b> | <b>58</b>  |
| <b>Приложение 2. Код уровня Domain .....</b>                        | <b>70</b>  |
| <b>Приложение 3. Код уровня Application.....</b>                    | <b>77</b>  |
| <b>Приложение 4. Код уровня Persistence.....</b>                    | <b>107</b> |
| <b>Приложение 5. Код уровня Presentation (Web Api) .....</b>        | <b>131</b> |
| <b>Приложение 6. Код уровня Presentation (Клиент) .....</b>         | <b>143</b> |
| <b>Приложение 7. Код тестирования алгоритмов .....</b>              | <b>191</b> |

|           |                |          |         |      |  |                          |      |        |
|-----------|----------------|----------|---------|------|--|--------------------------|------|--------|
|           |                |          |         |      | ДП.09.02.07.41.16.2022.01 ПЗ   |                          |      |        |
|           |                | № докум. | Подпись | Дата |  |                          |      |        |
| Разраб.   | Мозырский А.Д. |          |         |      | Разработка информационной системы для автоматизации управления персоналом на предприятиях с почасовой формой оплаты труда<br><br>Пояснительная записка | Лит.                     | Лист | Листов |
| Пров.     | Воронина С.Ю.  |          |         |      |  | Д                        | 4    | 203    |
| Реценз.   | Адоньев М.С.   |          |         |      |  | ГБПОУ Колледж «Царицыно» |      |        |
| Н. контр. | Воронина С.Ю.  |          |         |      |  |                          |      |        |
| Утв.      | Писчасова Е.Ф. |          |         |      |  |                          |      |        |

## **Введение**

Рассматривая структуру любого предприятия с почасовой формой оплаты труда, можно выделить распространенные причины убытков: отставания от планов и графиков, постоянную текучку персонала, необходимого для обеспечения работоспособности компании и излишнее количество человеко-часов, уделяемых на составление однообразной документации, а именно, дополнительных соглашений с сотрудниками на каждый месяц.

Поскольку прибыль компании напрямую зависит от отработанных сотрудниками человеко-часов, полнота и правильность распределения рабочих часов между сотрудниками являются ключевым фактором дохода компании.

Объектом исследования в данном дипломном проекте является процесс управления персоналом на предприятиях. Предметом является универсальная информационная система, позволяющая автоматизировать управление персоналом.

В настоящее время существует множество приложений, распространяемых как свободно, так и на коммерческой основе, позволяющих оптимизировать управление рабочей нагрузкой на работников компании. Среди них можно выделить: «1С-CRM», «Peopleforce», «Talantix» и т.д. Использование этих приложений возможно, однако, во-первых, они не способны оптимизировать процесс формирования дополнительных соглашений, а во-вторых, имеют ряд существенных ограничений.

При использовании перечисленных выше систем процесс тайм-менеджмента персонала лишь немного оптимизирован, но не автоматизирован полностью, оперативное изменение распределения рабочих часов невозможно. Также стоит отметить, что при создании базы данных, приведенные системы запрашивают очень большой объем информации,

которая, как правило, при работе с информационной системой затем не используется.

Учитывая все недостатки, складывается необходимость разработки информационной системы для автоматизации управления персоналом на предприятиях с почасовой формой оплаты труда, которая при минимальном наборе данных будет адекватно распределять рабочие часы между отделами и сотрудниками компании, с возможностью внесения изменений в нестандартных ситуациях. Поэтому целью дипломного проекта является проектирование базы данных для универсальной платформы управления персоналом предприятия и разработка масштабируемого WebAPI, реализующего базовый функционал HRM (Human Resource Management), а также обеспечивающего распределение рабочей нагрузки и формирование дополнительных соглашений.

Качество работы универсальной информационной системы определяется возможностью эффективного распределения рабочей нагрузки между отделами и персоналом организации, автоматическими расчетами заработной платы сотрудников, оперативным реагированием на изменение стандартов рабочего времени.

На данный момент времени в большинстве таких компаний руководящий персонал и бухгалтера вручную ведут учет персонала, структуры отделов и их рабочей нагрузки. Решают, сколько часов нужно отработать компании за определенный период времени для получения прибыли, сколько должен будет отработать каждый отдел и сотрудник, формируют дополнительные соглашения и рассчитывают заработную плату. Сотрудники вручную ведут учет истории своих дополнительных соглашений.

Исходя из цели, в рамках дипломного проекта необходимо реализовать следующие задачи:

1. Исследование методов распределения рабочей нагрузки.
2. Разработать универсальную базу данных управления персоналом предприятия.

3. Разработать алгоритмы работы WebAPI.
4. Реализовать взаимодействие WebAPI с базой данных.
5. Разработать пользовательский интерфейс клиентского приложения.
6. Провести комплексное тестирование основных функций WebAPI.

## **Глава 1 Теоретическая часть**

### **1.1 Исследование предметной области**

При рассмотрении объекта исследования и статистики различных организаций был сделан вывод о том, что, несмотря на специфику различных предприятий, руководители компаний стремятся усреднить рабочую нагрузку на персонал.

В соответствии с трудовым кодексом Российской Федерации (ТК РФ Статья 91) рабочая нагрузка не должна превышать 40 часов в неделю, однако в зависимости от должностных обязанностей, диапазон рабочей нагрузки варьируется от 28 до 200 часов в месяц. По данным портала OECD.Stat рабочая нагрузка среди наёмных работников составляет 120 ч.  $\pm$  5 ч. (в месяц) что составляет один полный рабочий день по графику 1 через 1.

Вследствие этого было решено, что модуль распределения рабочей нагрузки разрабатываемой системы будет иметь следующие ограничения:

1. Минимальная рабочая нагрузка не должна быть меньше, чем 4 ч. в день \* (кол-во дней в текущем месяце / 3). Таким образом достигается порог в одну 4х часовую смену по графику 1 через 2.

2. Максимальная рабочая нагрузка не должна превышать 9 ч. в день \* кол-во рабочих дней в текущем месяце. Данное ограничение берется из расчета возможной необходимости введения сверхурочных рабочих часов.

Распределение предполагает, что все сотрудники конкретного отдела получают равноценную рабочую нагрузку. Различные отделы могут иметь различные требования к персоналу, поэтому для каждого отдела можно задать свои стандарты рабочего времени.



## 1.2 Выбор СУБД

### Описание современных СУБД

СУБД представляет собой совокупность специальных языковых и программных средств, облегчающих пользователям выполнение всех операций, связанных с организацией хранения данных, их корректировкой и доступом к ним.

Современные реляционные СУБД обеспечивают:

- 1) Набор средств для поддержки таблиц и соотношений между связанными таблицами.
- 2) Развитый пользовательский интерфейс, позволяющий вводить и модифицировать информацию, выполнять поиск.
- 3) Средства программирования для разработки собственных приложений.

Для выполнения данного проекта рассматриваются несколько СУБД:

### MS SQL

Довольно популярная СУБД, которая является программным средством, разработанным компанией Microsoft. Данная СУБД, располагается на как на облачных, так и на локальных серверах, причем возможно комбинировать применяемые сервера одновременно. Еще программный продукт Microsoft, имеет лицензированную поддержку даже для бесплатной версии, большое сообщество, которое может быть полезным в решении возникших трудностей и задач, а также большую базу документации, также переведенной на русский язык.

Актуальная версия Microsoft SQL сервер имеет возможность поддержки ddm (динамическую маскировку данных), которая гарантирует, что только авторизованные пользователи с установленными правами будут видеть

данные. MS SQL Server имеет умный мастер импорта, также можно сформировать скрипт базы данных. В процессе изучения СУБД были выявлены следующие преимущества:

- 1) СУБД довольно проста.
- 2) Актуальная версия достаточно быстра и стабильна.
- 3) Доступна в бесплатном варианте для физических лиц.

И недостатки:

- 1) Высокая цена за платную версию.
- 2) Занимает все доступные аппаратные ресурсы.

Идеально подходит для крупных организаций, которые уже используют ряд продуктов Microsoft.

## PostgreSQL

PostgreSQL это бесплатная и довольно популярная СУБД, которая зачастую используется для создания и ведения баз данных web-сайтов. Это одна из первых СУБД, и поэтому сейчас ее функциональность очень хорошо развита, что позволяет пользователям управлять как структурированными, так и неструктурированными данными. Отлично показывает себя с задачами по импорту данных из других типов баз с помощью собственных возможностей.

Наиболее актуальная версия PostgreSQL дает возможность обработки огромных объемов данных и значительное увеличение одновременно задействованных пользователей. В процессе изучения СУБД были выявлены следующие преимущества:

- 1) Имеет возможность к масштабированию и способен обрабатывать большое количество данных.
- 2) Поддерживает формат json, что расширяет возможности.
- 3) Имеет множество predefined возможностей и функций.
- 4) Доступен ряд разнообразных интерфейсов.

И недостатки:

- 1) Мало официальной документации.
- 2) Скорость работы может “прыгать” время от времени.
- 3) Сложный в освоении интерфейс.

Очень хорошо подходит для пользователей с малым бюджетом. Однако необходим специалист высокого уровня, для возможности выбрать интерфейс и использовать json, без потери в скорости.

## MySQL

MySQL - наверное, наиболее популярная СУБД с богатой, бесплатной функциональностью. Хотя это и бесплатная СУБД, но обновления к ней приходят постоянно, расширяя ее возможности и модернизируя систему безопасности. В ней так же есть разнообразные платные версии, предназначенные для коммерческого пользования. Бесплатная версия программы производит основной упор в надежность СУБД и ее скорость, а не на вариативность функциональных возможностей.

СУБД MySQL обеспечивается поддержкой большого количества разнообразных типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Кроме того, MySQL предоставляется с особыми типами таблиц EXAMPLE. СУБД также имеет несложный интерфейс, обычные функции которого не требуют специфичной подготовки. Система надежна и не стремится использовать под себя все доступные ресурсы. В процессе изучения СУБД были выявлены следующие преимущества:

- 1) Бесплатное распространение.
- 2) Хорошо документирована на разных языках.
- 3) Различное множество функций.
- 4) Имеет возможность поддержки разнообразных пользовательских интерфейсов.

И недостатки:

- 1) Нет предустановленных возможностей решения простых задач.
- 2) В бесплатной версии отсутствует поддержка.

Подходит для тех, кому необходим бесплатный, но надежный инструмент управления базами данных.

### Обоснование выбранной СУБД

Исходя из целей данной работы, необходимо реализовать хранение часто меняющегося списка сотрудников компании, для обеспечения целостности данных, в соответствующей таблице должен быть предусмотрен атрибут id, содержащий в себе уникальное целочисленное значение. Для автоматизации заполнения данного атрибута в MS SQL предусмотрено свойство IDENTITY, которое по умолчанию запрещает пользователю изменять значение атрибута. Похожая функция есть и в других СУБД. В PostgreSQL для этого используется отдельный тип данных serial, что накладывает множество ограничений. В MySQL существует свойство AUTO\_INCREMENT, который скорее выполняет роль функции, подставляющей значение на основе уже заданных в данном столбце, то есть не ограничивает пользовательский ввод атрибута и сбрасывает структуру нумерации во время удаления строк. Эта особенность может негативно повлиять на качество базы данных.

При работе HR, нельзя исключать вероятность того, что при подаче резюме для работы, кандидат может случайно отправить его дважды или заполнить неполностью, рассмотрим для примера обработку ввода данных в атрибут email. Для обеспечения уникальности введенных значений язык SQL содержит ограничение UNIQUE, который по-разному работает в разных СУБД. При использовании MS SQL, данное ограничение гарантирует не только уникальность данных, но и то, что несколько строк не будут содержать NULL значения, в отличие от MySQL, в котором это допускается.

Исходя из данных особенностей работы разных СУБД, для реализации данного проекта самым подходящим вариантом является MS SQL. Так как разрабатываемое ПО подлежит внедрению в уже существующие компании, необходим быстрый и качественный импорт информации, которая ранее хранилась вне базы данных, в частности из файлов с расширением .xlsx и .csv. MS SQL предоставляет возможность импортировать и экспортировать данные из любых источников с помощью встроенного программного решения.

### **1.3 Выбор IDE**

Описание современных сред разработки

Среда разработки представляет из себя соединённые в одно решение программные средства, целью которой является создание и отладка программного продукта.

Среда разработки объединяет в себе:

- 1) Компилятор – программа, предназначенная для считывания исходного кода, который она преобразует в программный продукт.
- 2) Интерпретатор – программа, задачей которой является считывание команд, скрытых в исходном коде, и их последующее выполнение.
- 3) Отладчик – программа, ищущая ошибки в разрабатываемых программный продуктах, SQL-запросах и других разновидностях кода.
- 4) Среда автоматизации сборки – этап в процессе разработки ПП, который представляет из себя автоматизацию большого спектра задач, решаемых разработчиками.
- 5) Редактор текста – программа, функциональными возможностями которой являются создание и редактирование текстовых данных при разработке ПП, а также создания обычных текстовых файлов.

Если в среде разработки реализованы все вышеперечисленные компоненты, то тогда ее можно назвать интегрированной. Интегрированная среда разработки позволяет разработчику выбрать подходящий для данной задачи язык программирования (из языков, поддерживаемых данной средой).

## Visual Studio

Visual Studio — это IDE, позволяющая: отлаживать, редактировать и создавать код. Кроме стандартного отладчика и редактора, которые имеют место в большинстве IDE, Visual Studio имеет в своем арсенале средства выполнения кода, компиляторы, графические конструкторы и большое количество других функций для повышения комфорта разработки. В процессе изучения интегрированных средств разработки были выявлены следующие преимущества:

1) Учитывая то, что IDE создана в Microsoft очевидно, что она хорошо взаимодействует с другими продуктами данной компании. Кроме того, в определенных ситуациях без Visual Studio не обойтись — например, при реализации проекта WPF.

2) Функционала «Community edition» (бесплатная версия) для обычного пользователя будет достаточно.

3) Visual Studio имеет огромное количество разнообразных плагинов. Используя их, можно расширить функциональные возможности, а также подключать другие языки программирования.

4) Имеет поддержку платформы .NET. Данная среда разработки обладает широкими возможностями по созданию приложений для Windows, в том числе в .NET-сегменте.

И недостатки:

1) При долгом использовании могут появляться баги, не срабатывать нажатия, нарушаться работа с сервером.

2) Самоличное освоение Visual Studio новичком является довольно непростой задачей — большое количество функциональных возможностей и скрытых настроек.

Подходит для опытных пользователей, а также для тех, кому нужен огромный функционал и постоянная поддержка компании разработчика.

### JetBrains Rider

JetBrains Rider — кроссплатформенная интегрированная среда разработки программных продуктов для платформы .NET. Поддерживает разнообразные языки программирования, среди которых: C#, VB.NET и F#. Данная среда разработки поддерживает .NET Framework, новые кроссплатформенные .NET Core и моно-проекты. Это позволяет создавать широкий спектр приложений, включая: службы и библиотеки, игры Unity, приложения Xamarin, ASP.NET. В процессе изучения интегрированных средств разработки были выявлены следующие преимущества:

1) Особенность продуктов JetBrains, воссозданная в Project Rider. С ним вы сможете организовать весь цикл создания ПП: от идеи до поддержки.

2) Данная среда разработки дает возможность подключить MSBuild и XBuild, работать с CLI-проектами и организовать отладку приложений .NET и Mono.

3) Возможность поддержания нескольких запущенных программ.

4) Project Rider работает с Windows, Linux и MacOS.

И недостатки:

1) Так как проект новый, то часть обещанного функционала еще в разработке, а также не все стартовые баги исправлены.

2) Project Rider в самой дешевой сборке обойдется в довольно большую сумму за первый год использования. Хотя для данной среды разработки и есть триал-версия, но она обладает сильно урезанной функциональностью.

3) Может использоваться как опытными пользователями, так и новичками, однако слабо подходит для тех, кто прежде всего ценит бесперебойность работы.

## Code::Blocks

Code::Blocks — представляет из себя кроссплатформенную среду разработки, которая существует между мощными средами по созданию больших проектов, типа Visual Studio, и слабыми по функционалу, но удобными блокнотами типа Sublime, причем преимущества и тех, и других сочетаются и позволяют использовать данную среду, как для создания небольших ПП для встраиваемых приложений, так и для программирования простых приложений для РС. В процессе изучения интегрированных средств разработки были выявлены следующие преимущества:

- 1) Проект полностью бесплатный (open-source).
- 2) Среда Code::Blocks проста в освоении, необходимо лишь знать один из предлагаемых языков.

- 3) Данная IDE возможно запустить на любой десктопной ОС.

И недостатки:

- 1) Довольно слабая функциональность для IDE. Для создания комплексных приложений Code::Blocks практически не подходит.

- 2) Довольно часто происходят ошибки в работе всего проекта.

Данный продукт подойдет для разработки небольших приложений и простого написания кода, как и хорошо подходит для обучения новых пользователей. Однако бесполезен при разработке крупных проектов.

## Обоснование выбранной среды разработки

Для реализации данного проекта самым подходящим вариантом среды разработки является Visual Studio. Разрабатываемое приложение должно



выполнять взаимодействие с базой данных, для чего необходим функционал тонкой отладки и дебагинга, который может обеспечить только Visual Studio.

В процессе разработки Web Api должна использоваться система контроля версий, Visual Studio помогает интегрировать Git и отслеживать состояние проекта.

Так как Visual Studio является продуктом компании Microsoft, он содержит множество предустановленных библиотек для работы с такими платформами как MS SQL и ASP.NET CORE и шаблонов различных проектов, что существенно ускоряет процесс разработки.

Функциональные возможности бесплатной версии, являются наиболее оптимальными для решения поставленной задачи. Visual Studio обладает не только развернутой функциональностью для создания Web Api, но и упрощает взаимодействие с MS SQL.

## **1.4 Выбор платформы**

### **Описание современных платформ**

Платформа разработки программного решения, это совокупность языка программирования, фреймворка и множества вспомогательных технологий. В случае с веб приложениями, наибольший интерес представляют языки JavaScript, C# и Python, а так же фреймворки для них.

### **Django**

Django – это свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (англ. View), а презентационная логика

Представления реализуется в Django уровнем Шаблонов (англ. Template). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV). Проект поддерживается организацией Django Software Foundation.

В отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений.

Данный фреймворк имеет ряд преимуществ:

- 1) Диспетчер URL на основе регулярных выражений
- 2) Google App Engine Python работает с любой версией Django.
- 3) Встроенный интерфейс администратора

И недостатки:

- 1) Невозможность замены или редактирования методов ORM
- 2) Отклонение от паттерна MVC ведет к низкой эффективности работы приложения

## ASP .NET Core

ASP .NET Core – это свободно-распространяемый кроссплатформенный фреймворк для создания веб-приложений на платформе .NET, на языке C# с открытым исходным кодом. Данная платформа разрабатывается компанией Майкрософт совместно с сообществом. Имеет модульную структуру и совместима с такими операционными системами как Windows, Linux и macOS.

Несмотря на то, что это новый фреймворк, построенный на новом веб-стеке, он обладает высокой степенью совместимости концепций с ASP.NET. Приложения ASP.NET Core поддерживают параллельное управление версиями, при котором разные приложения, работающие на одном компьютере, могут ориентироваться на разные версии ASP.NET Core.

Данный фреймворк имеет следующие преимущества:

- 1) Высокая степень интеграции продуктов компании Microsoft
- 2) Регулярные обновления вспомогательных фреймворков и ресурсов

- 3) Качественная документация
- 4) Открытый исходный код
- 5) Универсальное применение целевого продукта
- 6) Шаблоны и WebAssembly

И недостатки:

- 1) Частые обновления платформы
- 2) Недостаточная оптимизация

### Обоснование выбранной платформы

Платформой для проектирования web API выбрана ASP .NET Core, поскольку, она позволяет создавать универсальные API с возможностью дальнейшего подключения к ним сторонних сервисов и клиентских приложений. Так же в данном проекте большая часть программных решений разработана компанией Microsoft, что упрощает интеграцию.

## 1.5 Выбор case-средств

### Описание современных case-средств

CASE-средство представляет собой набор специальных программных методов и технологий, которые помогают обеспечить отсутствие ошибок, высокое качество при проектировании и обслуживании программного продукта. Главные составляющие CASE-продукта таковы:

- 1) Методология, которая задает единый графический язык и правила работы с ним.
- 2) Графические редакторы, которые позволяют создавать диаграммы.
- 3) Генератор: по графическому представлению модели можно сгенерировать исходный код для различных платформ.

4) Репозиторий, своеобразная база данных для хранения результатов работы программистов.

draw.io

draw.io – это Интернет-ресурс, который обладает большой функциональностью, разнообразием видов диаграмм разнообразных факторов. Обладает удобным и гибким интерфейсом, а также возможностью его настройки под себя. Кроме того, разнообразие шаблонов позволяет экономить большое количество времени. В процессе изучения CASE-средства были выявлены следующие преимущества:

- 1) Богатый функциональный набор.
- 2) Бесплатная веб-версия.
- 3) Универсальная система.
- 4) Большое количество разнообразных шаблонов.

И недостатки:

- 1) Нет интеграции с СУБД.
- 2) Бесплатной является только веб-версия.

Подходит для тех, кому необходимо быстрое и удобное CASE-средство.

Toad Data Modeler

Toad Data Modeler – это средство позволяющие проектировать базы данных приложений. Понятный интерфейс позволяет даже неопытному пользователю успешно использовать данное CASE-средство. Кроме того, оно поддерживает разнообразные СУБД, что облегчает интегрирование модели. В процессе изучения CASE-средства были выявлены следующие преимущества:

- 1) Существует бесплатная учебная версия.
- 2) Поддержка СУБД.
- 3) Есть возможность написания SQL-скриптов.

4) Большой функционал возможностей в сфере применения.

И недостатки:

1) Платная полная версия.

2) Узкая сфера применения.

3) Документация только на английском.

Подходит для тех, кому нужен узконаправленный инструмент создания моделей баз данных.

### Microsoft Office Visio

Microsoft Office Visio — графическая среда, разработанная и поддерживаемая Microsoft. Имеет богатые функциональные возможности для реализации бизнес-планов, схем и документов. Так как данное средство является продуктом Microsoft, она хорошо совместима с остальными ее продуктами. В процессе изучения CASE-средства были выявлены следующие преимущества:

1) Большое количество документации.

2) Возможность создания диаграмм потоковых данных.

3) Возможности интеграции с продуктами Microsoft.

И недостатки:

1) Отсутствие удобства интерфейса.

2) Полный функционал является платным.

Отлично подходит для пользователей, которые уже используют ряд продуктов Microsoft.

### Обоснование выбранного case-средства

Минимальная необходимая функциональность CASE-средства ограничена графическим редактором для построения разнообразных диаграмм.

Учитывая поставленную в проекте задачу, в качестве основного CASE-средства, был выбран бесплатный интернет ресурс draw.io, так как он обладает необходимыми функциональными возможностями, удобным и простым интерфейсом, а также большим количеством импортируемых шаблонов.

## Глава 2 Практическая часть

### 1.1 Разработка базы данных

Исходя из предметной области в проекте были выделены следующие сущности:

Authorization, Employee, Interview, Candidate, Document, Dismissal, Department, Period, Department\_work\_load, Employee\_work\_load, Personal\_achivements, Passport\_info, Contact\_data.

Более подробная информация об описании и назначении сущностей представлена в таблице 1.

Описание множества сущностей.

Таблица 1.

| Номер сущности | Имя сущности  | Определение                                 | Описание  |
|----------------|---------------|---|---|
| E1             | Authorization | Аккаунт пользователя информационной системы | Новый экземпляр сущности появляется при регистрации сотрудника в информационной системе |
| E2             | Employee      | Человек, являющийся сотрудником компании    | Новый экземпляр сущности появляется в результате прохождения собеседования              |
| E3             | Interview     | Сервис найма или отказа                     | Новый экземпляр сущности появляется при   |

| Номер<br>сущности | Имя<br>сущности          | Определение   | Описание  |
|-------------------|--------------------------|---|---|
|                   |                          | потенциального<br>сотрудников   | прохождении кандидатом<br>собеседования   |
| E4                | Candidate                | Человек,<br>являющийся<br>потенциальным<br>сотрудником<br>компании            | Новый экземпляр<br>сущности появляется при<br>подаче кандидатом резюме<br>на работу в компании                          |
| E5                | Document                 | Электронный<br>документ,<br>содержащий не<br>обязательные<br>данные кандидата | Новый экземпляр<br>сущности появляется при<br>прикреплении кандидатом<br>к своему резюме<br>документа                   |
| E6                | Dismissal                | Сервис увольнения<br>сотрудников  | Новый экземпляр<br>сущности появляется при<br>увольнении сотрудника   |
| E7                | Department               | Структура<br>персонала компании   | Новый экземпляр<br>сущности появляется при<br>расширении<br>направленности компании                                     |
| E8                | Period                   | Временной<br>интервал,<br>длительностью 1<br>месяц                            | Новый экземпляр<br>сущности появляется при<br>назначении ответственным<br>лицом рабочей нагрузке на<br>следующий период |
| E9                | Department_<br>work_load | Электронный<br>документ,<br>содержащий данные                                 | Новый экземпляр<br>сущности появляется при<br>назначении ответственным  |

| Номер<br>сущности | Имя<br>сущности          | Определение  | Описание  |
|-------------------|--------------------------|--|---|
|                   |                          | о рабочей нагрузке<br>на отдел в<br>определенный<br>период времени   | лицом рабочей нагрузке на<br>следующий период   |
| E10               | Employee_<br>work_load   | Электронный<br>документ,<br>содержащий данные<br>о рабочей нагрузке<br>на сотрудника в<br>определенный<br>период времени | Новый экземпляр<br>сущности появляется при<br>назначении ответственным<br>лицом рабочей нагрузке на<br>следующий период |
| E11               | Personal_<br>achivements | Электронный<br>документ,<br>содержащий данные<br>о добровольной<br>сверх рабочей<br>нагрузке на<br>сотрудника            | Новый экземпляр<br>сущности появляется при<br>перевыполнении плана<br>работ сотрудником<br>компании                     |
| E12               | Passport_info            | Электронный<br>документ,<br>содержащий данные<br>о кандидате или<br>сотруднике   | Новый экземпляр<br>сущности появляется при<br>подаче кандидатом резюме  |
| E13               | Contact_data             | Электронный<br>документ,<br>содержащий данные<br>о способах связи с  | Новый экземпляр<br>сущности появляется при<br>подаче кандидатом резюме  |



| Номер<br>сущности | Имя<br>сущности | Определение                   | Описание |
|-------------------|-----------------|-------------------------------|----------|
|                   |                 | кандидатом или<br>сотрудником |          |

В качестве примера данных для импорта в базу данных приведены тестовые экземпляры для каждой сущности:

Authorization/E1: 0, NULL, admin, 12345, Administrator

Employee/E2: 1, 1, 1, 1, 1, true, 05f04074-a0d2-4abd-a039-72e3e8336f40

Interview/E3: 1, 1, True, 2020-12-09

Candidate/E4: 1, 1, 1, СПО, 0

Document/E5: 1, 1, Sertificate, localhost://document/plziwonthisjob.pdf

Dismissal/E6: 1, 1, 2021-11-17, 2021-12-17, По собственному желанию,  
1000

Department/E7: 1, 50, Тех. обслуживание, 250, 250

Period/E8: 1, 2021, 11, 10000

Department\_work\_load/E9: 1, 1, 1, 495, 495, True

Employee\_work\_load/E10: 1, 1, 1, 165, 165, 41250

Personal\_achivements/E11: 1, 1, 1, Выход на рабочее место в нерабочий  
день, 500

Passport\_info/E12: 1, 4819, 462281, Милана, Яскунова, Денисовна,  
Самарская область, Россия, г. Домодедово, Максима Горького ул., 3, 2, 73,

Contact\_data/E13: 1, milana.yaskunova@rambler.ru, 89984254851

Для более наглядного обозначения зависимостей сущностей была разработана матрица связей, представленная в таблице 2.

# Матрица связей.

Таблица 2.

|                          | Authorization/E1 | Employee/E2 | Interview/E3 | Candidate/E4 | Document/E5 | Dismissal/E6 | Department/E7 | Period/E8 | Department_wokload/E9 | Employee_wok_load/E10 | Personal_achivements/E11 | Pasport_info/E12 | Contact_data/E13 |
|--------------------------|------------------|-------------|--------------|--------------|-------------|--------------|---------------|-----------|-----------------------|-----------------------|--------------------------|------------------|------------------|
| Authorization/E1         |                  | R1          |              |              |             |              |               |           |                       |                       |                          |                  |                  |
| Employee/E2              | R1               |             | R2           |              |             |              | R6            |           |                       | R11                   | R12                      | R13              | R14              |
| Interview/E3             |                  | R2          |              | R3           |             |              |               |           |                       |                       |                          |                  |                  |
| Candidate/E4             |                  |             | R3           |              | R4          |              |               |           |                       |                       |                          | R15              | R16              |
| Document/E5              |                  |             |              | R4           |             |              |               |           |                       |                       |                          | R5               |                  |
| Dismissal/E6             |                  |             |              |              |             |              |               |           |                       |                       |                          |                  |                  |
| Department/E7            |                  | R6          |              |              |             |              |               |           | R10                   |                       |                          |                  |                  |
| Period/E8                |                  |             |              |              |             |              |               |           | R7                    | R8                    | R9                       |                  |                  |
| Department_work_load/E9  |                  |             |              |              |             |              | R10           | R7        |                       |                       |                          |                  |                  |
| Employee_work_load/E10   |                  | R11         |              |              |             |              |               | R8        |                       |                       |                          |                  |                  |
| Personal_achivements/E11 |                  | R12         |              |              |             |              |               | R9        |                       |                       |                          |                  |                  |
| Passport_info/E12        |                  | R13         |              | R15          |             | R5           |               |           |                       |                       |                          |                  |                  |
| Contact_data/E13         |                  | R14         |              | R16          |             |              |               |           |                       |                       |                          |                  |                  |

Далее, в таблице 3 представлено подробное описание связей сущностей и способы их реализации.

| Номер связи | Родительская сущность | Дочерняя сущность | Имя связи                 | Тип связи   | Мощность | Описание   |
|-------------|-----------------------|-------------------|---------------------------|---|----------|--|
| R1          | Employee/E2           | Authorization/E1  | FK_Authorization_Employee | Определенная<br>Не идентифицирующая<br>Необязательная | Z        | Один сотрудник может зарегистрироваться как пользователь или не делать этого, у каждого сотрудника может быть только 1 аккаунт |
| R2          | Interview/E3          | Employee/E2       | FK_Employee_Interview     | Определенная<br>Не идентифицирующая<br>Обязательная   | Z        | По результатам одного интервью может быть нанят 1 сотрудник или не нанят никто   |

| Номер связи | Родительская сущность | Дочерняя сущность | Имя связи                  | Тип связи   | Мощность | Описание  |
|-------------|-----------------------|-------------------|----------------------------|---|----------|---|
| R3          | Candidate/E4          | Interview/E3      | FK_Interview_Candidate     | Определенная<br>Не идентифицирующая<br>Обязательная | P        | Один кандидат может несколько раз приходить на интервью, при этом одному сотруднику всегда соответствует одно пройденное интервью |
| R4          | Candidate/E4          | Document/E5       | FK_Document_Candidate      | Определенная<br>Не идентифицирующая<br>Обязательная |          | Один кандидат может отправить различные документы, но может и не отправлять ничего  |
| R5          | Passport_info/E12     | Dismissal/E6      | FK_Dismissal_Passport_info | Определенная<br>Не идентифицирующая<br>Обязательная | Z        | Один сотрудник может быть уволен 1 раз, но может и не быть уволен   |

| Номер связи | Родительская сущность | Дочерняя сущность       | Имя связи                      | Тип связи  | Мощность | Описание  |
|-------------|-----------------------|-------------------------|--------------------------------|--|----------|---|
| R6          | Department/E7         | Employee/E2             | FK_Employee_Department         | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> |          | <p>Один отдел может содержать множество сотрудников, но сотруднику не обязательно состоять в каком-либо отделе</p>        |
| R7          | Period/E8             | Department_work_load/E9 | FK_Department_work_load_Period | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> | Р        | <p>Каждому периоду соответствует по одной нагрузке на каждый отдел, одной нагрузке на отдел соответствует один период</p> |

| Номер связи | Родительская сущность | Дочерняя сущность         | Имя связи                       | Тип связи  | Мощность | Описание  |
|-------------|-----------------------|---------------------------|---------------------------------|--|----------|---|
| R8          | Period/E8             | Employee_wok_load/E10     | FK_Employee_work_load_Period    | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> | P        | <p>Каждому периоду соответствует по одной нагрузке на каждого сотрудника, одной нагрузке на сотрудника соответствует один период</p>                        |
| R9          | Period/E8             | Personal_achievements/E11 | FK_Personal_achievements_Period | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> |          | <p>За каждый период может быть несколько достижений сотрудников, а может и не быть вообще, каждое достижение сотрудника привязано к конкретному периоду</p> |

| Номер связи | Родительская сущность | Дочерняя сущность       | Имя связи                          | Тип связи  | Мощность | Описание  |
|-------------|-----------------------|-------------------------|------------------------------------|--|----------|---|
| R10         | Department/E7         | Department_work_load/E9 | FK_Department_work_load_Department | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> | P        | <p>Один отдел получает множество нагрузок (по 1 за период), нагрузка направлена на конкретный отдел</p>           |
| R11         | Employee/E2           | Employee_work_load/E10  | FK_Employee_work_load_Employee     | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> | P        | <p>Один сотрудник получает множество нагрузок (по 1 за период), нагрузка направлена на конкретного сотрудника</p> |

| Номер связи | Родительская сущность | Дочерняя сущность         | Имя связи                         | Тип связи  | Мощность | Описание   |
|-------------|-----------------------|---------------------------|-----------------------------------|--|----------|--|
| R12         | Employee/E2           | Personal_achievements/E11 | FK_Personal_achievements_Employee | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> |          | <p>Каждый сотрудник</p> <p>может иметь несколько</p> <p>достижений, а может и</p> <p>не иметь вообще,</p> <p>каждое достижение</p> <p>привязано к</p> <p>конкретному</p> <p>сотруднику</p> |
| R13         | Passport_info/E12     | Employee/E2               | FK_Employee_Passport_info         | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> | Z        | <p>Каждому паспорту</p> <p>может соответствовать</p> <p>один сотрудник, а</p> <p>может 0</p>   |



| Номер связи | Родительская сущность | Дочерняя сущность | Имя связи                  | Тип связи  | Мощность | Описание   |
|-------------|-----------------------|-------------------|----------------------------|--|----------|--|
| R14         | Contact_data/E13      | Employee/E2       | FK_Employee_Contact_data   | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> | P        | <p>Каждому паспорту соответствует один кандидат, одному кандидату соответствует один паспорт</p>                                 |
| R15         | Passport_info/E12     | Candidate/E4      | FK_Candidate_Passport_info | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> | Z        | <p>Каждому набору контактных данных может соответствовать один сотрудник, а может 0</p>  |
| R16         | Contact_data/E13      | Candidate/E4      | FK_Candidate_Contact_data  | <p>Определенная</p> <p>Не идентифицирующая</p> <p>Обязательная</p> | P        | <p>Каждому набору контактных данных соответствует один кандидат, одному кандидату соответствует один набор контактных данных</p> |

На диаграмме уровня сущностей (см. Рисунок 4) представлено графическое представление описания сущностей.

На диаграмме уровня ключей (см. Рисунок 5) представлена логика взаимодействия сущностей.

Полноатрибутная диаграмма представляет собой наиболее полное описание структуры базы данных (см. Рисунок 6)

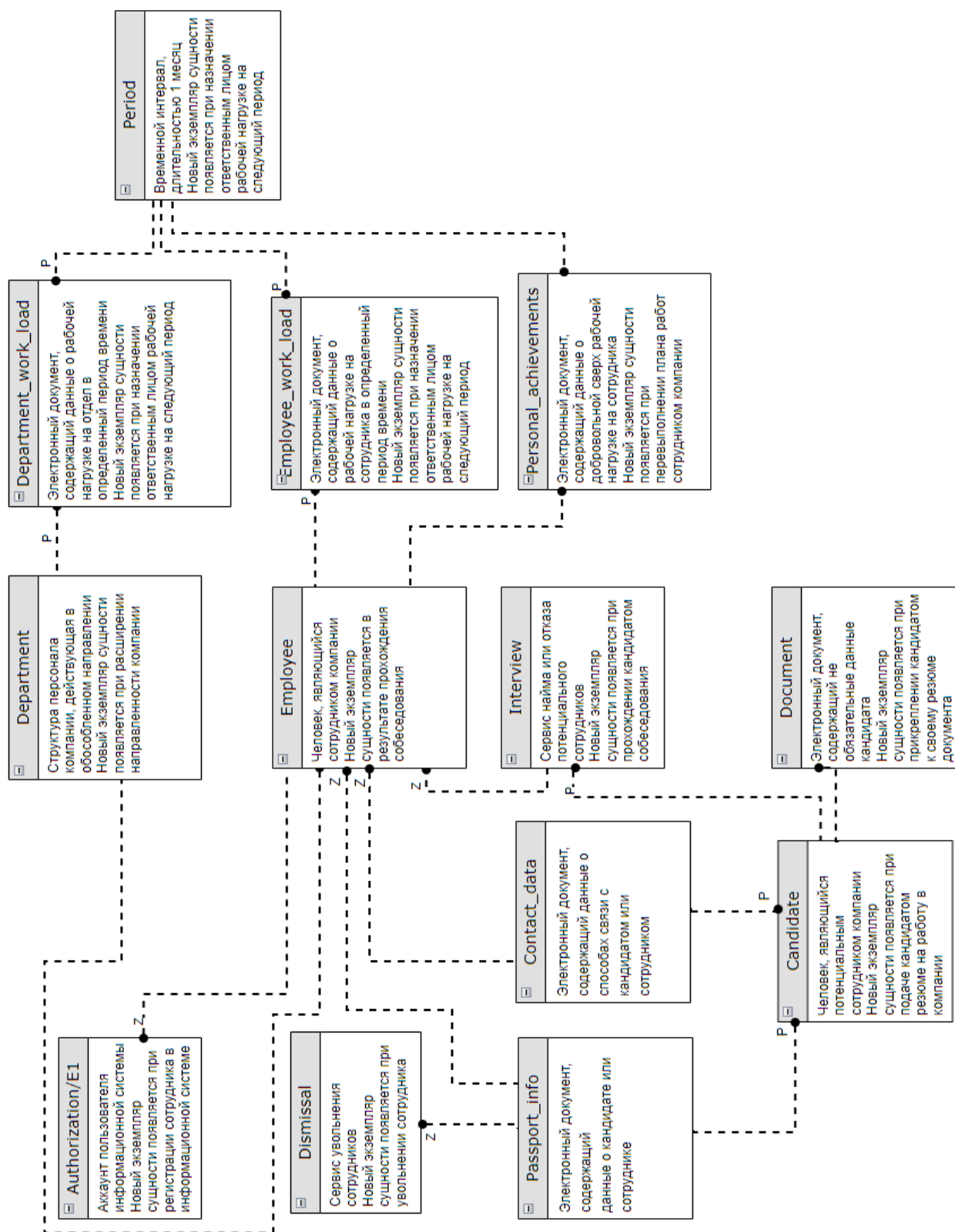


Рис. 4 Диаграмма уровня сущностей.

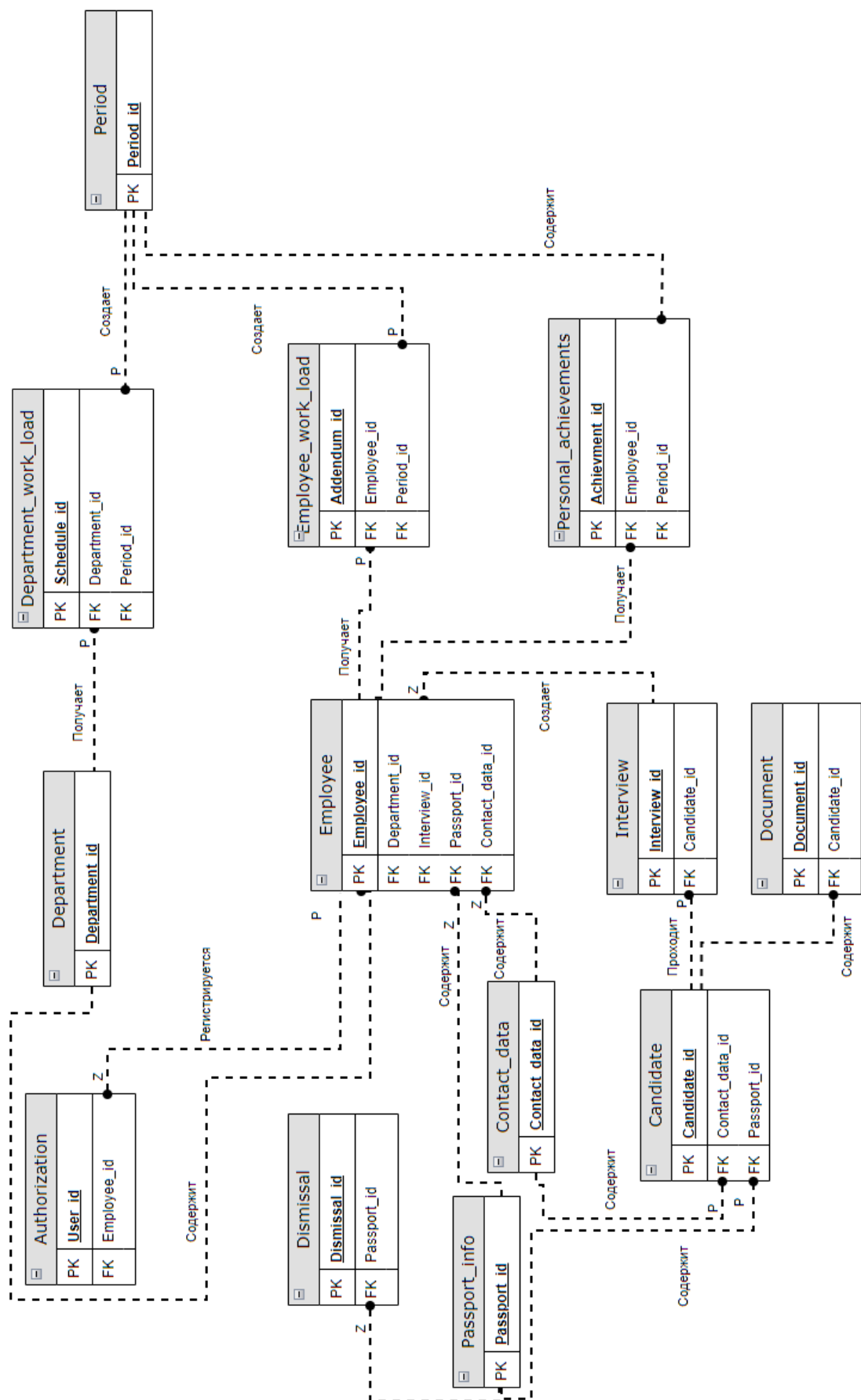


Рис. 5 Диаграмма уровня ключей.

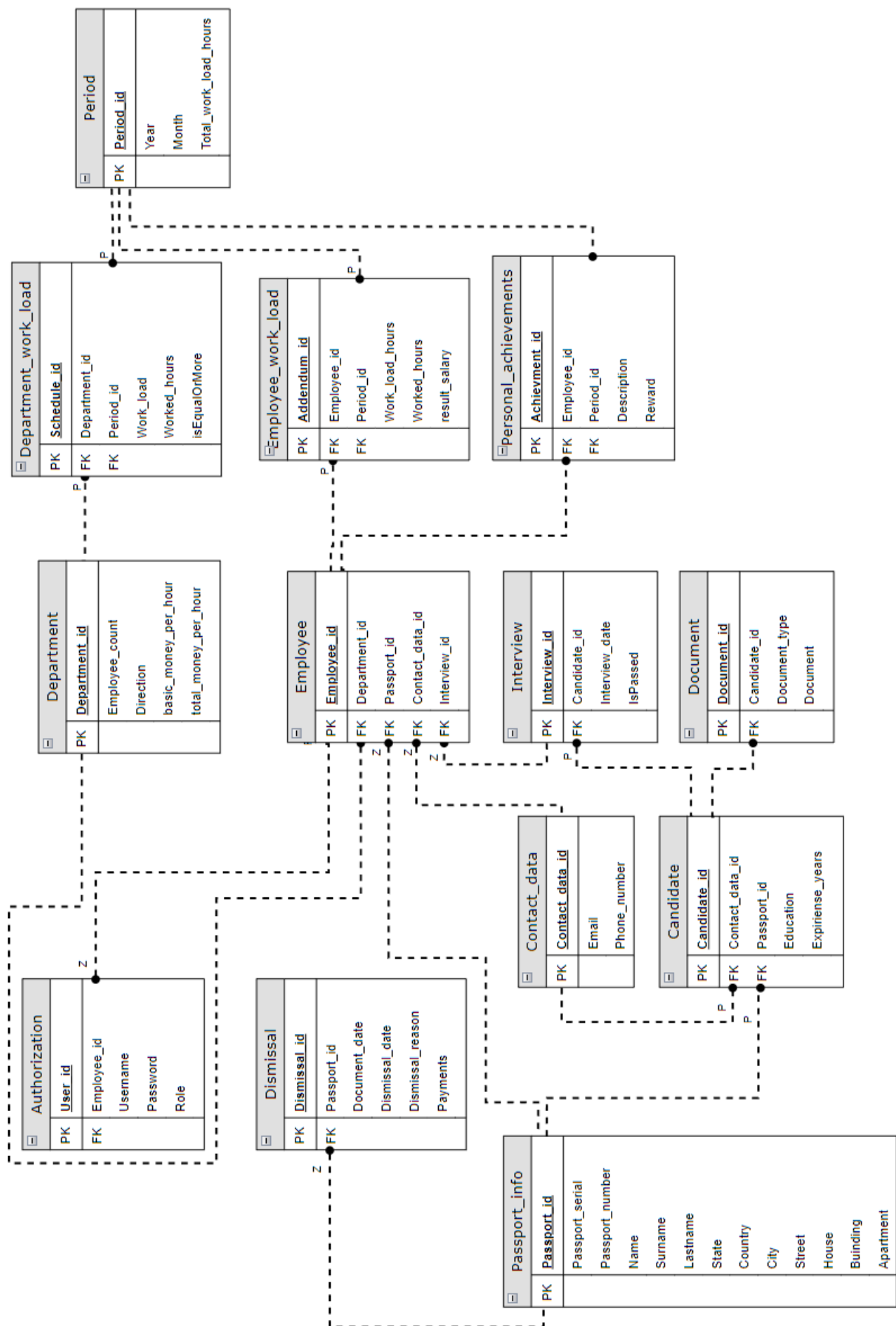


Рис. 6 Полноатрибутная диаграмма.

## 1.2 Структура информационной системы

На начальном этапе разработки была сформирована диаграмма активности (рис. 7), на основе которой, проводилась разработка информационной системы.

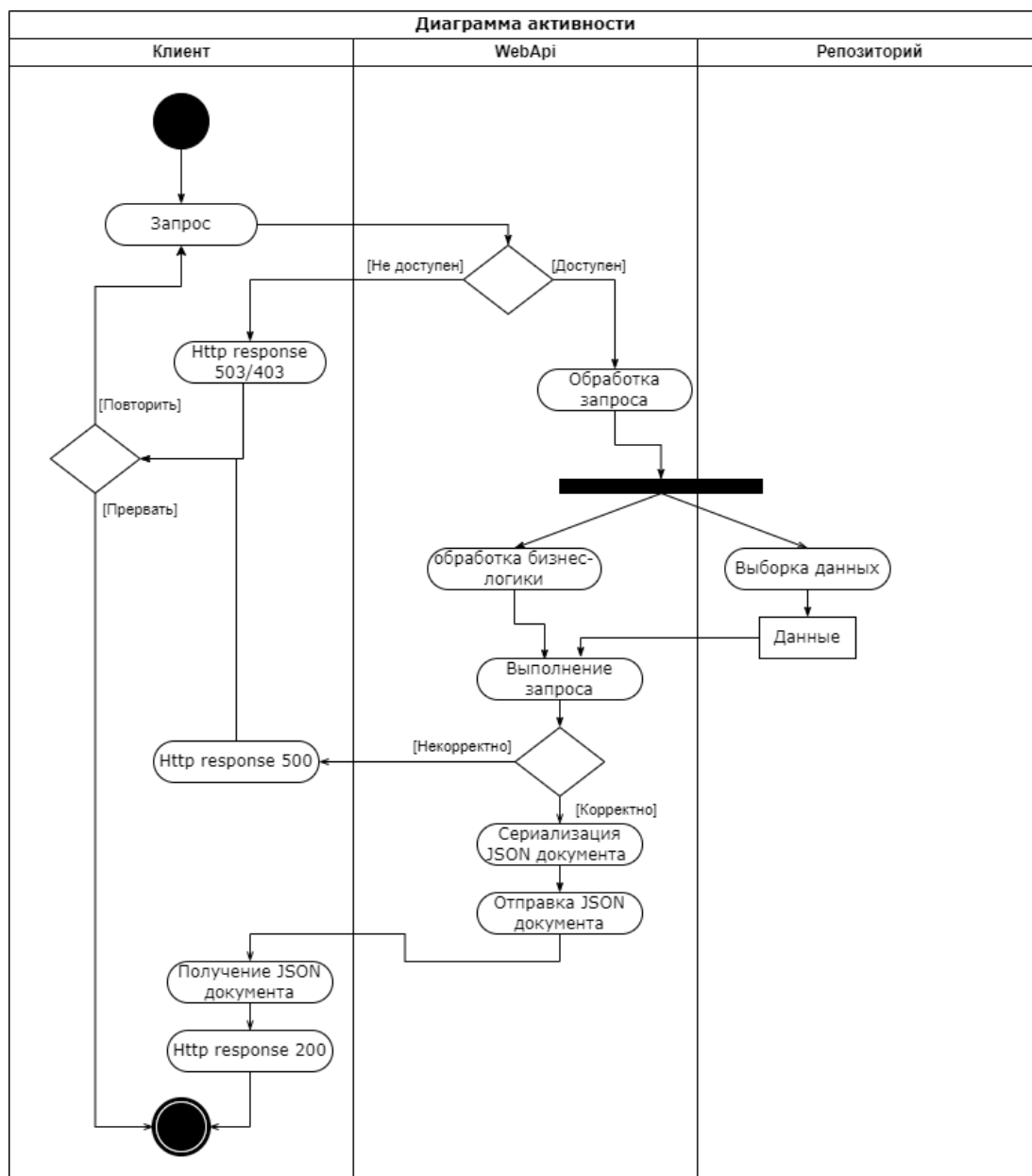


Рис. 7 Диаграмма активности

Основываясь на требованиях к системе, необходимом функционале, а так же особенностях работы была создана приблизительная модель работы информационной системы (см. рис. 8).

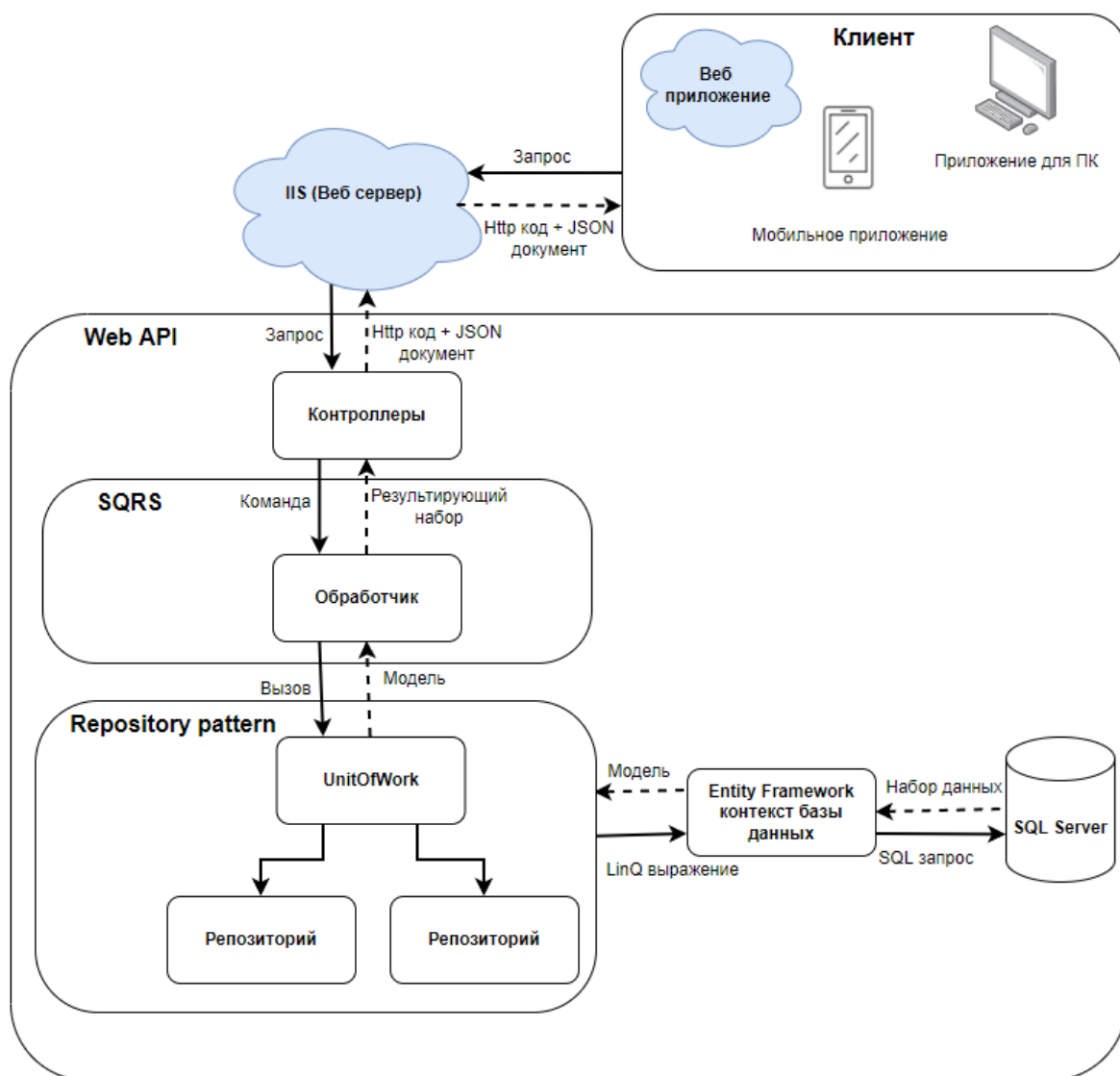


Рис. 8 Модель работы ИС

Для обособленной реализации логики и данных, Web Api построена по принципу чистой архитектуры (рис 9), что позволяет стандартизировать, обезопасить и ускорить выполнение реализованного функционала.

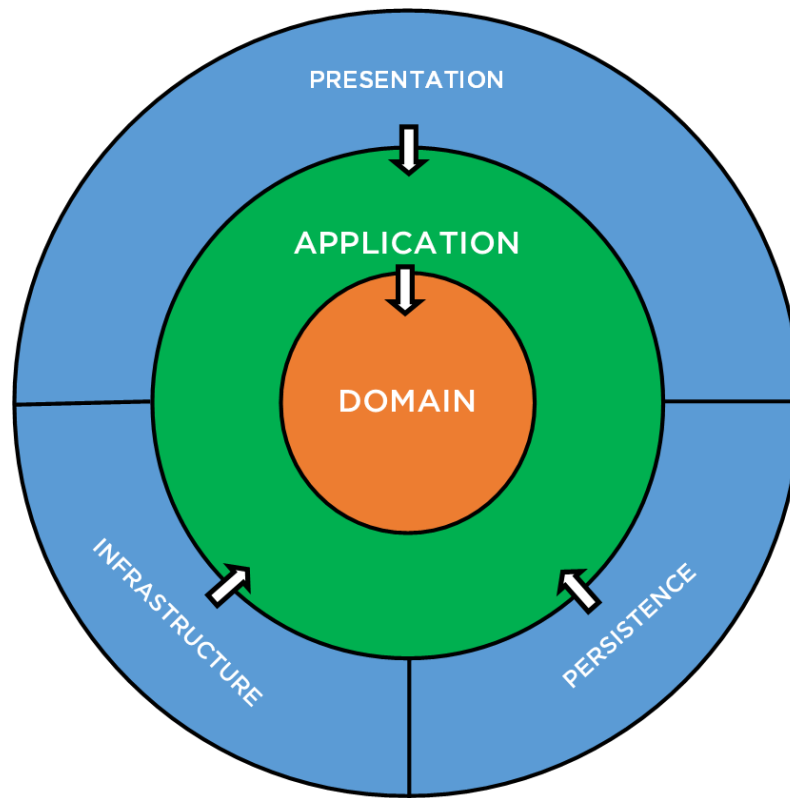


Рис. 9 Чистая архитектура

Разработка производилась от внутренних уровней к внешним (Рис. 10 Структура решения). Уровень Domain содержит модели данных, соответствующие сущностям в базе данных. Уровень Application содержит интерфейсы, бизнес-логику и Dependency Injection, для конфигурации ASP Net.Core API.

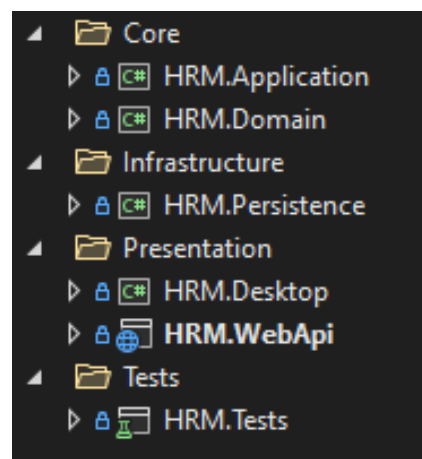


Рис. 10 Структура решения



Далее идет уровень Persistence, который позволяет осуществлять работу с базой данных. Так, как он находится на ступень выше, то может реализовывать интерфейсы уровня Application. Для работы с БД используется Entity Framework 6 (EF), однако в данном проекте, некоторая часть бизнес-логики должна существовать непосредственно на уровне данных. Таким образом было принято решение следовать паттерну IRepository. Данный принцип не только помогает соблюдать подход чистой архитектуры, но и отличается в лучшую сторону в вопросах миграции с одной платформы на другую: к примеру, изначально разработка функционала уровня работы с данными производится на EF 6 под MS SQL, но внезапно возникает необходимость переноса базы данных в Postgre SQL и перестройки инфраструктуры для использования Dapper. В случае прямой разработки, этот процесс может растянуться на месяцы и даже годы, однако, с использованием IRepository, это можно сделать за 5 минут.

Репозиторий можно разделить на 3 уровня. Базой для каждого репозитория служит CRUD – базовый функционал создания, чтения, обновления и удаления данных. Следующий уровень наследует предыдущий и дополнительно реализует необходимый функционал конкретной модели (сущности), к примеру вывод списка не авторизованных в системе сотрудников. Верхний уровень представляет собой unit of work, содержащий все репозитории 2го уровня, методы доступа к ним и метод для сохранения всех изменений.

Для обеспечения минимальной нагрузки на систему, а так же с целью предотвращения заикливания вызова данных, была разработана модель загрузки данных (см. рис. 11)

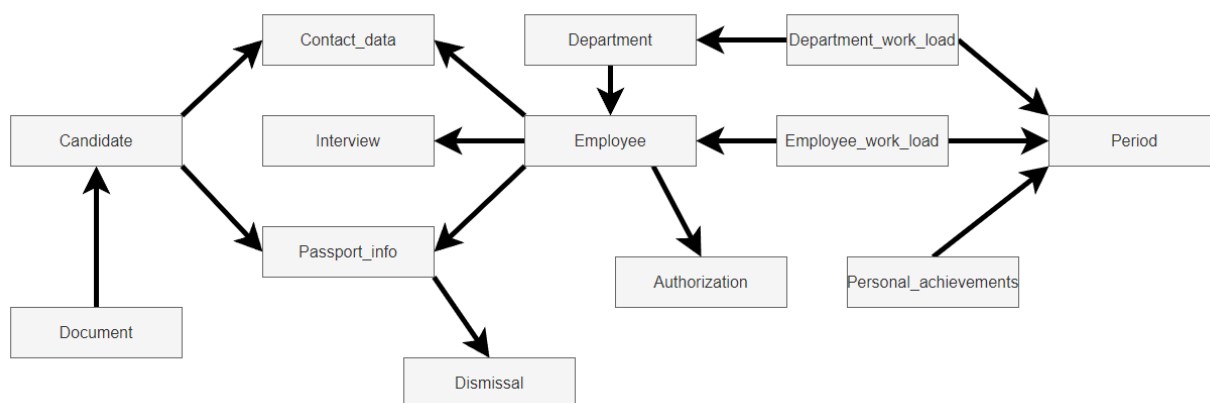


Рис.11 Модель загрузки данных

На данной модели прямоугольниками представлены сущности базы данных, а стрелочками зависимости загрузки. Для примера возьмем Документы кандидата: когда поступает запрос на выборку из базы данных документа кандидата, вместе с ним, для дальнейшей обработки загружаются только кандидат, его контактные и паспортные данные. Модель разработана с расчетом на производительность и возможности дальнейшей реализации необходимого функционала.

Уровень Presentation представлен двумя проектами, первый: ASP.Net Core WebApi, здесь подключаются Dependency Injections из Application и Persistence уровней, благодаря этому WebApi понимает, какую базу данных ему следует использовать, какие валидаторы и репозитории будут обрабатывать данные.

В этом проекте расположены MVC контроллеры, каждый контроллер реализует один или несколько Http методов: HttpGet, HttpPost, HttpPut, HttpDelete и другие.

В соответствии с их названиями, метод HttpGet вызывается автоматически при переходе по URL адресу контроллера и выводит тот набор данных (в виде JSON документа), который прописан в теле метода в контроллере.

Метод `HttpPost` вызывается при отправке формы (в виде JSON документа) на URL адрес контроллера с целью создания экземпляра какого-либо элемента.

Метод `HttpPut` вызывается при отправке формы (в виде JSON документа) при обращении к конкретному экземпляру какого-либо элемента контроллера с целью его изменения.

Метод `HttpDelete` вызывается при обращении к конкретному экземпляру какого-либо элемента с целью удалить его.

Подобное разделение называется `Restful API`, среди его достоинств можно отметить возможности по масштабированию и работе в условиях ограниченной пропускной способности.

Второй проект на этом уровне представляет собой один из вариантов клиентского приложения, в данном конкретном случае оно является приложением для ПК на базе Windows 8 и выше.

Клиентское приложение выполнено посредством технологии Windows Presentation Platform (WPF) .NET Framework 4.7.2 . За общение с `WebApi` отвечает 2 библиотеки: `System.Net.Http` и `Newtonsoft.Json`. Первая библиотека отвечает за получение и отправку `Json` документов на сервер `WebApi`, а вторая библиотека преобразует отправляемые запросы в `Json` документы и получаемые `Json` документы в результат.

### **1.3 Разработка программных модулей**

Помимо базового функционала `CRUD`, для полноценной работы системы необходимо реализовать более сложную логику. С такими задачами хорошо справляется система «команда + обработчик» частично заимствованная из паттерна `SQRS`.

Начнем с модуля распределения плановых рабочих часов на сотрудников по отделам. Для его реализации было написано 2 алгоритма:

- 1) Алгоритм статического распределения часов.
- 2) Алгоритм динамического распределения часов.

Первый алгоритм предполагает, что в систему будет внесено число, соответствующее общей рабочей нагрузке на компанию. При этом распределение будет производиться в первую очередь по сотрудникам, на основе этих данных будет произведен расчет нагрузки на отделы.

Второй алгоритм в качестве входных данных принимает список отделов и соответствующих им рабочих нагрузок. После назначения рабочей нагрузки отделам, происходит ее разделение на сотрудников. После прохождения всех этапов, выполняется перерасчет рабочих часов на отделах (в случае если введенные числа было невозможно поровну разделить между сотрудниками отдела), далее все суммируется и формирует общую нагрузку на компанию.

Модуль формирования дополнительных соглашений берет за основу шаблон, загружаемый в систему при ее настройке. Далее шаблон можно заменить на другой. Для уже существующего распределения рабочих часов можно сгенерировать набор дополнительных соглашений. После этого они хранятся в базе данных в виде массива битов до тех пор, пока не поступит запрос на загрузку. В зависимости от параметров запроса, файлы могут быть конвертированы в любой из представленных форматов. Если поступает запрос на загрузку 2х и более файлов, то они автоматически архивируются.

Модуль авторизации условно разделен на 3 алгоритма: регистрация, аутентификация и авторизация. Регистрация пользователя в системе происходит уже после прохождения всех этапов собеседования и зачисления в штат персонала. Для успешной регистрации пользователь должен указать в специальном поле свой идентификатор, который система автоматически присваивает всем сотрудникам. Узнать его он может у непосредственного начальства. Данная мера предосторожности исключает возможность появления нескольких учетных записей у одного сотрудника, возможность появления неопознанной учетной записи с доступом к внутренним данным

компании и т.д. Аутентификация и авторизация выполнены стандартным набором средств: имя пользователя / логин и пароль.

Модуль приема и увольнения сотрудников интегрирован в структуру персонала. Ключевыми элементами данной структуры являются паспортные и контактные данные. Кандидат записывается в систему после того, как оставит по крайней мере часть из них для дальнейшего прохождения собеседования. Так же кандидат может присылать файлы (дипломы, грамоты, сертификаты и т.д.), они будут прикреплены к личному делу кандидата. По результатам собеседования претендент может быть нанят, в таком случае он предоставляет все необходимые персональные данные, либо не нанят. Информация о кандидате и попытке прохождения собеседования при любом исходе остаются в системе в качестве статистических данных.

При найме сотрудник получает свой идентификатор для авторизации в системе.

Процесс увольнения сотрудника выполняется в несколько этапов:

- 1) Заполнение соответствующей формы.
- 2) Назначение сотруднику часов на 2 последующие недели в случае, если причиной увольнения является собственное желание сотрудника.
- 3) Ожидание окончания рабочего периода.
- 4) Выплата заработной платы сотруднику в количестве, соответствующем отработанным часам.
- 5) Удаление данных об авторизации сотрудника из системы.
- 6) Фактическое увольнение сотрудника.

После увольнения сотрудника, в системе остается информация о его кандидатуре и прохождении им собеседований в качестве статистических данных.

Расчет заработной платы происходит автоматически в конце каждого месяца. Для определения заработной платы сотрудника используются следующие параметры:

- 1) Отработанные часы.

- 2) Соответствие отработанных часов, запланированным.
- 3) Дежурная ставка отдела.
- 4) Временная ставка отдела.
- 5) Премия.

При этом за отработанные сверх плана часы начисляется именно премия, при заполнении менеджером отдела соответствующего пункта. Если сотрудник отработал меньше часов чем указано в его дополнительном соглашении, то с него так же может взиматься штраф, при заполнении менеджером отдела соответствующего пункта.

Дежурная, либо временная ставка отдела указана в дополнительном соглашении и не меняется за указанный период времени. Временная ставка может отличаться от дежурной в случаях, если отдел показал хорошую результативность и высшее руководство решит временно повысить ставку.

## **2.4 Разработка пользовательского интерфейса**

Поскольку клиентское приложение создано на технологии WPF, пользовательский интерфейс будет написан на XAML.

При запуске приложения пользователь заходит на страницу авторизации (см. рис. 12), для продолжения использования приложения необходимо заполнить все поля и нажать кнопку «Войти», если сотрудник еще не зарегистрирован в системе, он может перейти на страницу регистрации (см. рис. 13).

Логин:

Пароль:

☐ Запомнить меня Регистрация

Войти Выход

Рис 12 Страница авторизации.

Логин:

Пароль:

Повторите пароль:

Идентификатор

Продолжить Назад

Рис. 13 Страница регистрации

Для регистрации в системе, помимо логина и пароля, необходимо ввести ключ - идентификатор. Каждому сотруднику присвоен уникальный идентификатор в формате GUID, узнать свой идентификатор можно у менеджера своего отдела, либо у начальства.

После подтверждения личности, сотрудник переходит на главную страницу (см. рис. 14), далее навигация по приложению осуществляется 2мя способами: с помощью меню в верхней панели и кнопками на главной странице.

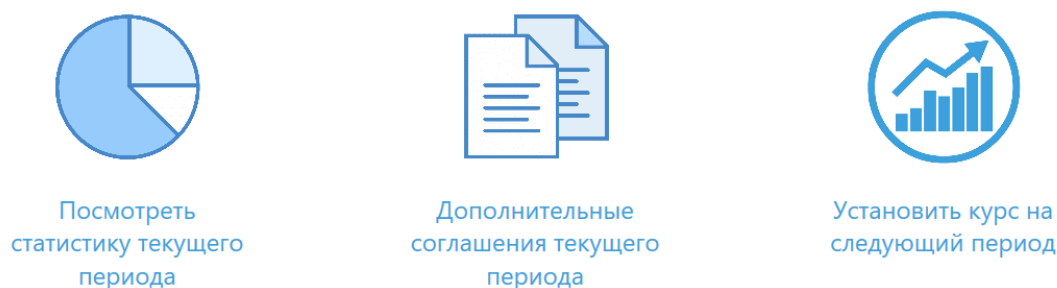


Рис. 14 Главная страница

Для менеджеров всех отделов и начальства предусмотрен следующий функционал: анализ распределения нагрузки по месяцам (см. рис. 15), загрузка (либо формирование) дополнительных соглашений для сотрудников по месяцам/отделам/конкретным сотрудникам (см. рис. 16), для начальства предусмотрен функционал изменения юридической информации предприятия (см. рис. 17), а также распределения рабочей нагрузки на отделы и их сотрудников (см. рис. 18), менеджеры отделов могут составить отчет о личных достижениях своих сотрудников за данный период и назначить премию (см. рис. 19)

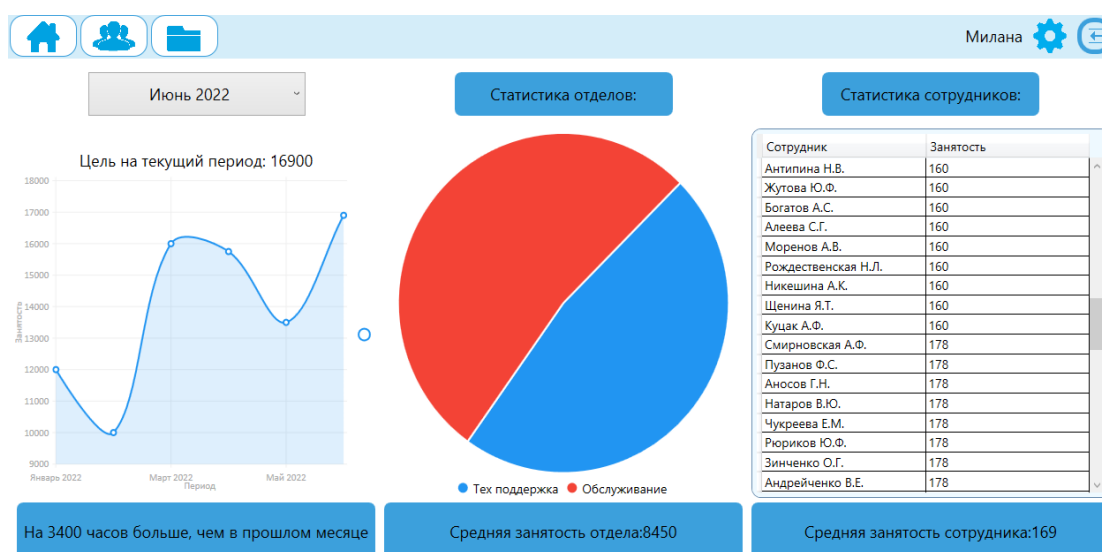


Рис. 15 Страница статистики распределения рабочей нагрузки





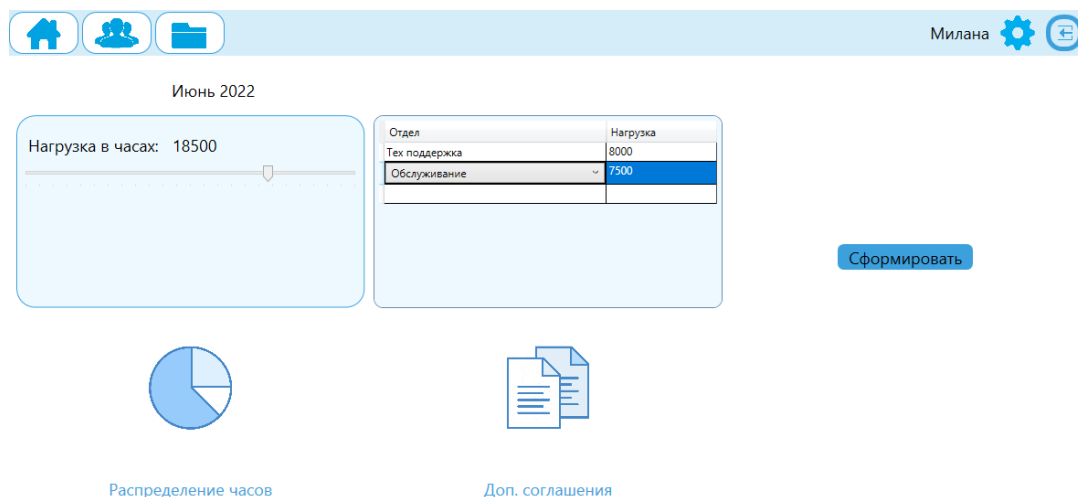


Рис. 18 Страница распределения рабочей нагрузки

| Сотрудник     | Описание  | Премия |
|---------------|---|--------|
| Крутеlev И.С. | Выход в ночную смену 01.06 вместо Коврова Д.Д.                                  | 1500   |
| Яскунова М.Д. | Призовое место во всероссийском чемпионате по компетенции Управление персоналом | 5000   |
| Липов П.Ф.    | Командировка в Кировскую область с 03.06 по 05.06                               | 2000   |
|               |   | 0      |

Рис. 19 Страница заполнения отчета о достижениях сотрудника

Данный пример наглядно иллюстрирует возможности по реализации клиентского приложения. Учитывая специфику конкретных сфер применения информационной системы, в качестве целевой платформы клиентского приложения могут выступать не только ПК, но и телефоны, планшеты и веб решения.

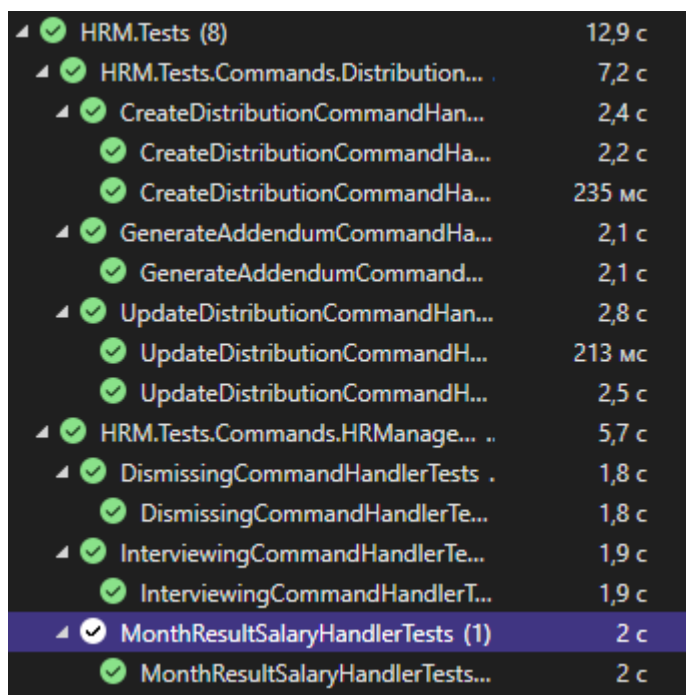
## 2.5 Тестирование и отладка

Тестирование информационной системы можно разделить на 2 этапа: тестирование бизнес-логики и тестирование работы WebApi.

Для выполнения первого этапа тестирования был создан отдельный проект на xUnit, в нем реализована InMemory база данных, идентичная той, которую использует разрабатываемая система. Далее проведено заполнение базы тестовыми данными. После были написаны тесты для каждой критически важной функции. Среди них алгоритмы:

- 1) Статического распределения рабочих часов.
- 2) Динамического распределения рабочих часов.
- 3) Статического обновления распределения рабочих часов.
- 4) Динамического обновления распределения рабочих часов.
- 5) Генерации дополнительных соглашений.
- 6) Прохождения собеседования.
- 7) Увольнения сотрудника.
- 8) Расчета заработной платы.

Все произведенные тесты показали удовлетворительный результат (см. рис. 20)



|  |        |
|--|--------|
| ▲ ✓ HRM.Tests (8)                      | 12,9 с |
| ▲ ✓ HRM.Tests.Commands.Distribution... | 7,2 с  |
| ▲ ✓ CreateDistributionCommandHan...    | 2,4 с  |
| ✓ CreateDistributionCommandHa...       | 2,2 с  |
| ✓ CreateDistributionCommandHa...       | 235 мс |
| ▲ ✓ GenerateAddendumCommandHa...       | 2,1 с  |
| ✓ GenerateAddendumCommand...           | 2,1 с  |
| ▲ ✓ UpdateDistributionCommandHan...    | 2,8 с  |
| ✓ UpdateDistributionCommandH...        | 213 мс |
| ✓ UpdateDistributionCommandH...        | 2,5 с  |
| ▲ ✓ HRM.Tests.Commands.HRManage... ..  | 5,7 с  |
| ▲ ✓ DismissingCommandHandlerTests .    | 1,8 с  |
| ✓ DismissingCommandHandlerTe...        | 1,8 с  |
| ▲ ✓ InterviewingCommandHandlerTe...    | 1,9 с  |
| ✓ InterviewingCommandHandlerT...       | 1,9 с  |
| ▲ ✓ MonthResultSalaryHandlerTests (1)  | 2 с    |
| ✓ MonthResultSalaryHandlerTests...     | 2 с    |

Рис. 20 Результаты тестирования бизнес-логики

Тестирование работы WebApi реализовано с помощью Swagger и его визуального интерфейса (см. рис. 21)

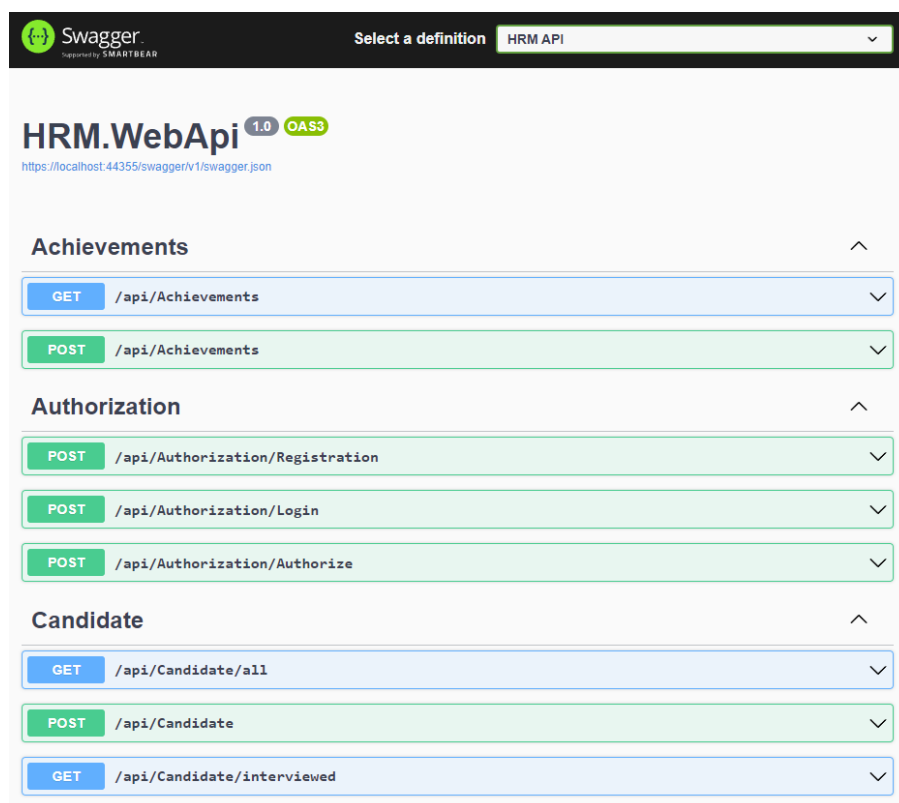


Рис. 21 Визуальный интерфейс Swagger

Преимущества использования Swagger – это возможность редактировать JSON документы перед отправкой, а также всевозможная техническая информация при получении ответа (см. рис. 22, рис. 23).

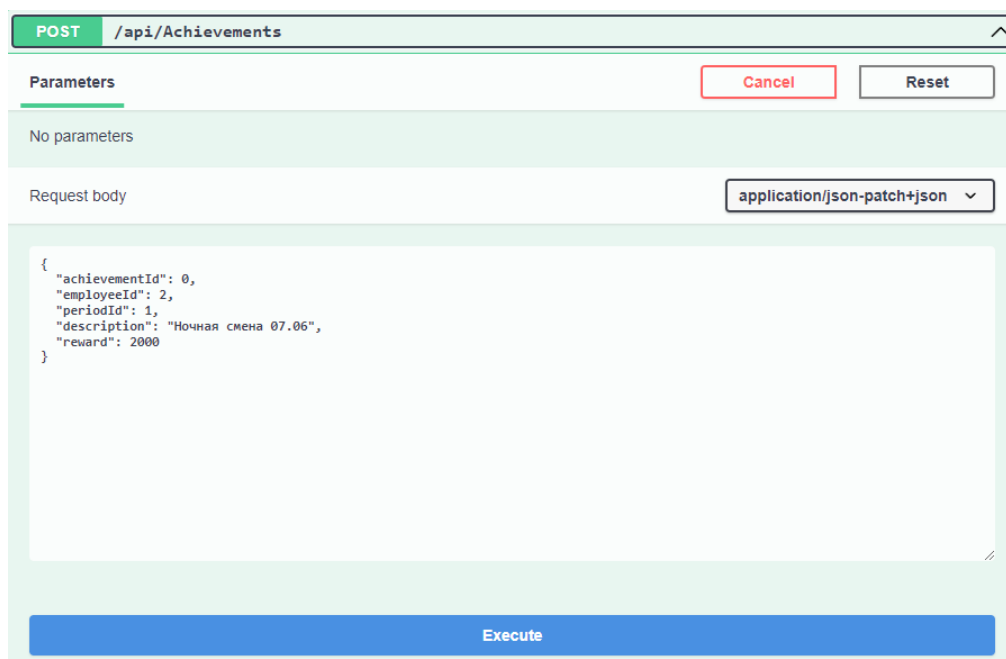


Рис. 22 Пример создания запроса в интерфейсе Swagger

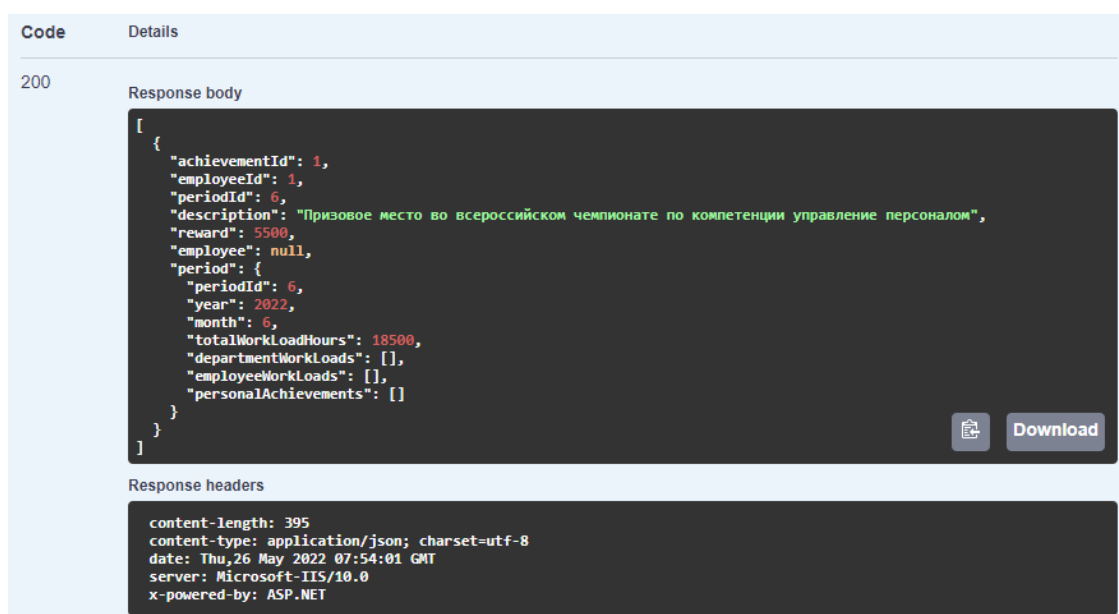


Рис. 23 Пример полученного результата запроса в интерфейсе Swagger

Тестирование посредством Swagger использовалось на протяжении всей работы и помогло выявить и устранить множество дефектов.

## Заключение

Целью данного проекта являлась разработка информационной системы для автоматизации управления персоналом на предприятиях с почасовой формой оплаты труда.

В результате проделанной работы были проанализированы особенности условий труда при найме сотрудников.

Проведено исследование программных средств проектирования и выбраны наиболее подходящие под конкретную задачу, а именно: MS SQL, Visual Studio и Draw.io.

Используя их функционал, была разработана многоуровневая информационная система, включающая 7 проектов:

- 1) База данных HRM
- 2) Библиотека классов (БК) HRM.Domain,
- 3) (БК) HRM.Application,
- 4) (БК) HRM.Persistence,
- 5) (ASP.NET Core) HRM.WebApi,
- 6) (WPF) HRM.Desktop,
- 7) (xUnit) HRM.Tests

В рамках данной информационной системы были реализованы программные модули для распределение плановых рабочих часов на сотрудников по отделам, формирования дополнительных соглашений, регистрации и авторизации пользователей, приема и увольнения сотрудников, расчета заработной платы сотрудников в соответствии с отработанными часами и т.д.

На основании этого можно утверждать, что структура информационной системы является гибко масштабируемой. Функционал Web Api содержит универсальные методы, применимые для различных типов клиентских приложений

Проведено комплексное тестирование программных модулей и всей информационной системы.

Таким образом, поставленные задачи выполнены, цель достигнута.

## **Список используемой литературы**

### **Используемая литература**

1. Преснякова Г. В. Проектирование интегрированных реляционных баз данных: Учебное пособие / Г. В. Преснякова. / М.: КДУ, 2018. / 224 с.
2. Советов Б. Я. Моделирование систем / Б. Я. Советов, С. А. Яковлев. / М.: Высшая школа, 2018. / 343 с.
3. Стружкин Н. П. Базы данных. Проектирование. Учебник / Н. П. Стружкин, В. В. Годин. / М.: Юрайт, 2018. / 478 с.
4. Хомоненко А. Д., Цыганков В. М., Мальцев М. Г. / Базы данных / М.: Учебник для высших учебных заведений, 2018. / 325 с
5. Фуфаев Э. В. Базы данных / Э. В. Фуфаев, Д. Э. Фуфаев. / М.: Академия, 2018. / 320 с.
6. Matthew MacDonald, Pro WPF 4.5 in C# 4th edition 2018. – 1095 с.
7. Brian Driscoll Entity Framework 6 Recipes 2019. - 536 с.
8. Тейлор, Аллен SQL для чайников / Аллен Тейлор. - М.: Вильямс, 2020. - 416 с.
9. James Chamber, ASP.NET Core Application Development, 2018. – 464 с.

### **Интернет-источники**

1. [Анализ распределения рабочей нагрузки] 2022. <https://stats.oecd.org>
2. [Интегрированная среда разработки] 2020. [https://ru.wikipedia.org/wiki/Интегрированная\\_среда\\_разработки](https://ru.wikipedia.org/wiki/Интегрированная_среда_разработки)
3. [Генерация псевдослучайных тестовых данных] 2022. <https://www.mockaroo.com/>
4. [Реверс инжиниринг базы данных] 2022 <https://docs.microsoft.com/ru-ru/ef/core/managing-schemas/scaffolding?tabs=dotnet-core-cli>
5. [Работа с форматом документов docx] 2021 <https://docs.microsoft.com/ru-ru/office/open-xml/open-xml-sdk>
6. [Entity Framework] 2016 <https://metanit.com/sharp/entityframework/>



7. [Entity Framework] 2021 [https://en.wikipedia.org/wiki/Entity\\_Framework](https://en.wikipedia.org/wiki/Entity_Framework)
8. [ItemsControl.ItemTemplate Property] 2021 <https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.controls.itemscontrol.itemtemplate>
9. [JetBrains Rider] 2017 <https://blog.jetbrains.com/dotnet/2017/08/03/rider-2017-1-jetbrains-net-ide-hits-rtm/>
10. [Microsoft Visual Studio] 2021. [https://ru.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio)
11. [Microsoft.Office.Interop.Word Пространство имен] 2021. <https://docs.microsoft.com/ru-ru/dotnet/api/microsoft.office.interop.word>
12. [MS SQL Server 2019] 2021 <https://metanit.com/sql/sqlserver/>
13. [Style.Triggers Property] 2021 <https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.style.triggers>
14. [Style.Setters Property] 2021 <https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.style.setters>
15. [SQLite, MySQL и PostgreSQL сравнения]/ Иван Бирюков 2016. <https://tproger.ru/translations/sqlite-mysql-postgresql-comparison/>
16. [SQLite, MySQL и PostgreSQL сравнения]/ Павел Соловьёв 2018. <https://devacademy.ru/article/sqlite-vs-mysql-vs-postgresql>

## Приложение 1. Код SQL-инструкций для создания базы данных

```
USE [master]
GO
CREATE DATABASE [HRM]
GO
USE [HRM]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Authorization](
    [User_id] [int] IDENTITY(1,1) NOT NULL,
    [Employee_id] [int] NULL,
    [Username] [nvarchar](50) NOT NULL,
    [Password] [nvarchar](50) NOT NULL,
    [Role] [nvarchar](50) NOT NULL,
    CONSTRAINT [PK_Authorization] PRIMARY KEY CLUSTERED
(
    [User_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Candidate](
    [Candidate_id] [int] IDENTITY(1,1) NOT NULL,
    [Passport_id] [int] NOT NULL,
    [Contact_data_id] [int] NOT NULL,
    [Education] [nvarchar](20) NULL,
    [Expiriense_years] [int] NOT NULL,
    CONSTRAINT [PK_Candidate] PRIMARY KEY CLUSTERED
(
    [Candidate_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[CompanyData](
    [CompanyName] [nvarchar](50) NOT NULL,
```

```

        [DirectorName] [nvarchar](100) NOT NULL,
        [CompanyAddress] [nvarchar](100) NOT NULL,
        [INN] [nvarchar](20) NOT NULL,
        [BIK] [nvarchar](9) NOT NULL,
        [KPP] [nvarchar](9) NOT NULL,
        [PAcc] [nvarchar](20) NOT NULL,
        [CAcc] [nvarchar](20) NOT NULL,
        [Bank] [nvarchar](20) NOT NULL,
    CONSTRAINT [PK_CompanyData] PRIMARY KEY CLUSTERED
    (
        [CompanyName] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
    OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Contact_data](
    [Contact_data_id] [int] NOT NULL,
    [Email] [nvarchar](50) NULL,
    [Phone_number] [nvarchar](12) NULL,
    CONSTRAINT [PK_Contact_data] PRIMARY KEY CLUSTERED
    (
        [Contact_data_id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
    OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Department](
    [Department_id] [int] IDENTITY(1,1) NOT NULL,
    [Employee_count] [int] NOT NULL,
    [Direction] [nvarchar](50) NOT NULL,
    [basic_money_per_hour] [float] NOT NULL,
    [total_money_per_hour] [float] NOT NULL,
    [Manager_id] [int] NULL,
    CONSTRAINT [PK_Department] PRIMARY KEY CLUSTERED
    (
        [Department_id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
    OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]
GO

```

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Department_work_load](
    [Schedule_id] [int] IDENTITY(1,1) NOT NULL,
    [Department_id] [int] NOT NULL,
    [Period_id] [int] NOT NULL,
    [Work_load] [int] NOT NULL,
    [Worked_hours] [int] NULL,
    [isEqualOrMore] [bit] NULL,
    CONSTRAINT [PK_Department_work_load] PRIMARY KEY CLUSTERED
(
    [Schedule_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Dismissal](
    [Dismissal_id] [int] IDENTITY(1,1) NOT NULL,
    [Passport_id] [int] NOT NULL,
    [Document_date] [date] NOT NULL,
    [Dismissal_date] [date] NOT NULL,
    [Dismissal_reason] [nvarchar](50) NOT NULL,
    [Payments] [float] NOT NULL,
    CONSTRAINT [PK_Dismissal] PRIMARY KEY CLUSTERED
(
    [Dismissal_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Document](
    [Document_id] [int] NOT NULL,
    [Candidate_id] [int] NOT NULL,
    [Document_type] [nvarchar](50) NOT NULL,
    [Document_url] [nvarchar](50) NOT NULL,
    CONSTRAINT [PK_Document] PRIMARY KEY CLUSTERED
(
    [Document_id] ASC

```

```

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Employee](
    [Employee_id] [int] IDENTITY(1,1) NOT NULL,
    [Passport_id] [int] NOT NULL,
    [Department_id] [int] NULL,
    [Contact_data_id] [int] NOT NULL,
    [Interview_id] [int] NOT NULL,
    [Active] [bit] NOT NULL,
    [Authorization_code] [nvarchar](50) NOT NULL,
    CONSTRAINT [PK_Employee] PRIMARY KEY CLUSTERED
(
    [Employee_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Employee_work_load](
    [Addendum_id] [int] IDENTITY(1,1) NOT NULL,
    [Employee_id] [int] NOT NULL,
    [Period_id] [int] NOT NULL,
    [Work_load_hours] [int] NOT NULL,
    [Worked_hours] [int] NULL,
    [result_salary] [float] NULL,
    CONSTRAINT [PK_Employee_work_load] PRIMARY KEY CLUSTERED
(
    [Addendum_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Files](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [PeriodId] [int] NULL,

```

```

        [EmployeeId] [int] NULL,
        [DepartmentId] [int] NULL,
        [Name] [nvarchar](50) NOT NULL,
        [Data] [varbinary](max) NULL,
    CONSTRAINT [PK_Files] PRIMARY KEY CLUSTERED
    (
        [Id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
    OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Interview](
    [Interview_id] [int] IDENTITY(1,1) NOT NULL,
    [Candidate_id] [int] NOT NULL,
    [IsPassed] [bit] NOT NULL,
    [Date] [date] NOT NULL,
    CONSTRAINT [PK_Interview] PRIMARY KEY CLUSTERED
    (
        [Interview_id] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
    OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Passport_info](
    [Passport_id] [int] NOT NULL,
    [Passport_serial] [int] NOT NULL,
    [Passport_number] [int] NOT NULL,
    [Name] [nvarchar](50) NOT NULL,
    [Surname] [nvarchar](50) NOT NULL,
    [Lastname] [nvarchar](50) NULL,
    [State] [nvarchar](50) NULL,
    [Country] [nvarchar](50) NOT NULL,
    [City] [nvarchar](50) NOT NULL,
    [Street] [nvarchar](50) NOT NULL,
    [House] [int] NOT NULL,
    [Buinding] [int] NULL,
    [Apartment] [int] NOT NULL,
    CONSTRAINT [PK_Passport_info] PRIMARY KEY CLUSTERED
    (
        [Passport_id] ASC

```

```

    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Period](
    [Period_id] [int] IDENTITY(1,1) NOT NULL,
    [Year] [int] NOT NULL,
    [Month] [int] NOT NULL,
    [Total_work_load_hours] [int] NOT NULL,
    CONSTRAINT [PK_Period] PRIMARY KEY CLUSTERED
(
    [Period_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Personal_achievements](
    [Achievement_id] [int] IDENTITY(1,1) NOT NULL,
    [Employee_id] [int] NOT NULL,
    [Period_id] [int] NOT NULL,
    [Description] [nvarchar](200) NOT NULL,
    [Reward] [float] NOT NULL,
    CONSTRAINT [PK_Personal_achievements] PRIMARY KEY CLUSTERED
(
    [Achievement_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
    ) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Authorization_Employee_id] ON
[dbo].[Authorization]
(
    [Employee_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Candidate_Passport_id] ON
[dbo].[Candidate]

```

```

        (
            [Passport_id] ASC
        )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE UNIQUE NONCLUSTERED INDEX [UQ__Candidat__31660442B77D070C]
ON [dbo].[Candidate]
(
    [Contact_data_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE
= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Department_Manager_id] ON
[dbo].[Department]
(
    [Manager_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Department_work_load_Department_id]
ON [dbo].[Department_work_load]
(
    [Department_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Department_work_load_Period_id] ON
[dbo].[Department_work_load]
(
    [Period_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Dismissal_Passport_id] ON
[dbo].[Dismissal]
(
    [Passport_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]

```



```

GO
CREATE NONCLUSTERED INDEX [IX_Document_Candidate_id] ON
[dbo].[Document]
(
    [Candidate_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Employee_Contact_data_id] ON
[dbo].[Employee]
(
    [Contact_data_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Employee_Department_id] ON
[dbo].[Employee]
(
    [Department_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Employee_Interview_id] ON
[dbo].[Employee]
(
    [Interview_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Employee_Passport_id] ON
[dbo].[Employee]
(
    [Passport_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Employee_work_load_Employee_id] ON
[dbo].[Employee_work_load]
(
    [Employee_id] ASC

```

```

        )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Employee_work_load_Period_id] ON
[dbo].[Employee_work_load]
(
    [Period_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Interview_Candidate_id] ON
[dbo].[Interview]
(
    [Candidate_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Personal_achievements_Employee_id]
ON [dbo].[Personal_achievements]
(
    [Employee_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
CREATE NONCLUSTERED INDEX [IX_Personal_achievements_Period_id] ON
[dbo].[Personal_achievements]
(
    [Period_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS
= ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
GO
ALTER TABLE [dbo].[Authorization] WITH CHECK ADD CONSTRAINT
[FK_Authorization_Employee] FOREIGN KEY([Employee_id])
REFERENCES [dbo].[Employee] ([Employee_id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Authorization] CHECK CONSTRAINT
[FK_Authorization_Employee]
GO
ALTER TABLE [dbo].[Candidate] WITH CHECK ADD CONSTRAINT
[FK_Candidate_Contact_data] FOREIGN KEY([Contact_data_id])

```

```

REFERENCES [dbo].[Contact_data] ([Contact_data_id])
GO
ALTER TABLE [dbo].[Candidate] CHECK CONSTRAINT
[FK_Candidate_Contact_data]
GO
ALTER TABLE [dbo].[Candidate] WITH CHECK ADD CONSTRAINT
[FK_Candidate_Passport_info] FOREIGN KEY([Passport_id])
REFERENCES [dbo].[Passport_info] ([Passport_id])
GO
ALTER TABLE [dbo].[Candidate] CHECK CONSTRAINT
[FK_Candidate_Passport_info]
GO
ALTER TABLE [dbo].[Department] WITH CHECK ADD CONSTRAINT
[FK_Department_Employee] FOREIGN KEY([Manager_id])
REFERENCES [dbo].[Employee] ([Employee_id])
GO
ALTER TABLE [dbo].[Department] CHECK CONSTRAINT
[FK_Department_Employee]
GO
ALTER TABLE [dbo].[Department_work_load] WITH CHECK ADD
CONSTRAINT [FK_Department_work_load_Department] FOREIGN
KEY([Department_id])
REFERENCES [dbo].[Department] ([Department_id])
GO
ALTER TABLE [dbo].[Department_work_load] CHECK CONSTRAINT
[FK_Department_work_load_Department]
GO
ALTER TABLE [dbo].[Department_work_load] WITH CHECK ADD
CONSTRAINT [FK_Department_work_load_Period] FOREIGN KEY([Period_id])
REFERENCES [dbo].[Period] ([Period_id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Department_work_load] CHECK CONSTRAINT
[FK_Department_work_load_Period]
GO
ALTER TABLE [dbo].[Dismissal] WITH CHECK ADD CONSTRAINT
[FK_Dismissal_Passport_info] FOREIGN KEY([Passport_id])
REFERENCES [dbo].[Passport_info] ([Passport_id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Dismissal] CHECK CONSTRAINT
[FK_Dismissal_Passport_info]
GO
ALTER TABLE [dbo].[Document] WITH CHECK ADD CONSTRAINT
[FK_Document_Candidate] FOREIGN KEY([Candidate_id])
REFERENCES [dbo].[Candidate] ([Candidate_id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Document] CHECK CONSTRAINT
[FK_Document_Candidate]
GO

```

```

ALTER TABLE [dbo].[Employee] WITH CHECK ADD CONSTRAINT
[FK_Employee_Contact_data] FOREIGN KEY([Contact_data_id])
REFERENCES [dbo].[Contact_data] ([Contact_data_id])
GO
ALTER TABLE [dbo].[Employee] CHECK CONSTRAINT
[FK_Employee_Contact_data]
GO
ALTER TABLE [dbo].[Employee] WITH CHECK ADD CONSTRAINT
[FK_Employee_Department] FOREIGN KEY([Department_id])
REFERENCES [dbo].[Department] ([Department_id])
GO
ALTER TABLE [dbo].[Employee] CHECK CONSTRAINT
[FK_Employee_Department]
GO
ALTER TABLE [dbo].[Employee] WITH CHECK ADD CONSTRAINT
[FK_Employee_Interview] FOREIGN KEY([Interview_id])
REFERENCES [dbo].[Interview] ([Interview_id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Employee] CHECK CONSTRAINT
[FK_Employee_Interview]
GO
ALTER TABLE [dbo].[Employee] WITH CHECK ADD CONSTRAINT
[FK_Employee_Passport_info] FOREIGN KEY([Passport_id])
REFERENCES [dbo].[Passport_info] ([Passport_id])
GO
ALTER TABLE [dbo].[Employee] CHECK CONSTRAINT
[FK_Employee_Passport_info]
GO
ALTER TABLE [dbo].[Employee_work_load] WITH CHECK ADD CONSTRAINT
[FK_Employee_work_load_Employee] FOREIGN KEY([Employee_id])
REFERENCES [dbo].[Employee] ([Employee_id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Employee_work_load] CHECK CONSTRAINT
[FK_Employee_work_load_Employee]
GO
ALTER TABLE [dbo].[Employee_work_load] WITH CHECK ADD CONSTRAINT
[FK_Employee_work_load_Period] FOREIGN KEY([Period_id])
REFERENCES [dbo].[Period] ([Period_id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Employee_work_load] CHECK CONSTRAINT
[FK_Employee_work_load_Period]
GO
ALTER TABLE [dbo].[Interview] WITH CHECK ADD CONSTRAINT
[FK_Interview_Candidate] FOREIGN KEY([Candidate_id])
REFERENCES [dbo].[Candidate] ([Candidate_id])
ON DELETE CASCADE
GO

```

```

        ALTER        TABLE        [dbo].[Interview]        CHECK        CONSTRAINT
[FK_Interview_Candidate]
GO
        ALTER  TABLE  [dbo].[Personal_achievements]        WITH  CHECK  ADD
CONSTRAINT        [FK_Personal_achievements_Employee]        FOREIGN
KEY([Employee_id])
        REFERENCES [dbo].[Employee] ([Employee_id])
        ON DELETE CASCADE
GO
        ALTER  TABLE  [dbo].[Personal_achievements]        CHECK  CONSTRAINT
[FK_Personal_achievements_Employee]
GO
        ALTER  TABLE  [dbo].[Personal_achievements]        WITH  CHECK  ADD
CONSTRAINT [FK_Personal_achievements_Period] FOREIGN KEY([Period_id])
        REFERENCES [dbo].[Period] ([Period_id])
        ON DELETE CASCADE
GO
        ALTER  TABLE  [dbo].[Personal_achievements]        CHECK  CONSTRAINT
[FK_Personal_achievements_Period]
GO
        USE [master]
GO
        ALTER DATABASE [HRM] SET READ_WRITE
GO

```

## Приложение 2. Код уровня Domain

Класс Authorizations:

```
namespace HRM.Domain
{
    public partial class Authorization
    {
        public int UserId { get; set; }
        public int? EmployeeId { get; set; }
        public string Username { get; set; } = null!;
        public string Password { get; set; } = null!;
        public string Role { get; set; } = null!;

        public virtual Employee? Employee { get; set; }
    }
}
```

Класс Candidate:

```
namespace HRM.Domain
{
    public partial class Candidate
    {
        public Candidate()
        {
            Documents = new HashSet<Document>();
            Interviews = new HashSet<Interview>();
        }

        public int CandidateId { get; set; }
        public int PassportId { get; set; }
        public int ContactDataId { get; set; }
        public string? Education { get; set; }
        public int ExpirienseYears { get; set; }

        public virtual ContactData ContactData { get; set; } =
null!;
        public virtual PassportInfo Passport { get; set; } = null!;
        public virtual ICollection<Document> Documents { get; set; }
    }

    public virtual ICollection<Interview> Interviews { get;
set; }
}
}
```

Класс CompanyData:

```
namespace HRM.Domain
{
    public class CompanyData
```

```

    {
        public string CompanyName { get; set; }
        public string DirectorName { get; set; }
        public string CompanyAddress { get; set; }
        public string INN { get; set; }
        public string BIK { get; set; }
        public string KPP { get; set; }
        public string PAcc { get; set; }
        public string CAcc { get; set; }
        public string Bank { get; set; }
    }
}

```

Класс ContactData:

```

namespace HRM.Domain
{
    public partial class ContactData
    {
        public ContactData()
        {
            Employees = new HashSet<Employee>();
        }

        public int ContactDataId { get; set; }
        public string? Email { get; set; }
        public string? PhoneNumber { get; set; }

        public virtual Candidate Candidate { get; set; } = null!;
        public virtual ICollection<Employee> Employees { get; set; }
    }
}

```

Класс Department:

```

namespace HRM.Domain
{
    public partial class Department
    {
        public Department()
        {
            DepartmentWorkLoads = new
HashSet<DepartmentWorkLoad>();
            Employees = new HashSet<Employee>();
        }

        public int DepartmentId { get; set; }
        public int EmployeeCount { get; set; }
        public string Direction { get; set; } = null!;
        public double BasicMoneyPerHour { get; set; }
    }
}

```

```

        public double TotalMoneyPerHour { get; set; }
        public int? ManagerId { get; set; }

        public virtual Employee? Manager { get; set; }
        public virtual ICollection<DepartmentWorkLoad>
DepartmentWorkLoads { get; set; }
        public virtual ICollection<Employee> Employees { get; set;
}
    }
}

```

Класс DepartmentWorkLoad:

```

namespace HRM.Domain
{
    public partial class DepartmentWorkLoad
    {
        public int ScheduleId { get; set; }
        public int DepartmentId { get; set; }
        public int PeriodId { get; set; }
        public int WorkLoad { get; set; }
        public int? WorkedHours { get; set; }
        public bool? IsEqualOrMore { get; set; }

        public virtual Department Department { get; set; } = null!;
        public virtual Period Period { get; set; } = null!;
    }
}

```

Класс Dismissal:

```

namespace HRM.Domain
{
    public partial class Dismissal
    {
        public int DismissalId { get; set; }
        public int PassportId { get; set; }
        public DateTime DocumentDate { get; set; }
        public DateTime DismissalDate { get; set; }
        public string DismissalReason { get; set; } = null!;
        public double Payments { get; set; }

        public virtual PassportInfo Passport { get; set; } = null!;
    }
}

```

Класс Document:

```

namespace HRM.Domain
{
    public partial class Document

```



```

    {
        public int DocumentId { get; set; }
        public int CandidateId { get; set; }
        public string DocumentType { get; set; } = null!;
        public string DocumentUrl { get; set; } = null!;

        public virtual Candidate Candidate { get; set; } = null!;
    }
}

```

Класс Employee:

```

namespace HRM.Domain
{
    public partial class Employee
    {
        public Employee()
        {
            Authorizations = new HashSet<Authorization>();
            Departments = new HashSet<Department>();
            EmployeeWorkLoads = new HashSet<EmployeeWorkLoad>();
            PersonalAchievements = new
HashSet<PersonalAchievement>();
        }

        public int EmployeeId { get; set; }
        public int PassportId { get; set; }
        public int? DepartmentId { get; set; }
        public int ContactDataId { get; set; }
        public int InterviewId { get; set; }
        public bool Active { get; set; }
        public string AuthorizationCode { get; set; } = null!;

        public virtual ContactData ContactData { get; set; } =
null!;
        public virtual Department? Department { get; set; }
        public virtual Interview Interview { get; set; } = null!;
        public virtual PassportInfo Passport { get; set; } = null!;
        public virtual ICollection<Authorization> Authorizations {
get; set; }
        public virtual ICollection<Department> Departments { get;
set; }
        public virtual ICollection<EmployeeWorkLoad>
EmployeeWorkLoads { get; set; }
        public virtual ICollection<PersonalAchievement>
PersonalAchievements { get; set; }
    }
}

```

Класс EmployeeWorkLoad:

```

namespace HRM.Domain
{
    public partial class EmployeeWorkLoad
    {
        public int AddendumId { get; set; }
        public int EmployeeId { get; set; }
        public int PeriodId { get; set; }
        public int WorkLoadHours { get; set; }
        public int? WorkedHours { get; set; }
        public double? ResultSalary { get; set; }

        public virtual Employee Employee { get; set; } = null!;
        public virtual Period Period { get; set; } = null!;
    }
}

```

Класс File:

```

namespace HRM.Domain
{
    public class File
    {
        public int Id { get; set; }
        public int? EmployeeId { get; set; }
        public int? DepartmentId { get; set; }
        public int? PeriodId { get; set; }
        public string Name { get; set; }
        public byte[] Data { get; set; }
    }
}

```

Класс Interview:

```

namespace HRM.Domain
{
    public partial class Interview
    {
        public Interview()
        {
            Employees = new HashSet<Employee>();
        }

        public int InterviewId { get; set; }
        public int CandidateId { get; set; }
        public bool IsPassed { get; set; }
        public DateTime Date { get; set; }

        public virtual Candidate Candidate { get; set; } = null!;
        public virtual ICollection<Employee> Employees { get; set; }
    }
}

```

```
}
```

Класс PassportInfo:

```
namespace HRM.Domain
{
    public partial class PassportInfo
    {
        public PassportInfo()
        {
            Candidates = new HashSet<Candidate>();
            Employees = new HashSet<Employee>();
            Dismissals = new HashSet<Dismissal>();
        }

        public int PassportId { get; set; }
        public int PassportSerial { get; set; }
        public int PassportNumber { get; set; }
        public string Name { get; set; } = null!;
        public string Surname { get; set; } = null!;
        public string? Lastname { get; set; }
        public string? State { get; set; }
        public string Country { get; set; } = null!;
        public string City { get; set; } = null!;
        public string Street { get; set; } = null!;
        public int House { get; set; }
        public int? Building { get; set; }
        public int Apartment { get; set; }

        public virtual ICollection<Candidate> Candidates { get;
set; }
        public virtual ICollection<Employee> Employees { get; set;
}
        public virtual ICollection<Dismissal> Dismissals { get;
set; }
    }
}
```

Класс Period:

```
namespace HRM.Domain
{
    public partial class Period
    {
        public Period()
        {
            DepartmentWorkLoads = new
HashSet<DepartmentWorkLoad>();
            EmployeeWorkLoads = new HashSet<EmployeeWorkLoad>();
            PersonalAchievements = new
HashSet<PersonalAchievement>();
        }
    }
}
```

```

    }

    public int PeriodId { get; set; }
    public int Year { get; set; }
    public int Month { get; set; }
    public int TotalWorkLoadHours { get; set; }

    public virtual ICollection<DepartmentWorkLoad>
DepartmentWorkLoads { get; set; }
    public virtual ICollection<EmployeeWorkLoad>
EmployeeWorkLoads { get; set; }
    public virtual ICollection<PersonalAchievement>
PersonalAchievements { get; set; }
    }
}

```

Класс PersonalAchievement:

```

namespace HRM.Domain
{
    public partial class PersonalAchievement
    {
        public int AchievementId { get; set; }
        public int EmployeeId { get; set; }
        public int PeriodId { get; set; }
        public string Description { get; set; } = null!;
        public double Reward { get; set; }

        public virtual Employee Employee { get; set; } = null!;
        public virtual Period Period { get; set; } = null!;
    }
}

```

### Приложение 3. Код уровня Application

Интерфейсы репозиториев

Интерфейс IRepositoryBase:

```
namespace HRM.Application.BuisnessLogic.Base
{
    public interface IRepositoryBase<T>
    {
        Task<T> CreateAsync(T entity);
        Task UpdateAsync(T entity);
        Task DeleteAsync(T entity);
        Task<T?> GetByIdAsync(int id);
        Task<T?> FirstAsync();
        Task<T?> LastAsync();
        Task<List<T>> GetAllAsync();
    }
}
```

Интерфейс IAchievementRepository:

```
using HRM.Application.BuisnessLogic.Base;
using HRM.Domain;

namespace HRM.Application.BuisnessLogic
{
    public interface IAchievementRepository :
    IRepositoryBase<PersonalAchievement>
    {
    }
}
```

Интерфейс IAuthorizationRepository:

```
using HRM.Application.BuisnessLogic.Base;

namespace HRM.Application.BuisnessLogic
{
    public interface IAuthorizationRepository :
    IRepositoryBase<Domain.Authorization>
    {
        public bool IsUsed(string username);
    }
}
```

```

    }
}

```

Интерфейс ICandidateRepository:

```

using HRM.Application.BuisnessLogic.Base;
using HRM.Domain;

namespace HRM.Application.BuisnessLogic
{
    public interface ICandidateRepository :
    IRepositoryBase<Candidate>
    {
        Task<List<Candidate>> GetNotInterviewed();
        Task<List<Candidate>> GetInterviewed();
        bool IsInterviewed(int candidateId);
    }
}

```

Интерфейс ICompanyDataRepository:

```

using HRM.Application.BuisnessLogic.Base;

namespace HRM.Application.BuisnessLogic
{
    public interface ICompanyDataRepository :
    IRepositoryBase<Domain.CompanyData>
    {

    }
}

```

Интерфейс IContactDataRepository:

```

using HRM.Application.BuisnessLogic.Base;
using HRM.Domain;

namespace HRM.Application.BuisnessLogic
{
    public interface IContactDataRepository :
    IRepositoryBase<ContactData>
    {
        IEnumerable<ContactData> GetAllWithMissing();
        IEnumerable<ContactData>
        GetAllWithMissingByDepartment(int id);
    }
}

```

Интерфейс IDepartmentRepository:

```

using HRM.Application.BuisnessLogic.Base;
using HRM.Domain;

namespace HRM.Application.BuisnessLogic
{
    public interface IDepartmentRepository :
    IRepositoryBase<Department>
    {
        bool IsEmployeeManager(int employeeId);
    }
}

```

Интерфейс IDepartmentWorkLoadRepository:

```

using HRM.Application.BuisnessLogic.Base;
using HRM.Domain;

namespace HRM.Application.BuisnessLogic
{
    public interface IDepartmentWorkLoadRepository :
    IRepositoryBase<DepartmentWorkLoad>
    {
        IEnumerable<DepartmentWorkLoad> GetWithNotEnoughHours();
        IEnumerable<DepartmentWorkLoad> GetByPeriodId(int
periodId);
        DepartmentWorkLoad GetByPeriodIdPerDepartment(int
periodId, int departmentId);
    }
}

```

Интерфейс IDismissalRepository:

```

using HRM.Application.BuisnessLogic.Base;
using HRM.Domain;

namespace HRM.Application.BuisnessLogic
{
    public interface IDismissalRepository :
    IRepositoryBase<Dismissal>
    {
    }
}

```

Интерфейс IEmployeeRepository:

```

using HRM.Application.BuisnessLogic.Base;
using HRM.Domain;

namespace HRM.Application.BuisnessLogic
{

```

```

        public interface IEmployeeRepository :
        IRepositoryBase<Employee>
        {
            IEnumerable<Employee> GetActive();
            IEnumerable<Employee> GetActiveByDepartmentId(int id);
            Employee GetByAuthData(string login, string password);
            Employee GetByAuthCode(string authCode);
            IEnumerable<Employee> GetWithMissingContactData();
            IEnumerable<Employee> GetAuthorizer();
            IEnumerable<Employee> GetUnauthorized();
            IEnumerable<Employee>
            GetActiveWithNoWorkLoadByPeriodId(int periodId);
            IEnumerable<Employee>
            GetActiveWithNoWorkLoadByPeriodIdPerDepartment(int periodId, int
            departmentId);

            string GetFullName(Employee employee);
            string GetInits(Employee employee);
            string GetAddress(Employee employee);
        }
    }

```

Интерфейс IEmployeeWorkLoadRepository:

```

using HRM.Application.BuisnessLogic.Base;
using HRM.Domain;

namespace HRM.Application.BuisnessLogic
{
    public interface IEmployeeWorkLoadRepository :
    IRepositoryBase<EmployeeWorkLoad>
    {
        IEnumerable<EmployeeWorkLoad> GetWithNotEnoughHours();
        IEnumerable<EmployeeWorkLoad> GetByPeriodId(int periodId);
        IEnumerable<EmployeeWorkLoad>
        GetByPeriodIdPerDepartment(int periodId, int departmentId);
        int SumPeriodWorkLoadHoursByDepartmentId(int periodId, int
        departmentId);
        int SumPeriodWorkLoadHours(int periodId);
    }
}

```

Интерфейс IFileRepository:

```

using HRM.Application.BuisnessLogic.Base;
using File = HRM.Domain.File;

namespace HRM.Application.BuisnessLogic
{
    public interface IFileRepository : IRepositoryBase<File>
    {

```



```

        public IEnumerable<File> GetAllByEmployeeId(int
employeeId);
        public IEnumerable<File> GetAllByPeriodId(int periodId);
        public File GetOne(int employeeId, int periodId);
        public IEnumerable<File> GetByDepartmentId(int
departmentId, int periodId);

        public Task ClearStorage(string path);
    }
}

```

Интерфейс InterviewRepository:

```

using HRM.Application.BuisnessLogic.Base;
using HRM.Domain;

namespace HRM.Application.BuisnessLogic
{
    public interface IInterviewRepository :
IRepositoryBase<Interview>
    {
    }
}

```

Интерфейс IPeriodRepository:

```

using HRM.Application.BuisnessLogic.Base;
using HRM.Domain;

namespace HRM.Application.BuisnessLogic
{
    public interface IPeriodRepository : IRepositoryBase<Period>
    {
        Period Next();
        Period CreateFirst();
        Period? GetPeriodByDate(DateTime date);
        public DateTime GetDateFromPeriod(Peiod period);
    }
}

```

Интерфейс IUnitOfWork:

```

using HRM.Application.BuisnessLogic;

namespace HRM.Application.Interfaces
{
    public interface IUnitOfWork
    {
        IAuthorizationRepository Authorization { get; set; }
        IAchievementRepository Achievement { get; set; }
    }
}

```

```

        ICandidateRepository Candidate { get; set; }
        ICompanyDataRepository CompanyData { get; set; }
        IContactDataRepository ContactData { get; set; }
        IDepartmentRepository Department { get; set; }
        IDepartmentWorkLoadRepository DepartmentWorkLoad { get;
set; }

        IEmployeeRepository Employee { get; set; }
        IEmployeeWorkLoadRepository EmployeeWorkLoad { get; set; }
        IFileRepository File { get; set; }
        IDismissalRepository Dismissal { get; set; }
        IInterviewRepository Interview { get; set; }
        IPeriodRepository Period { get; set; }
        Task Save();
    }
}

```

Интерфейс контекста базы данных IHRMDBContext:

```

using HRM.Domain;
using Microsoft.EntityFrameworkCore;
using File = HRM.Domain.File;

namespace HRM.Application.Interfaces
{
    public interface IHRMDBContext
    {
        public DbSet<Domain.Authorization> Authorizations { get;
set; }

        public DbSet<Candidate> Candidates { get; set; }
        public DbSet<Domain.CompanyData> CompanyData { get; set; }
        public DbSet<ContactData> ContactData { get; set; }
        public DbSet<Department> Departments { get; set; }
        public DbSet<DepartmentWorkLoad> DepartmentWorkLoads {
get; set; }

        public DbSet<Dismissal> Dismissals { get; set; }
        public DbSet<Document> Documents { get; set; }
        public DbSet<Employee> Employees { get; set; }
        public DbSet<EmployeeWorkLoad> EmployeeWorkLoads { get;
set; }

        public DbSet<Interview> Interviews { get; set; }
        public DbSet<PassportInfo> PassportInfos { get; set; }
        public DbSet<Period> Periods { get; set; }
        public DbSet<PersonalAchievement> PersonalAchievements {
get; set; }

        public DbSet<File> Files { get; set; }

        Task<int> SaveChangesAsync(CancellationToken
cancellationToken);
    }
}

```

## Классы ошибок

Класс ошибки валидации запроса ValidationException:

```
namespace HRM.Application.Exceptions
{
    public class ValidationException : Exception
    {
        public ValidationException(string key) : base($"{key}
validation failed.") { }
    }
}
```

## Модуль авторизации пользователя

Класс команды входа в систему LoginCommand:

```
using System.ComponentModel.DataAnnotations;

namespace HRM.Application.Authorization.Login
{
    public class LoginCommand
    {
        [Required]
        public string Username { get; set; }
        [Required]
        public string Password { get; set; }
    }
}
```

Класс обработчика команды входа в систему LoginCommandHandler:

```
using HRM.Application.Interfaces;

namespace HRM.Application.Authorization.Login
{
    public class LoginCommandHandler
    {
        private readonly IUnitOfWork unitOfWork;
        public LoginCommandHandler(IUnitOfWork unitOfWork)
        {
            this.unitOfWork = unitOfWork;
        }
        public async Task<int> TryLogin(LoginCommand request)
        {
            foreach (var auth in unitOfWork.Authorization.GetAllAsync().Result)
            {
                if (request.Username == auth.Username &&
request.Password == auth.Password)
                {
                    if (auth.Role == "Administrator")

```

```

        return 3;
    else if (auth.Role == "Manager")
        return 2;
    else if (auth.Role == "User")
        return 1;
    }
}
return 0;
}
}
}

```

Класс команды регистрации пользователя в системе

RegistrationCommand:

```

using System.ComponentModel.DataAnnotations;

namespace HRM.Application.Authorization.Registration
{
    public class RegistrationCommand
    {
        [Required]
        public string Username { get; set; }
        [Required]
        public string Password { get; set; }
        public string? AuthCode { get; set; }
    }
}

```

Класс обработчика команды регистрации пользователя в системе

RegistrationCommandHandler:

```

using HRM.Application.Interfaces;
using HRM.Application.Exceptions;

namespace HRM.Application.Authorization.Registration
{
    public class RegistrationCommandHandler
    {
        private IUnitOfWork unitOfWork;
        public RegistrationCommandHandler(IUnitOfWork unitOfWork)
        {
            this.unitOfWork = unitOfWork;
        }
        public async Task Registration(RegistrationCommand
request)
        {
            if (!RegistrationCommandValidator.Validate(request))
                throw new
ValidationException(nameof(RegistrationCommand));

```

```

        var current = new Domain.Authorization()
        {
            Username = request.Username,
            Password = request.Password,
        };

        if
(!unitOfWork.Authorization.GetAllAsync().Result.Any())
        {
            current.Role = "Administrator";
            await
unitOfWork.Authorization.CreateAsync(current);
            await unitOfWork.Save();
        }
        else
        {
            if (request.AuthCode == null)
                throw new Exception(); //auth code is required
            var person =
unitOfWork.Employee.GetByAuthCode(request.AuthCode);
            if (person == null)
                throw new Exception(); //registration fail

            if(unitOfWork.Authorization.IsUsed(request.Username))
                throw new Exception(); //choose another
            username exception

            current.EmployeeId = person.EmployeeId;

            if
(unitOfWork.Department.IsEmployeeManager(person.EmployeeId))
                current.Role = "Manager";
            else
                current.Role = "User";
            await
unitOfWork.Authorization.CreateAsync(current);
            await unitOfWork.Save();
        }
    }
}

```

Класс валидатор команды регистрации пользователя в системе  
RegistrationCommandValidator:

```

namespace HRM.Application.Authorization.Registration
{
    public static class RegistrationCommandValidator
    {

```

```

        public static bool Validate(RegistrationCommand request)
        {
            if (request.Username.Contains(' '))
                return false;
            if (request.Username.Length < 5 || request.Username.Length > 20)
                return false;
            if (request.Password.Contains(' '))
                return false;
            if (request.Password.Length < 5)
                return false;
            return true;
        }
    }
}

```

Модуль заполнения данных о компании

Класс команды заполнения данных о компании

FillCompanyDataCommand:

```

namespace HRM.Application.CompanyData
{
    public class FillCompanyDataCommand
    {
        public string? CompanyName { get; set; }
        public string? DirectorName { get; set; }
        public string? CompanyAddress { get; set; }
        public string? INN { get; set; }
        public string? BIK { get; set; }
        public string? KPP { get; set; }
        public string? PAcc { get; set; }
        public string? CAcc { get; set; }
        public string? Bank { get; set; }
    }
}

```

Класс обработчик команды заполнения данных о компании

FillCompanyDataCommandHandler:

```

using HRM.Application.Interfaces;
using System.Text.RegularExpressions;

namespace HRM.Application.CompanyData
{
    public class FillCompanyDataCommandHandler
    {
        private IUnitOfWork unitOfWork;
        public FillCompanyDataCommandHandler(IUnitOfWork unitOfWork)
    }
}

```

```

        {
            this.unitOfWork = unitOfWork;
        }
        public async Task AddOrUpdate(FillCompanyDataCommand
request)
        {
            var data = new Domain.CompanyData();
            Regex reg = new Regex(@"^[0-9]+$");

            if (unitOfWork.CompanyData.GetAllAsync().Result.Count
!= 0)
                data =
unitOfWork.CompanyData.GetAllAsync().Result.FirstOrDefault();
            else
                if (request.CompanyName != null &&
request.CompanyName.Any())
                    data.CompanyName = request.CompanyName;

                if (request.CompanyAddress != null &&
request.CompanyAddress.Any())
                    data.CompanyAddress = request.CompanyAddress;
                if (request.DirectorName != null &&
request.DirectorName.Any())
                    data.DirectorName = request.DirectorName;
                if (request.Bank != null && request.Bank.Any())
                    data.Bank = request.Bank;

                if (request.INN != null && reg.IsMatch(request.INN))
                    data.INN = request.INN;
                if (request.KPP != null && reg.IsMatch(request.KPP))
                    data.KPP = request.KPP;
                if (request.BIK != null && reg.IsMatch(request.BIK))
                    data.BIK = request.BIK;
                if (request.CAcc != null && reg.IsMatch(request.CAcc))
                    data.CAcc = request.CAcc;
                if (request.PAcc != null && reg.IsMatch(request.PAcc))
                    data.PAcc = request.PAcc;

            if (unitOfWork.CompanyData.GetAllAsync().Result.Count
== 0)
                await unitOfWork.CompanyData.CreateAsync(data);
            else
                await unitOfWork.CompanyData.UpdateAsync(data);
            await unitOfWork.Save();
        }
    }
}

```

## Модуль увольнения сотрудников

### Класс команды увольнения сотрудника DismissingCommand:

```
namespace HRM.Application.Dismissing
{
    public class DismissingCommand
    {
        public int EmployeeId { get; set; }
        public string Reason { get; set; }
        public double Payments { get; set; }
    }
}
```

Класс      валидатора      команды      увольнения      сотрудника

### DismissingCommandValidator:

```
using FluentValidation;

namespace HRM.Application.Dismissing
{
    public class DismissalCommandValidators :
AbstractValidator<DismissingCommand>
    {
        public DismissalCommandValidators()
        {
            RuleFor(c => c.Payments).GreaterThanOrEqualTo(0);
        }
    }
}
```

Класс      обработчик      команды      увольнения      сотрудника

### DismissingCommandHandler:

```
using HRM.Application.Interfaces;
using HRM.Domain;
using Quartz;
using Quartz.Impl;

namespace HRM.Application.Dismissing
{
    public class DismissingCommandHandler : IJob
    {
        private IUnitOfWork unitOfWork;
        int EmployeeId;
        public DismissingCommandHandler(IUnitOfWork unitOfWork)
        {
            this.unitOfWork = unitOfWork;
        }
        public async Task Dismiss(DismissingCommand request)
```



```

        {
            EmployeeId = request.EmployeeId;
            var PassportId =
unitOfWork.Employee.GetByIdAsync(EmployeeId).Result.PassportId;
            await unitOfWork.Dismissal.CreateAsync(
                new Dismissal
                {
                    PassportId = PassportId,
                    DocumentDate = DateTime.Now,
                    DismissalDate = DateTime.Now.AddDays(14),
                    DismissalReason = request.Reason,
                    Payments = request.Payments
                });
            if (request.Reason == "По собственному желанию")
            {
                IScheduler scheduler = await
StdSchedulerFactory.GetDefaultScheduler();
                await scheduler.Start();

                IJobDetail job =
JobBuilder.Create<DismissingCommandHandler>().Build();

                ITrigger trigger = TriggerBuilder.Create() //
создаем триггер
                    .WithIdentity("trigger1", "group1") //
идентифицируем триггер с именем и группой
                    .StartAt(DateBuilder.FutureDate(14,
IntervalUnit.Day)) // запуск в конкретную дату 14 дней
                    .Build(); //
создаем триггер

                await scheduler.ScheduleJob(job, trigger); //
начинаем выполнение работы
            }
            else
            {
                await Execute(null);
            }
            public async Task Execute(IJobExecutionContext? context)
            {
                await
unitOfWork.Employee.DeleteAsync(unitOfWork.Employee.GetByIdAsync(Emplo
yeeId).Result);
                await unitOfWork.Save();
            }
        }
    }
}

```

Модуль приема кандидатов на работу

InterviewingCommand:

```
using System.ComponentModel.DataAnnotations;

namespace HRM.Application.Interviewing
{
    public class InterviewingCommand
    {
        [Required]
        public int CandidateId { get; set; }
        [Required]
        public bool IsPassed { get; set; }
        [Required]
        public DateTime PassDate { get; set; }
        public int DepartmentId { get; set; }
    }
}
```

Класс обработчика команды прохождения кандидатом собеседования

InterviewingCommandHandler:

```
using HRM.Application.Interfaces;
using HRM.Domain;

namespace HRM.Application.Interviewing
{
    public class InterviewingCommandHandler
    {
        private readonly IUnitOfWork unitOfWork;
        public InterviewingCommandHandler(IUnitOfWork unitOfWork)
        {
            this.unitOfWork = unitOfWork;
        }
        public async Task TakeInterview(InterviewingCommand request)
        {
            var candidate = await unitOfWork.Candidate.GetByIdAsync(request.CandidateId);
            if (unitOfWork.Candidate.IsInterviewed(candidate.CandidateId))
                throw new Exception(); //Already interviewed
            var current = new Interview()
            {
                CandidateId = candidate.CandidateId,
                Date = request.PassDate,
                IsPassed = request.IsPassed
            };
            await unitOfWork.Interview.CreateAsync(current);
            if(current.IsPassed)
```

```

        {
            await unitOfWork.Employee.CreateAsync(new Employee
            {
                PassportId = candidate.PassportId,
                DepartmentId = request.DepartmentId,
                ContactDataId = candidate.ContactDataId,
                InterviewId = current.InterviewId,
                Active = true,
                AuthorizationCode = Guid.NewGuid().ToString()
            });
        }
        await unitOfWork.Save();
    }
}

```

Модуль заполнения достижений сотрудников

Класс команды заполнения достижений сотрудников

AddOrUpdatePersonalAchievementCommand:

```

namespace HRM.Application.AchievementsHandling
{
    public class AddOrUpdatePersonalAchievementCommand
    {
        public int AchievementId { get; set; }
        public int EmployeeId { get; set; }
        public int PeriodId { get; set; }
        public string Description { get; set; }
        public double Reward { get; set; }
    }
}

```

Класс обработчика команды заполнения достижений сотрудников

AddOrUpdatePersonalAchievementCommandHandler:

```

using HRM.Application.Interfaces;
using HRM.Domain;

namespace HRM.Application.AchievementsHandling
{
    public class AddOrUpdatePersonalAchievementCommandHandler
    {
        private readonly IUnitOfWork unitOfWork;

        public
AddOrUpdatePersonalAchievementCommandHandler(IUnitOfWork unitOfWork)
        {
            this.unitOfWork = unitOfWork;
        }
    }
}

```

```

        public async Task
AddOrUpdate(AddOrUpdatePersonalAchievementCommand request)
    {
        var achiev = new PersonalAchievement()
        {
            AchievementId = request.AchievementId,
            Description = request.Description,
            EmployeeId = request.EmployeeId,
            PeriodId = request.PeriodId,
            Reward = request.Reward
        };
        await unitOfWork.Achievement.UpdateAsync(achiev);
        await unitOfWork.Save();
    }
}

```

## Модуль распределения заработной платы

Класс отложенной обработки распределения заработной платы

MonthResultSalaryHandler:

```

using HRM.Application.Interfaces;
using Quartz;

namespace HRM.Application.Salary
{
    public class MonthResultSalaryHandler : IJob
    {
        private readonly IUnitOfWork unitOfWork;

        public MonthResultSalaryHandler(IUnitOfWork unitOfWork)
        {
            this.unitOfWork = unitOfWork;
        }

        public async Task Execute(IJobExecutionContext context)
        {
            var periodId =
unitOfWork.Period.LastAsync().Result.PeriodId;
            foreach (var dep in
unitOfWork.DepartmentWorkLoad.GetByPeriodId(periodId))
            {
                var moneyPerHour =
dep.Department.TotalMoneyPerHour;
                foreach (var employee in
unitOfWork.EmployeeWorkLoad.GetByPeriodIdPerDepartment(periodId,
dep.DepartmentId))
                {

```

```

        employee.ResultSalary = employee.WorkedHours
<=      employee.WorkLoadHours      ?      employee.WorkedHours      :
employee.WorkLoadHours;
        employee.ResultSalary *= moneyPerHour;
    }
}
}
}
}

```

Модуль распределения рабочей нагрузки

Класс базовой команды распределения рабочей нагрузки

DistributionCommand:

```

namespace HRM.Application.WorkLoadDistribution
{
    public class DistributionCommand
    {
        public int MonthlyHours { get; set; }
        public List<DistributionOption>? Options { get; set; }
    }
}

```

Класс варианта распределения рабочей нагрузки DistributionOption:

```

namespace HRM.Application.WorkLoadDistribution
{
    public class DistributionOption
    {
        public int DepartmentId { get; set; }
        public string? DepartmentTitle { get; set; }
        public int StaticHours { get; set; }
    }
}

```

Класс валидатора базовой команды распределения рабочей нагрузки

CommandValidation:

```

using HRM.Application.Interfaces;
using HRM.Domain;

namespace HRM.Application.WorkLoadDistribution
{
    public class CommandValidation
    {
        private IUnitOfWork unitOfWork;
        public CommandValidation(IUnitOfWork unitOfWork)
        {
            this.unitOfWork = unitOfWork;
        }
    }
}

```

```

    }
    public bool ValidateOptions(List<DistributionOption>
options, Period period)
    {
        if (options == null)
            return true;
        foreach (var option in options)
        {
            if (!ValidateOption(option, period))
                return false;
        }
        return true;
    }
    private bool ValidateOption(DistributionOption option,
Period period)
    {
        if
            (option.StaticHours
            >=
unitOfWork.Department.GetByIdAsync(option.DepartmentId).Result.Employee
eCount * 4 * WorkDays(period) &&
            option.StaticHours
            <=
unitOfWork.Department.GetByIdAsync(option.DepartmentId).Result.Employee
eCount * 10 * WorkDays(period))
            return true;
        else
            return false;
    }
    public bool ValidateStatic(int hours, Period period)
    {
        if (hours >= unitOfWork.Employee.GetActive().Count() *
4 * WorkDays(period) &&
            hours <= unitOfWork.Employee.GetActive().Count() *
10 * WorkDays(period))
            return true;
        else
            return false;
    }
    private int WorkDays(Period period)
    {
        int workDays = DateTime.DaysInMonth(period.Year,
period.Month);
        for (int i = 1; i < DateTime.DaysInMonth(period.Year,
period.Month) - 1; i++)
        {
            DateTime dt = new DateTime(period.Year,
period.Month, i);
            if (dt.DayOfWeek == DayOfWeek.Saturday ||
dt.DayOfWeek == DayOfWeek.Sunday)
                workDays--;
        }
        return workDays;
    }
}

```

```

    }
}

```

Класс логических операций при распределении рабочей нагрузки  
DistributionLogic:

```

using HRM.Application.Interfaces;
using HRM.Domain;

namespace HRM.Application.WorkLoadDistribution
{
    public class DistributionLogic
    {
        private IUnitOfWork unitOfWork;
        public DistributionLogic(IUnitOfWork unitOfWork)
        {
            this.unitOfWork = unitOfWork;
        }
        public int DistributedHours(int hours, int departmentId)
        {
            return (int)Math.Round((double)hours /
unitOfWork.Employee.GetActiveByDepartmentId(departmentId).Count());
        }
        public int DistributedHours(int hours)
        {
            return (int)Math.Round((double)hours /
unitOfWork.Employee.GetActive().Count());
        }
        public async Task FinalSum(Period current)
        {
            current.TotalWorkLoadHours =
unitOfWork.EmployeeWorkLoad.SumPeriodWorkLoadHours(current.PeriodId);
            await unitOfWork.Period.UpdateAsync(current);
        }
    }
}

```

Класс команды создания нового распределения рабочей нагрузки  
CreateDistributionCommand:

```

namespace HRM.Application.WorkLoadDistribution.CreateDistribution
{
    public class CreateDistributionCommand : DistributionCommand
    {
    }
}

```

Класс обработчика команды создания нового распределения рабочей  
нагрузки CreateDistributionCommandHandler:

```

using HRM.Application.Exceptions;
using HRM.Application.Interfaces;
using HRM.Application.WorkLoadDistribution.GenerateAddendum;
using HRM.Domain;

namespace HRM.Application.WorkLoadDistribution.CreateDistribution
{
    public class CreateDistributionCommandHandler
    {
        private IUnitOfWork unitOfWork;
        private CommandValidation validation;
        private DistributionLogic logic;
        private GenerateAddendumCommandHandler handler;
        Period current;
        public CreateDistributionCommandHandler(IUnitOfWork
unitOfWork)
        {
            this.unitOfWork = unitOfWork;
            validation = new CommandValidation(unitOfWork);
            logic = new DistributionLogic(unitOfWork);
            handler = new
GenerateAddendumCommandHandler(unitOfWork);
            current = unitOfWork.Period.Next();
        }
        public async Task Distribute(CreateDistributionCommand
request)
        {
            await unitOfWork.Period.CreateAsync(current);
            current.TotalWorkLoadHours = request.MonthlyHours;

            if (request.Options == null)
            {
                if
(validation.ValidateStatic(request.MonthlyHours, current))
                    await StaticDistribute(request);
                else
                    throw new
ValidationException(nameof(CreateDistributionCommand));
            }
            else
            {
                if(validation.ValidateOptions(request.Options,
current))
                    await DinamicDistribute(request);
                else
                    throw new
ValidationException(nameof(CreateDistributionCommand));
            }
        }
        private async Task
StaticDistribute(CreateDistributionCommand request)

```



```

        {
            int hoursPerEmployee =
logic.DistributedHours(request.MonthlyHours);
            foreach (var employee in
unitOfWork.Employee.GetActive())
            {
                employee.EmployeeWorkLoads.Add(new
EmployeeWorkLoad
                {
                    EmployeeId = employee.EmployeeId,
                    PeriodId = current.PeriodId,
                    WorkLoadHours = hoursPerEmployee,
                });
            }
            await unitOfWork.Save();
            foreach (var department in
unitOfWork.Department.GetAllAsync().Result)
            {
                department.DepartmentWorkLoads.Add(new
DepartmentWorkLoad
                {
                    DepartmentId = department.DepartmentId,
                    PeriodId = current.PeriodId,
                    WorkLoadHours =
unitOfWork.EmployeeWorkLoad.SumPeriodWorkLoadHoursByDepartmentId(curre
nt.PeriodId, department.DepartmentId),
                });
            }
            await logic.FinalSum(current);
            await unitOfWork.Save();
        }
        private async Task
DynamicDistribute(CreateDistributionCommand request)
        {
            foreach (var option in request.Options)
            {
                int hoursPerDepartmentEmployee =
logic.DistributedHours(option.StaticHours, option.DepartmentId);

                foreach (var employee in
unitOfWork.Employee.GetActiveByDepartmentId(option.DepartmentId))
                {
                    employee.EmployeeWorkLoads.Add(new
EmployeeWorkLoad
                    {
                        EmployeeId = employee.EmployeeId,
                        PeriodId = current.PeriodId,
                        WorkLoadHours =
hoursPerDepartmentEmployee,
                    });
                }
            }
        }
    }

```

```

        await unitOfWork.Save();
        Department dep = await
unitOfWork.Department.GetByIdAsync(option.DepartmentId);
        var wl = new DepartmentWorkLoad
        {
            DepartmentId = dep.DepartmentId,
            PeriodId = current.PeriodId,
            WorkLoad =
unitOfWork.EmployeeWorkLoad.SumPeriodWorkLoadHoursByDepartmentId(curre
nt.PeriodId, dep.DepartmentId),
        };
        dep.DepartmentWorkLoads.Add(wl);
        if (wl.WorkLoad != option.StaticHours) ;
        //messege округлено
    }
    await logic.FinalSum(current);
    await unitOfWork.Save();
}
}
}

```

Класс команды обновления существующего распределения рабочей нагрузки UpdateDistributionCommand:

```

namespace HRM.Application.WorkLoadDistribution.UpdateDistribution
{
    public class UpdateDistributionCommand : DistributionCommand
    {
        public int PeriodId { get; set; }
    }
}

```

Класс обработчика команды обновления существующего распределения рабочей нагрузки UpdateDistributionCommandHandler:

```

using HRM.Application.Exceptions;
using HRM.Application.Interfaces;
using HRM.Domain;

namespace HRM.Application.WorkLoadDistribution.UpdateDistribution
{
    public class UpdateDistributionCommandHandler
    {
        private IUnitOfWork unitOfWork;
        private CommandValidation validation;
        private DistributionLogic logic;
        private Period current;
        public UpdateDistributionCommandHandler(IUnitOfWork
unitOfWork)
    {

```

```

        this.unitOfWork = unitOfWork;
        validation = new CommandValidation(unitOfWork);
        logic = new DistributionLogic(unitOfWork);
    }

    public async Task
    UpdateDistribution(UpdateDistributionCommand request)
    {
        current =
        unitOfWork.Period.GetByIdAsync(request.PeriodId).Result;
        if (current == null)
            throw new Exception(); // not found exception

        if (request.Options == null)
        {
            if
            (validation.ValidateStatic(request.MonthlyHours, current))
                await StaticUpdate(request);
            else
                throw new
                ValidationException(nameof(UpdateDistributionCommand));
        }
        else
        {
            if(validation.ValidateOptions(request.Options,
            current))
                await DinamicUpdate(request);
            else
                throw new
                ValidationException(nameof(UpdateDistributionCommand));
        }
    }

    private async Task DinamicUpdate(UpdateDistributionCommand
    request)
    {
        foreach (var option in request.Options)
        {
            int hoursPerDepartmentEmployee =
            logic.DistributedHours(option.StaticHours, option.DepartmentId);

            foreach (var workload in
            unitOfWork.EmployeeWorkLoad.GetByPeriodIdPerDepartment(request.PeriodI
            d, option.DepartmentId))
            {
                if (workload.WorkedHours != null &&
                hoursPerDepartmentEmployee <= workload.WorkedHours)
                    workload.WorkLoadHours =
                    (int)workload.WorkedHours;
                else
            }
        }
    }

```

```

                                workload.WorkLoadHours
hoursPerDepartmentEmployee;

                                await
unitOfWork.EmployeeWorkLoad.UpdateAsync(workload);
                                }
                                foreach (var employee in
unitOfWork.Employee.GetActiveWithNoWorkLoadByPeriodIdPerDepartment(request.PeriodId, option.DepartmentId)) //добавление в план новых
сотрудников
                                {
                                EmployeeWorkLoad
                                employee.EmployeeWorkLoads.Add(new
                                {
                                EmployeeId = employee.EmployeeId,
                                PeriodId = current.PeriodId,
                                WorkLoadHours
                                hoursPerDepartmentEmployee,
                                });
                                }
                                await unitOfWork.Save();

                                var schedule
unitOfWork.DepartmentWorkLoad.GetByPeriodIdPerDepartment(request.PeriodId, option.DepartmentId);
                                schedule.WorkLoad
unitOfWork.EmployeeWorkLoad.SumPeriodWorkLoadHoursByDepartmentId(current.PeriodId, schedule.DepartmentId);
                                await
unitOfWork.DepartmentWorkLoad.UpdateAsync(schedule);
                                if (schedule.WorkLoad != option.StaticHours) ;
                                //messege округлено
                                }
                                await logic.FinalSum(current);
                                await unitOfWork.Save();
                                }

                                private async Task StaticUpdate(UpdateDistributionCommand
request)
                                {
                                int hoursPerEmployee
                                logic.DistributedHours(request.MonthlyHours);
                                foreach(var workload in
unitOfWork.EmployeeWorkLoad.GetByPeriodId(request.PeriodId))
                                {
                                if (workload.WorkedHours != null &&
hoursPerEmployee <= workload.WorkedHours)
                                workload.WorkLoadHours
                                (int)workload.WorkedHours;
                                else
                                workload.WorkLoadHours = hoursPerEmployee;

```

```

        await
unitOfWork.EmployeeWorkLoad.UpdateAsync(workload);
    }
    foreach (var employee in
unitOfWork.Employee.GetActiveWithNoWorkLoadByPeriodId(request.PeriodId
)) //добавление в план новых сотрудников
    {
        employee.EmployeeWorkLoads.Add(new
EmployeeWorkLoad
        {
            EmployeeId = employee.EmployeeId,
            PeriodId = current.PeriodId,
            WorkLoadHours = hoursPerEmployee,
        });
    }
    await unitOfWork.Save();
    foreach (var departmentWorkLoad in
unitOfWork.DepartmentWorkLoad.GetByPeriodId(request.PeriodId))
    {
        departmentWorkLoad.WorkLoad =
unitOfWork.EmployeeWorkLoad.SumPeriodWorkLoadHoursByDepartmentId(curre
nt.PeriodId, departmentWorkLoad.DepartmentId);
        await
unitOfWork.DepartmentWorkLoad.UpdateAsync(departmentWorkLoad);
    }
    await logic.FinalSum(current);
    await unitOfWork.Save();
}
}
}
}

```

Класс обработчика вывода результатов распределения рабочей нагрузки

ReadDistributionCommandHandler:

```

using HRM.Application.Interfaces;
using HRM.Domain;

namespace HRM.Application.WorkLoadDistribution.ReadDistribution
{
    public class ReadDistributionCommandHandler
    {
        private IUnitOfWork unitOfWork;
        Period current;
        public ReadDistributionCommandHandler(IUnitOfWork
unitOfWork)
        {
            this.unitOfWork = unitOfWork;
        }
    }
}

```

```

        public async Task<List<EmployeeWorkLoad>>
ReadEmployeeWorkLoads(DateTime request)
        {
            current = unitOfWork.Period.GetPeriodByDate(request);
            var Workloads =
unitOfWork.EmployeeWorkLoad.GetByPeriodId(current.PeriodId);
            return Workloads.ToList();
        }
        public async Task<List<DepartmentWorkLoad>>
ReadDepartmentWorkLoads(DateTime request)
        {
            current = unitOfWork.Period.GetPeriodByDate(request);
            var Workloads =
unitOfWork.DepartmentWorkLoad.GetByPeriodId(current.PeriodId);
            return Workloads.ToList();
        }
    }
}

```

Модуль генерации дополнительных соглашений

Класс команды генерации дополнительных соглашений

GenerateAddendumCommand:

```

namespace HRM.Application.WorkLoadDistribution.GenerateAddendum
{
    public class GenerateAddendumCommand
    {
        public int EmployeeId { get; set; }
        public int WorkLoad { get; set; }
        public int PeriodId { get; set; }
    }
}

```

Класс обработчика команды генерации дополнительных соглашений

GenerateAddendumCommandHandler:

```

using HRM.Application.Interfaces;
using HRM.Domain;
using DocumentFormat.OpenXml.Packaging;
using System.Text.RegularExpressions;

namespace HRM.Application.WorkLoadDistribution.GenerateAddendum
{
    public class GenerateAddendumCommandHandler
    {
        private IUnitOfWork unitOfWork;
        public GenerateAddendumCommandHandler(IUnitOfWork
unitOfWork)
        {

```

```

        this.unitOfWork = unitOfWork;
    }
    public async Task GenerateAddendum(GenerateAddendumCommand
request)
    {
        Period            currentPeriod            =            await
unitOfWork.Period.GetByIdAsync(request.PeriodId);
        Employee            currentEmployee            =            await
unitOfWork.Employee.GetByIdAsync(request.EmployeeId);
        Department            currentDepartment            =            await
unitOfWork.Department.GetByIdAsync((int)currentEmployee.DepartmentId);
        Domain.CompanyData            values            =            await
unitOfWork.CompanyData.FirstAsync();

        byte[] prefabData = null;
        if (unitOfWork.File.GetByIdAsync(1).Result != null)
            prefabData =
unitOfWork.File.GetByIdAsync(1).Result.Data;

        string fileName = currentEmployee.Passport.Surname + "
" + unitOfWork.Period.GetDateFromPeriod(currentPeriod).ToString("Y") +
".docx";
        string path = @"wwwroot/files/" + fileName;

        using (var stream = new FileStream(path,
FileMode.Create, FileAccess.Write))
        {
            await stream.WriteAsync(prefabData, 0,
prefabData.Length);
        }

        using (WordprocessingDocument wordDoc =
WordprocessingDocument.Open(path, true))
        {
            string docText = null;
            using (StreamReader sr = new
StreamReader(wordDoc.MainDocumentPart.GetStream()))
            {
                docText = sr.ReadToEnd();
            }
            Regex regexInterview = new
Regex(@"InterviewDate");
            Regex regexPeriod = new Regex(@"PeriodDate");
            Regex regexCName = new Regex(@"CompanyName");
            Regex regexDName = new Regex(@"DirectorName");
            Regex regexFullName = new Regex(@"FullName");
            Regex regexDepartment = new Regex(@"Department");
            Regex regexWorkLoad = new Regex(@"WorkLoad");
            Regex regexMPH = new Regex(@"MoneyPerHour");
            Regex regexCAddress = new
Regex(@"CompanyAddress");

```

```

        Regex regexINN = new Regex(@"INN");
        Regex regexKPP = new Regex(@"KPP");
        Regex regexBIK = new Regex(@"BIK");
        Regex regexPAcc = new Regex(@"PAcc");
        Regex regexCAcc = new Regex(@"CAcc");
        Regex regexBank = new Regex(@"Bank");
        Regex regexPSer = new Regex(@"PSerial");
        Regex regexPNum = new Regex(@"PNumber");
        Regex regexInit = new Regex(@"InitName");
        Regex regexAddress = new Regex(@"Address");

        docText = regexInterview.Replace(docText,
currentEmployee.Interview.Date.ToString("d"));
        docText = regexPeriod.Replace(docText,
unitOfWork.Period.GetDateFromPeriod(currentPeriod).ToString("Y"));
        docText = regexCName.Replace(docText,
values.CompanyName);
        docText = regexDName.Replace(docText,
values.DirectorName);
        docText = regexFullName.Replace(docText,
unitOfWork.Employee.GetFullName(currentEmployee));
        docText = regexDepartment.Replace(docText,
currentDepartment.Direction);
        docText = regexWorkLoad.Replace(docText,
request.WorkLoad.ToString());
        docText = regexMPH.Replace(docText,
currentDepartment.TotalMoneyPerHour.ToString());
        docText = regexCAddress.Replace(docText,
values.CompanyAddress);
        docText = regexINN.Replace(docText, values.INN);
        docText = regexKPP.Replace(docText, values.KPP);
        docText = regexBIK.Replace(docText, values.BIK);
        docText = regexPAcc.Replace(docText, values.PAcc);
        docText = regexCAcc.Replace(docText, values.CAcc);
        docText = regexBank.Replace(docText, values.Bank);
        docText = regexPSer.Replace(docText,
currentEmployee.Passport.PassportSerial.ToString());
        docText = regexPNum.Replace(docText,
currentEmployee.Passport.PassportNumber.ToString());
        docText = regexInit.Replace(docText,
unitOfWork.Employee.GetInits(currentEmployee));
        docText = regexAddress.Replace(docText,
unitOfWork.Employee.GetAddress(currentEmployee));

        using (StreamWriter sw = new
StreamWriter(wordDoc.MainDocumentPart.GetStream(FileMode.Create)))
        {
            sw.Write(docText);
        }
    }
}

```



```

        var data = await
System.IO.File.ReadAllBytesAsync(path);

        await unitOfWork.File.CreateAsync(
new Domain.File
{
    Name = fileName,
    PeriodId = currentPeriod.PeriodId,
    EmployeeId = currentEmployee.EmployeeId,
    DepartmentId = currentEmployee.DepartmentId,
    Data = data
});
    }
}
}

```

Класс DependencyInjection:

```

using FluentValidation;
using Microsoft.Extensions.DependencyInjection;
using System.Reflection;
using Quartz;
using HRM.Application.Salary;

namespace HRM.Application
{
    public static class DependencyInjection
    {
        public static IServiceCollection AddApplication(this
IServiceCollection services)
        {
            services.AddValidatorsFromAssemblies(new[] {
Assembly.GetExecutingAssembly() });
            services.AddQuartz(q =>
            {
                q.UseMicrosoftDependencyInjectionJobFactory();

                q.AddJob<MonthResultSalaryHandler>(opt =>
opt.WithIdentity("summriize"))
                .AddTrigger(opts => opts
                .ForJob("summriize")
                .WithIdentity("summriizeTrigger")

                .WithSchedule(CronScheduleBuilder.MonthlyOnDayAndHourAndMinute(1, 6, 0)
                .InTimeZone(TimeZoneInfo.Local))
                );
            });
            //services.AddTransient(typeof(IPipelineBehavior<,>),
typeof(ValidationBehavior<,>));
            return services;
        }
    }
}

```

} }

## Приложение 4. Код уровня Persistence

Реализация интерфейсов репозитория

Класс RepositoryBase:

```
using HRM.Application.BuisnessLogic.Base;
using Microsoft.EntityFrameworkCore;

namespace HRM.Persistence.Repositories.Base
{
    public class RepositoryBase<T> : IRepositoryBase<T> where T :
class
    {
        internal readonly HRMdbContext context;

        public RepositoryBase(HRMdbContext context)
        {
            this.context = context;
        }
        public async Task<T> CreateAsync(T entity)
        {
            await context.Set<T>().AddAsync(entity);
            await context.SaveChangesAsync();
            return entity;
        }

        public async Task DeleteAsync(T entity)
        {
            context.Set<T>().Remove(entity);
            //await context.SaveChangesAsync();
        }

        public virtual async Task<List<T>> GetAllAsync()
        {
            return await context.Set<T>().ToListAsync();
        }

        public virtual async Task<T?> GetByIdAsync(int id)
        {
            return await context.Set<T>().FindAsync(id);
        }

        public virtual async Task<T?> FirstAsync()
        {
            return await context.Set<T>().FirstOrDefaultAsync();
        }

        public async Task UpdateAsync(T entity)
        {

```

```

        if (context.Set<T>().Contains(entity))
        {
            context.Set<T>().Update(entity);
        }
        else
        {
            await context.Set<T>().AddAsync(entity);
        }
        //await context.SaveChangesAsync();
    }

    public virtual async Task<T?> LastAsync()
    {
        return await context.Set<T>().LastOrDefaultAsync();
    }
}

```

Класс AchievementRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistence.Repositories.Base;

namespace HRM.Persistence.Repositories
{
    public class AchievementRepository :
RepositoryBase<PersonalAchievement>, I AchievementRepository
    {
        public AchievementRepository(HRMDBContext context) :
base(context)
        {
        }
    }
}

```

Класс AuthorizationRepository:

```

using HRM.Domain;
using HRM.Application.BuisnessLogic;
using HRM.Persistence.Repositories.Base;

namespace HRM.Persistence.Repositories
{
    public class AuthorizationRepository :
RepositoryBase<Authorization>, IAuthorizationRepository
    {
        public AuthorizationRepository(HRMDBContext context) :
base(context)
        {
        }
    }
}

```

```

    }

    public bool IsUsed(string username)
    {
        foreach(var auth in context.Authorizations)
        {
            if(auth.Username == username)
                return true;
        }
        return false;
    }
}
}

```

Класс CandidateRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistence.Repositories.Base;
using Microsoft.EntityFrameworkCore;

namespace HRM.Persistence.Repositories
{
    public class CandidateRepository : RepositoryBase<Candidate>,
ICandidateRepository
    {
        public CandidateRepository(HRMDBContext context) :
base(context)
        {
        }
        public async Task<List<Candidate>> GetInterviewed()
        {
            return await context.Interviews.Select(x=>x.Candidate)
                .OrderBy(x => x.Passport.Surname)
                .AsNoTracking()
                .ToListAsync();
        }

        public async Task<List<Candidate>> GetNotInterviewed()
        {
            return await context.Candidates
                .OrderBy(x => x.Passport.Surname)
                .Where(x => !(x.Interviews.Any()))
                .AsNoTracking()
                .ToListAsync();
        }

        public bool IsInterviewed(int candidateId)
        {
            var candidate = context.Candidates.Find(candidateId);

```

```

        if (GetInterviewed().Result.Where(x=>x.CandidateId ==
candidateId).Any())
            return true;
        else
            return false;
    }
}

```

Класс CompanyDataRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistence.Repositories.Base;

namespace HRM.Persistence.Repositories
{
    public class CompanyDataRepository :
RepositoryBase<CompanyData>, ICompanyDataRepository
    {
        public CompanyDataRepository(HRMDBContext context) :
base(context)
        {
        }
    }
}

```

Класс ContactDataRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistence.Repositories.Base;
using Microsoft.EntityFrameworkCore;

namespace HRM.Persistence.Repositories
{
    public class ContactDataRepository :
RepositoryBase<ContactData>, IContactDataRepository
    {
        public ContactDataRepository(HRMDBContext context) :
base(context)
        {
        }
        public IEnumerable<ContactData> GetAllWithMissing()
        {
            return context.ContactData.Where(a =>
a.Employees.Any()).Where(x => x.PhoneNumber == null || x.Email ==
null).AsNoTracking();
        }
    }
}

```

```

        public                                     IEnumerable<ContactData>
        GetAllWithMissingByDepartment(int id)
        {
            return context.ContactData
                .Where(a => a.Employees.Any())
                .Where(i                                     =>
i.Employees.FirstOrDefault().DepartmentId == id)
                .Where(x => x.PhoneNumber == null || x.Email ==
null)
                .AsNoTracking();
        }
    }
}

```

Класс DepartmentRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistence.Repositories.Base;
using Microsoft.EntityFrameworkCore;

namespace HRM.Persistence.Repositories
{
    public class DepartmentRepository :
RepositoryBase<Department>, IDepartmentRepository
    {
        public DepartmentRepository(HRMDBContext context) :
base(context)
        {
        }

        public bool IsEmployeeManager(int employeeId)
        {
            foreach(var dep in context.Departments.ToList())
            {
                if(dep.ManagerId == employeeId)
                    return true;
            }
            return false;
        }
    }
}

```

Класс DepartmentWorkLoadRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistence.Repositories.Base;

namespace HRM.Persistence.Repositories
{

```

```

        public class DepartmentWorkLoadRepository :
RepositoryBase<DepartmentWorkLoad>, IDepartmentWorkLoadRepository
    {
        public DepartmentWorkLoadRepository(HRMDBContext context)
: base(context)
        {
        }

        public IEnumerable<DepartmentWorkLoad> GetByPeriodId(int
periodId)
        {
            return context.DepartmentWorkLoads
                .Where(x=>x.PeriodId == periodId);
        }

        public DepartmentWorkLoad GetByPeriodIdPerDepartment(int
periodId, int departmentId)
        {
            return context.DepartmentWorkLoads
                .Where(x => x.PeriodId == periodId &&
x.DepartmentId == departmentId)
                .FirstOrDefault();
        }

        public
        IEnumerable<DepartmentWorkLoad>
GetWithNotEnoughHours()
        {
            return context.DepartmentWorkLoads
                .Where(x => x.IsEqualOrMore == true);
        }
    }
}

```

Класс DismissalRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistence.Repositories.Base;

namespace HRM.Persistence.Repositories
{
    public class DismissalRepository : RepositoryBase<Dismissal>,
IDismissalRepository
    {
        public DismissalRepository(HRMDBContext context) :
base(context)
        {
        }
    }
}

```



Класс EmployeeRepository:

```
using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistance.Repositories.Base;
using Microsoft.EntityFrameworkCore;

namespace HRM.Persistance.Repositories
{
    public class EmployeeRepository : RepositoryBase<Employee>,
IEmployeeRepository
    {
        public EmployeeRepository(HRMDBContext context) :
base(context)
        {
        }

        public IEnumerable<Employee> GetActive()
        {
            return context.Employees
                .Where(x=>x.Active == true);
        }

        public IEnumerable<Employee> GetActiveByDepartmentId(int
id)
        {
            return context.Employees
                .Where(x => x.Active == true && x.DepartmentId ==
id);
        }

        public IEnumerable<Employee>
GetActiveWithNoWorkLoadByPeriodId(int periodId)
        {
            return context.Employees
                .Where(x => !x.EmployeeWorkLoads.Where(y =>
y.PeriodId == periodId)
                .Any());
        }

        public IEnumerable<Employee>
GetActiveWithNoWorkLoadByPeriodIdPerDepartment(int periodId, int
departmentId)
        {
            return context.Employees
                .Where(x => x.DepartmentId == departmentId &&
!x.EmployeeWorkLoads.Where(y => y.PeriodId == periodId)
                .Any());
        }
    }
}
```

```

public string GetAddress(Employee employee)
{
    return string.Join(", ",
        new string[]
        {
            employee.Passport.City,
            employee.Passport.Street,
            "д." + employee.Passport.House.ToString(),
            "к." + employee.Passport.Buinding.ToString(),
            "кв." + employee.Passport.Apartment.ToString()
        });
}

public IEnumerable<Employee> GetAuthorizer()
{
    return context.Employees
        .Where(x => x.Authorizations.Any())
        .AsNoTracking();
}

public Employee GetByAuthCode(string authCode)
{
    return context.Employees
        .Where(x=>x.AuthorizationCode == authCode &&
!x.Authorizations.Any())
        .FirstOrDefault();
}

public Employee GetByAuthData(string login, string
password)
{
    return context.Employees
        .Where(x
x.Authorizations.FirstOrDefault().Username == login &&
x.Authorizations.FirstOrDefault().Password == password)
        .FirstOrDefault();
}

public string GetFullName(Employee employee)
{
    return string.Join(" ", new string[]
    {
        employee.Passport.Surname,
        employee.Passport.Name,
        employee.Passport.Lastname == null? "":
employee.Passport.Lastname
    });
}

public string GetInits(Employee employee)
{

```

```

        return string.Join(" ", new string[]{
employee.Passport.Surname,
        string.Join(".", new string[]{
employee.Passport.Name[0].ToString(),
        employee.Passport.Lastname == null ? "":
employee.Passport.Lastname[0].ToString()}) });
    }

    public IEnumerable<Employee> GetUnauthorized()
    {
        return context.Employees
            .Where(x => !x.Authorizations.Any())
            .AsNoTracking();
    }

    public IEnumerable<Employee> GetWithMissingContactData()
    {
        return context.Employees
            .Where(x => x.ContactData.PhoneNumber != null &&
x.ContactData.Email != null)
            .AsNoTracking();
    }
}
}
}

```

Класс EmployeeWorkLoadRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistance.Repositories.Base;
using Microsoft.EntityFrameworkCore;

namespace HRM.Persistance.Repositories
{
    public class EmployeeWorkLoadRepository :
RepositoryBase<EmployeeWorkLoad>, IEmployeeWorkLoadRepository
    {
        public EmployeeWorkLoadRepository(HRMDBContext context) :
base(context)
        {
        }

        public IEnumerable<EmployeeWorkLoad> GetByPeriodId(int
periodId)
        {
            return context.EmployeeWorkLoads
                .Where(x => x.PeriodId == periodId)
                .ToArray();
        }
    }
}

```

```

        public                               IEnumerable<EmployeeWorkLoad>
GetByPeriodIdPerDepartment(int periodId, int departmentId)
    {
        return context.EmployeeWorkLoads
            .Where(x=>x.PeriodId == periodId &&
x.Employee.DepartmentId == departmentId);
    }

    public                               IEnumerable<EmployeeWorkLoad>
GetWithNotEnoughHours()
    {
        return context.EmployeeWorkLoads
            .Where(x => x.WorkLoadHours >= x.WorkLoadHours)
            .AsNoTracking();
    }

    public int SumPeriodWorkLoadHours(int periodId)
    {
        return (int)context.EmployeeWorkLoads
            .Where(x => x.PeriodId == periodId)
            .Sum(x => x.WorkLoadHours);
    }

    public int SumPeriodWorkLoadHoursByDepartmentId(int
periodId, int departmentId)
    {
        return (int)context.EmployeeWorkLoads
            .Where(x =>
                x.Employee.DepartmentId == departmentId &&
                x.PeriodId == periodId)
            .Sum(x => x.WorkLoadHours);
    }
}
}

```

Класс FileRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Persistence.Repositories.Base;
using Microsoft.EntityFrameworkCore;

namespace HRM.Persistence.Repositories
{
    public class FileRepository : RepositoryBase<Domain.File>,
IFileRepository
    {
        public FileRepository(HRMDBContext context) :
base(context)
        {

```

```

    }

    public Task ClearStorage(string path)
    {
        foreach(var file in
System.IO.Directory.GetFiles(path))
        {
            System.IO.File.Delete(file);
        }
        return Task.CompletedTask;
    }

    public IEnumerable<Domain.File> GetAllByEmployeeId(int
employeeId)
    {
        return context.Files.Where(x => x.EmployeeId ==
employeeId).AsNoTracking().ToArray();
    }

    public IEnumerable<Domain.File> GetAllByPeriodId(int
periodId)
    {
        return context.Files.Where(x=>x.PeriodId ==
periodId).AsNoTracking().ToArray();
    }

    public IEnumerable<Domain.File> GetByDepartmentId(int
departmentId, int periodId)
    {
        return context.Files.Where(x=>x.DepartmentId ==
departmentId && x.PeriodId == periodId).AsNoTracking().ToArray();
    }

    public Domain.File GetOne(int employeeId, int periodId)
    {
        return context.Files.Where(x=>x.EmployeeId ==
employeeId && x.PeriodId == periodId).AsNoTracking().FirstOrDefault();
    }
}

```

Класс InterviewRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistence.Repositories.Base;

namespace HRM.Persistence.Repositories
{
    public class InterviewRepository : RepositoryBase<Interview>,
IInterviewRepository

```

```

        {
            public InterviewRepository(HRMDBContext context) :
base(context)
            {
            }
        }
    }
}

```

Класс PeriodRepository:

```

using HRM.Application.BuisnessLogic;
using HRM.Domain;
using HRM.Persistence.Repositories.Base;

namespace HRM.Persistence.Repositories
{
    public class PeriodRepository : RepositoryBase<Period>,
IPeriodRepository
    {
        public PeriodRepository(HRMDBContext context) :
base(context)
        {
        }

        public Period Next()
        {
            if (!context.Periods.Any())
                return CreateFirst();
            Period last = context.Periods.OrderBy(x =>
x.PeriodId).LastOrDefault();
            Period next = new Period()
            {
                Year = last.Month == 12 ? last.Year + 1 : last.Year,
                Month = last.Month == 12 ? 1 : last.Month + 1
            };
            return next;
        }

        public Period CreateFirst()
        {
            Period first = new Period()
            {
                Year = DateTime.Now.Year,
                Month = DateTime.Now.Month
            };
            return first;
        }

        public Period? GetPeriodByDate(DateTime date)
        {

```

```

        return context.Periods.Where(x => x.Year == date.Year
&& x.Month == date.Month).FirstOrDefault();
    }

    public DateTime GetDateFromPeriod(Period period)
    {
        return new DateTime(period.Year, period.Month, 1);
    }
}

```

Класс UnitOfWork:

```

using HRM.Application.BuisnessLogic;
using HRM.Application.Interfaces;
using HRM.Persistence.Repositories;

namespace HRM.Persistence
{
    public class UnitOfWork : IUnitOfWork
    {
        private readonly HRMdbContext context;
        public UnitOfWork(HRMdbContext _context)
        {
            context = _context;
            Authorization = new AuthorizationRepository(context);
            Achievement = new AchievementRepository(context);
            Candidate = new CandidateRepository(context);
            CompanyData = new CompanyDataRepository(context);
            ContactData = new ContactDataRepository(context);
            Department = new DepartmentRepository(context);
            DepartmentWorkLoad = new
DepartmentWorkLoadRepository(context);
            File = new FileRepository(context);
            Employee = new EmployeeRepository(context);
            EmployeeWorkLoad = new
EmployeeWorkLoadRepository(context);
            Period = new PeriodRepository(context);
            Interview = new InterviewRepository(context);
            Dismissal = new DismissalRepository(context);
        }

        public IAchievementRepository Achievement { get; set; }
        public ICompanyDataRepository CompanyData { get; set; }
        public ICandidateRepository Candidate { get; set; }
        public IContactDataRepository ContactData { get; set; }
        public IDepartmentRepository Department { get; set; }
        public IDepartmentWorkLoadRepository DepartmentWorkLoad {
get; set; }
        public IEmployeeRepository Employee { get; set; }
    }
}

```

```

        public IEmployeeWorkLoadRepository EmployeeWorkLoad { get;
set; }

        public IDismissalRepository Dismissal{ get; set; }
        public IFileRepository File { get; set; }
        public IPeriodRepository Period { get; set; }
        public IAuthorizationRepository Authorization { get; set;
}

        public IInterviewRepository Interview { get; set; }

        public async Task Save()
        {
            await context.SaveChangesAsync();
        }
    }
}

```

Класс HRMDBContext:

```

using HRM.Application.Interfaces;
using HRM.Domain;
using Microsoft.EntityFrameworkCore;

namespace HRM.Persistence
{
    public partial class HRMDBContext : DbContext, IHRMDBContext
    {
        public HRMDBContext()
        {
        }

        public HRMDBContext(DbContextOptions<HRMDBContext>
options)
            : base(options)
        {
        }

        public virtual DbSet<Authorization> Authorizations { get;
set; } = null!;
        public virtual DbSet<Candidate> Candidates { get; set; } =
null!;
        public virtual DbSet<ContactData> ContactData { get; set;
} = null!;
        public virtual DbSet<Department> Departments { get; set; }
= null!;
        public virtual DbSet<DepartmentWorkLoad>
DepartmentWorkLoads { get; set; } = null!;
        public virtual DbSet<Dismissal> Dismissals { get; set; } =
null!;
        public virtual DbSet<Document> Documents { get; set; } =
null!;

```



```

        public virtual DbSet<Employee> Employees { get; set; } =
null!;
        public virtual DbSet<EmployeeWorkLoad> EmployeeWorkLoads {
get; set; } = null!;
        public virtual DbSet<Interview> Interviews { get; set; } =
null!;
        public virtual DbSet<PassportInfo> PassportInfos { get;
set; } = null!;
        public virtual DbSet<Period> Periods { get; set; } = null!;
        public virtual DbSet<PersonalAchievement>
PersonalAchievements { get; set; } = null!;
        public virtual DbSet<CompanyData> CompanyData { get; set;
} = null!;
        public virtual DbSet<Domain.File> Files { get; set; }

        protected override void OnModelCreating(ModelBuilder
modelBuilder)
        {
            modelBuilder.Entity<CompanyData>(entity =>
            {
                entity.HasKey(e => e.CompanyName);

                entity.Property(e
e.CompanyAddress).HasMaxLength(100);
                entity.Property(e
e.CompanyName).HasMaxLength(50);
                entity.Property(e => e.CAcc).HasMaxLength(20);
                entity.Property(e => e.BIK).HasMaxLength(9);
                entity.Property(e => e.PAcc).HasMaxLength(20);
                entity.Property(e => e.INN).HasMaxLength(20);
                entity.Property(e
e.DirectorName).HasMaxLength(100);
                entity.Property(e => e.KPP).HasMaxLength(9);
                entity.Property(e => e.Bank).HasMaxLength(20);
            });
            modelBuilder.Entity<Domain.File>(entity =>
            {
                entity.HasKey(e => e.Id);

                entity.Property(e => e.Name).HasMaxLength(50);
            });
            modelBuilder.Entity<Authorization>(entity =>
            {
                entity.HasKey(e => e.UserId);

                entity.ToTable("Authorization");

                entity.Property(e
e.UserId).HasColumnName("User_id");
            });
        }
    }
}

```

```

        entity.Property(e
e.EmployeeId).HasColumnName("Employee_id");

        entity.Property(e => e.Password).HasMaxLength(50);

        entity.Property(e => e.Role).HasMaxLength(50);

        entity.Property(e => e.Username).HasMaxLength(50);

        entity.HasOne(d => d.Employee)
            .WithMany(p => p.Authorizations)
            .HasForeignKey(d => d.EmployeeId)
            .OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("FK_Authorization_Employee");

    });

    modelBuilder.Entity<Candidate>(entity =>
    {
        entity.ToTable("Candidate");

        entity.HasIndex(e
        => e.ContactDataId,
        "UQ__Candidat__31660442B77D070C")
            .IsUnique();

        entity.Property(e
e.CandidateId).HasColumnName("Candidate_id");

        entity.Property(e
e.ContactDataId).HasColumnName("Contact_data_id");

        entity.Property(e
e.Education).HasMaxLength(20);

        entity.Property(e
e.ExpirienseYears).HasColumnName("Expiriense_years");

        entity.Property(e
e.PassportId).HasColumnName("Passport_id");

        entity.HasOne(d => d.ContactData)
            .WithOne(p => p.Candidate)
            .HasForeignKey<Candidate>(d
d.ContactDataId)
            .OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK_Candidate_Contact_data");

        entity.HasOne(d => d.Passport)
            .WithMany(p => p.Candidates)

```

```

        .HasForeignKey(d => d.PassportId)
        .OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK_Candidate_Passport_info");

        entity.Navigation(x =>
x.ContactData).AutoInclude();
        entity.Navigation(x => x.Passport).AutoInclude();
    });

modelBuilder.Entity<ContactData>(entity =>
{
    entity.HasKey(e => e.ContactDataId);

    entity.ToTable("Contact_data");

    entity.Property(e => e.ContactDataId)
        .ValueGeneratedNever()
        .HasColumnName("Contact_data_id");

    entity.Property(e => e.Email).HasMaxLength(50);

    entity.Property(e => e.PhoneNumber)
        .HasMaxLength(12)
        .HasColumnName("Phone_number");

});

modelBuilder.Entity<Department>(entity =>
{
    entity.ToTable("Department");

    entity.Property(e
e.DepartmentId).HasColumnName("Department_id");

    entity.Property(e
e.BasicMoneyPerHour).HasColumnName("basic_money_per_hour");

    entity.Property(e
e.Direction).HasMaxLength(50);

    entity.Property(e
e.EmployeeCount).HasColumnName("Employee_count");

    entity.Property(e
e.ManagerId).HasColumnName("Manager_id");

    entity.Property(e
e.TotalMoneyPerHour).HasColumnName("total_money_per_hour");

    entity.HasOne(d => d.Manager)

```

```

        .WithMany(p => p.Departments)
        .HasForeignKey(d => d.ManagerId)
        .HasConstraintName("FK_Department_Employee");

    entity.Navigation(x => x.Employees).AutoInclude();
});

modelBuilder.Entity<DepartmentWorkLoad>(entity =>
{
    entity.HasKey(e => e.ScheduleId);

    entity.ToTable("Department_work_load");

    entity.Property(e
e.ScheduleId).HasColumnName("Schedule_id"); =>

    entity.Property(e
e.DepartmentId).HasColumnName("Department_id"); =>

    entity.Property(e
e.IsEqualOrMore).HasColumnName("isEqualOrMore"); =>

    entity.Property(e
e.PeriodId).HasColumnName("Period_id"); =>

    entity.Property(e
e.WorkLoad).HasColumnName("Work_load"); =>

    entity.Property(e
e.WorkedHours).HasColumnName("Worked_hours"); =>

    entity.HasOne(d => d.Department)
        .WithMany(p => p.DepartmentWorkLoads)
        .HasForeignKey(d => d.DepartmentId)
        .OnDelete(DeleteBehavior.ClientSetNull)

    .HasConstraintName("FK_Department_work_load_Department");

    entity.HasOne(d => d.Period)
        .WithMany(p => p.DepartmentWorkLoads)
        .HasForeignKey(d => d.PeriodId)

    .HasConstraintName("FK_Department_work_load_Period");

    entity.Navigation(x
x.Department).AutoInclude(); =>
    entity.Navigation(x => x.Period).AutoInclude();
});

modelBuilder.Entity<Dismissal>(entity =>
{

```

```

        entity.ToTable("Dismissal");

        entity.Property(e => e.DismissalId)
            .HasColumnName("Dismissal_id");

        entity.Property(e => e.DismissalDate)
            .HasColumnType("date")
            .HasColumnName("Dismissal_date");

        entity.Property(e => e.DismissalReason)
            .HasMaxLength(50)
            .HasColumnName("Dismissal_reason");

        entity.Property(e => e.DocumentDate)
            .HasColumnType("date")
            .HasColumnName("Document_date");

        entity.Property(e => e.PassportId)
            .HasColumnName("Passport_id");

        entity.HasOne(d => d.Passport)
            .WithMany(p => p.Dismissals)
            .HasForeignKey(d => d.PassportId)
            .HasConstraintName("FK_Dismissal_Passport_info");

        entity.Navigation(x => x.Passport).AutoInclude();
    });

    modelBuilder.Entity<Document>(entity =>
    {
        entity.ToTable("Document");

        entity.Property(e => e.DocumentId)
            .ValueGeneratedNever()
            .HasColumnName("Document_id");

        entity.Property(e => e.CandidateId)
            .HasColumnName("Candidate_id");

        entity.Property(e => e.DocumentType)
            .HasMaxLength(50)
            .HasColumnName("Document_type");

        entity.Property(e => e.DocumentUrl)
            .HasMaxLength(50)
            .HasColumnName("Document_url");

        entity.HasOne(d => d.Candidate)
            .WithMany(p => p.Documents)
            .HasForeignKey(d => d.CandidateId)

```

```

        .HasConstraintName("FK_Document_Candidate");

        entity.Navigation(x => x.Candidate).AutoInclude();
    });

    modelBuilder.Entity<Employee>(entity =>
    {
        entity.ToTable("Employee");

        entity.Property(e
e.EmployeeId).HasColumnName("Employee_id"); =>

        entity.Property(e => e.AuthorizationCode)
            .HasMaxLength(50)
            .HasColumnName("Authorization_code");

        entity.Property(e
e.ContactDataId).HasColumnName("Contact_data_id"); =>

        entity.Property(e
e.DepartmentId).HasColumnName("Department_id"); =>

        entity.Property(e
e.InterviewId).HasColumnName("Interview_id"); =>

        entity.Property(e
e.PassportId).HasColumnName("Passport_id"); =>

        entity.HasOne(d => d.ContactData)
            .WithMany(p => p.Employees)
            .HasForeignKey(d => d.ContactDataId)
            .OnDelete(DeleteBehavior.ClientSetNull)

        .HasConstraintName("FK_Employee_Contact_data");

        entity.HasOne(d => d.Department)
            .WithMany(p => p.Employees)
            .HasForeignKey(d => d.DepartmentId)
            .HasConstraintName("FK_Employee_Department");

        entity.HasOne(d => d.Interview)
            .WithMany(p => p.Employees)
            .HasForeignKey(d => d.InterviewId)
            .HasConstraintName("FK_Employee_Interview");

        entity.HasOne(d => d.Passport)
            .WithMany(p => p.Employees)
            .HasForeignKey(d => d.PassportId)
            .OnDelete(DeleteBehavior.ClientSetNull)

        .HasConstraintName("FK_Employee_Passport_info");
    });

```

```

        entity.Navigation(x
x.Authorizations).AutoInclude();
        entity.Navigation(x
x.ContactData).AutoInclude();
        entity.Navigation(x=>x.Passport).AutoInclude();
        entity.Navigation(x => x.Interview).AutoInclude();
    });

    modelBuilder.Entity<EmployeeWorkLoad>(entity =>
    {
        entity.HasKey(e => e.AddendumId);

        entity.ToTable("Employee_work_load");

        entity.Property(e
e.AddendumId).HasColumnName("Addendum_id");

        entity.Property(e
e.EmployeeId).HasColumnName("Employee_id");

        entity.Property(e
e.PeriodId).HasColumnName("Period_id");

        entity.Property(e
e.ResultSalary).HasColumnName("result_salary");

        entity.Property(e
e.WorkLoadHours).HasColumnName("Work_load_hours");

        entity.Property(e
e.WorkedHours).HasColumnName("Worked_hours");

        entity.HasOne(d => d.Employee)
            .WithMany(p => p.EmployeeWorkLoads)
            .HasForeignKey(d => d.EmployeeId)

        .HasConstraintName("FK_Employee_work_load_Employee");

        entity.HasOne(d => d.Period)
            .WithMany(p => p.EmployeeWorkLoads)
            .HasForeignKey(d => d.PeriodId)

        .HasConstraintName("FK_Employee_work_load_Period");

        entity.Navigation(x => x.Employee).AutoInclude();
        entity.Navigation(x => x.Period).AutoInclude();
    });

    modelBuilder.Entity<Interview>(entity =>
    {

```

```

        entity.ToTable("Interview");

        entity.Property(e
e.InterviewId).HasColumnName("Interview_id"); =>

        entity.Property(e
e.CandidateId).HasColumnName("Candidate_id"); =>

        entity.Property(e
e.Date).HasColumnType("date"); =>

        entity.HasOne(d => d.Candidate)
            .WithMany(p => p.Interviews)
            .HasForeignKey(d => d.CandidateId)
            .HasConstraintName("FK_Interview_Candidate");

        entity.Navigation(x => x.Candidate).AutoInclude();
    });

modelBuilder.Entity<PassportInfo>(entity =>
{
    entity.HasKey(e => e.PassportId);

    entity.ToTable("Passport_info");

    entity.Property(e => e.PassportId)
        .ValueGeneratedNever()
        .HasColumnName("Passport_id");

    entity.Property(e => e.City).HasMaxLength(50);

    entity.Property(e => e.Country).HasMaxLength(50);

    entity.Property(e => e.Lastname).HasMaxLength(50);

    entity.Property(e => e.Name).HasMaxLength(50);

    entity.Property(e
e.PassportNumber).HasColumnName("Passport_number"); =>

    entity.Property(e
e.PassportSerial).HasColumnName("Passport_serial"); =>

    entity.Property(e => e.State).HasMaxLength(50);

    entity.Property(e => e.Street).HasMaxLength(50);

    entity.Property(e => e.Surname).HasMaxLength(50);
});

modelBuilder.Entity<Period>(entity =>

```



```

        {
            entity.ToTable("Period");

            entity.Property(e
e.PeriodId).HasColumnName("Period_id");

            entity.Property(e
e.TotalWorkLoadHours).HasColumnName("Total_work_load_hours");
        });

        modelBuilder.Entity<PersonalAchievement>(entity =>
        {
            entity.HasKey(e => e.AchievementId);

            entity.ToTable("Personal_achievements");

            entity.Property(e
e.AchievementId).HasColumnName("Achievement_id");

            entity.Property(e
e.Description).HasMaxLength(200);

            entity.Property(e
e.EmployeeId).HasColumnName("Employee_id");

            entity.Property(e
e.PeriodId).HasColumnName("Period_id");

            entity.HasOne(d => d.Employee)
                .WithMany(p => p.PersonalAchievements)
                .HasForeignKey(d => d.EmployeeId)

            .HasConstraintName("FK_Personal_achievements_Employee");

            entity.HasOne(d => d.Period)
                .WithMany(p => p.PersonalAchievements)
                .HasForeignKey(d => d.PeriodId)

            .HasConstraintName("FK_Personal_achievements_Period");

            entity.Navigation(x => x.Period).AutoInclude();
        });

        modelBuilder.OnModelCreatingPartial(modelBuilder);
    }

    partial void OnModelCreatingPartial(ModelBuilder
modelBuilder);
}
}

```

Класс DBInitializer:

```
namespace HRM.Persistence
{
    public static class DBInitializer
    {
        public static void Initialize(HRMDBContext context)
        {
            context.Database.EnsureCreated();
        }
    }
}
```

Класс DependencyInjection:

```
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.EntityFrameworkCore;
using HRM.Application.Interfaces;

namespace HRM.Persistence
{
    public static class DependencyInjection
    {
        public static IServiceCollection AddPersistence(this
IServiceCollection services, IConfiguration configuration)
        {
            var connectionString = configuration["DbConnection"];
            services.AddDbContext<HRMDBContext>(options =>
            {
                options.UseSqlServer(connectionString);
            });
            services.AddScoped<IHRMDBContext>(provider =>
                provider.GetService<HRMDBContext>());

            services.AddTransient<IUnitOfWork, UnitOfWork>();
            //services.AddTransient<IFileRepository,
FileRepository>();

            return services;
        }
    }
}
```

## Приложение 5. Код уровня Presentation (Web Api)

Класс Program:

```
using HRM.Application;
using HRM.Persistence;

var builder = WebApplication.CreateBuilder(args);
//Configure Services

builder.Services.AddApplication();
builder.Services.AddPersistence(builder.Configuration);
builder.Services.AddControllers().AddNewtonsoftJson(options =>
    options.SerializerSettings.ReferenceLoopHandling =
Newtonsoft.Json.ReferenceLoopHandling.Ignore); //игнор    циклического
вызова связанных сущностей
//builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var host = builder.Build();
//Configure
if(host.Environment.IsDevelopment())
{
    host.UseDeveloperExceptionPage();
}

host.UseSwagger();

host.UseSwaggerUI(Config =>
{
    Config.RoutePrefix = string.Empty;
    Config.SwaggerEndpoint("swagger/v1/swagger.json", "HRM API");
});
host.UseRouting();

host.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();
});

using (var scope = host.Services.CreateScope())
{
    var serviceProvider = scope.ServiceProvider;
    try
    {
        //var context
        serviceProvider.GetRequiredService<HRMDBContext>();
        //DBInitializer.Initialize(context);
    }
}
```

```

        catch { }
    }

```

```

host.Run();

```

## Контроллеры REST

Класс контроллера авторизации AuthorizationController:

```

using HRM.Application.Authorization.Login;
using HRM.Application.Authorization.Registration;
using HRM.Application.Interfaces;
using HRM.Domain;
using Microsoft.AspNetCore.Mvc;

namespace HRM.WebApi.Controllers.Authorization
{
    [ApiController]
    [Route("api/[controller]")]
    public class AuthorizationController : Controller
    {
        private readonly IUnitOfWork context;
        public AuthorizationController(IUnitOfWork context)
        {
            this.context = context;
        }

        [HttpPost("Registration")]
        public async Task<IActionResult>
Registration(RegistrationCommand request)
        {
            var handler = new RegistrationCommandHandler(context);
            await handler.Registration(request);
            return Ok(request);
        }

        [HttpPost("Login")]
        public async Task<ActionResult<int>> Login(LoginCommand
request)
        {
            var handler = new LoginCommandHandler(context);
            int permission = await handler.TryLogin(request);
            if (permission == 3)
                return 3; //returnUrl to admin home page
            if (permission == 2)
                return 2; //returnUrl to admin home page
            if (permission == 1)
                return 1; //returnUrl to admin home page
            else
                return 0;
        }
    }
}

```

```

        [HttpPost("Authorize")]
        public async Task<ActionResult<Employee>>
Authorize(LoginCommand request)
        {
            var employee =
context.Employee.GetByAuthData(request.Username, request.Password);
            if (employee != null)
                return employee;
            else
                return null;
        }
    }
}

```

Класс контроллера достижений AchievementController:

```

using HRM.Application.Interfaces;
using HRM.Application.AchievementsHandling;
using HRM.Domain;
using Microsoft.AspNetCore.Mvc;

namespace HRM.WebApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class AchievementsController : Controller
    {
        private readonly IUnitOfWork context;
        public AchievementsController(IUnitOfWork context)
        {
            this.context = context;
        }
        [HttpGet]
        public async Task<List<PersonalAchievement>>
GetAchievementsList()
        {
            return await context.Achievement.GetAllAsync();
        }
        [HttpPost]
        public async Task<IActionResult>
UpdateAchievementsList(AddOrUpdatePersonalAchievementCommand request)
        {
            var handler = new
AddOrUpdatePersonalAchievementCommandHandler(context);
            await handler.AddOrUpdate(request);
            return Ok();
        }
    }
}

```

Класс контроллера кандидатов CandidateController:

```
using HRM.Application.Interfaces;
using HRM.Application.Interviewing;
using HRM.Domain;
using Microsoft.AspNetCore.Mvc;

namespace HRM.WebApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class CandidateController : ControllerBase
    {
        private readonly IUnitOfWork context;
        public CandidateController(IUnitOfWork context)
        {
            this.context = context;
        }

        [HttpGet("all")]
        public async Task<ActionResult<List<Candidate>>> GetAll()
        {
            var res = await context.Candidate.GetAllAsync();
            return Ok(res);
        }

        [HttpPost]
        public async Task<ActionResult>
        TakeInterview(InterviewingCommand request)
        {
            InterviewingCommandHandler handler = new
            InterviewingCommandHandler(context);
            await handler.TakeInterview(request);
            return Ok(request);
        }

        [HttpGet("interviewed")]
        public async Task<ActionResult<List<Candidate>>>
        GetInterviewed()
        {
            var res = context.Candidate.GetInterviewed();
            return Ok(res);
        }
    }
}
```

Класс контроллера данных компании CompanyDataController:

```
using HRM.Application.CompanyData;
using HRM.Application.Interfaces;
using HRM.Domain;
```

```

using Microsoft.AspNetCore.Mvc;

namespace HRM.WebApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class CompanyDataController : Controller
    {
        private readonly IUnitOfWork context;
        public CompanyDataController(IUnitOfWork context)
        {
            this.context = context;
        }

        [HttpPost]
        public async Task<IActionResult>
FillCompanyData(FillCompanyDataCommand request)
        {
            var handler = new
FillCompanyDataCommandHandler(context);
            await handler.AddOrUpdate(request);
            return Ok(request);
        }

        [HttpGet]
        public async Task<ActionResult<CompanyData>>
CheckCompanyData()
        {
            return await context.CompanyData.FirstAsync();
        }
    }
}

```

Класс контроллера увольнений DismissalController:

```

using HRM.Application.Dismissing;
using HRM.Application.Interfaces;
using Microsoft.AspNetCore.Mvc;

namespace HRM.WebApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class DismissalController : ControllerBase
    {
        private readonly IUnitOfWork context;
        public DismissalController(IUnitOfWork context)
        {
            this.context = context;
        }

        [HttpDelete]

```

```

        public async Task<IActionResult>
DismissEmployee(DismissingCommand request)
    {
        var handler = new DismissingCommandHandler(context);
        await handler.Dismiss(request);
        return Ok(request);
    }
}

```

Класс контроллера распределения рабочей нагрузки

DistributionController:

```

using HRM.Application.Interfaces;
using HRM.Application.WorkLoadDistribution.CreateDistribution;
using HRM.Application.WorkLoadDistribution.GenerateAddendum;
using HRM.Application.WorkLoadDistribution.UpdateDistribution;
using Microsoft.AspNetCore.Mvc;
using System.IO.Compression;

namespace HRM.WebApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]

    public class DistributionController : Controller
    {
        private IUnitOfWork context { get; }

        public DistributionController(IUnitOfWork _context)
        {
            context = _context;
        }

        [HttpPost]
        public async Task<IActionResult>
Distribute(CreateDistributionCommand request)
        {
            var handler = new
CreateDistributionCommandHandler(context);
            await handler.Distribute(request);
            return Ok(request);
        }

        [HttpPut]
        public async Task<IActionResult>
UpdateDistribution(UpdateDistributionCommand request)
        {
            var handler = new
UpdateDistributionCommandHandler(context);
            await handler.UpdateDistribution(request);
        }
    }
}

```



```

        return Ok(request);
    }

    [HttpHead("Addendum")]
    public async Task<ActionResult> GenerateAddendum()
    {
        var handler = new
GenerateAddendumCommandHandler(context);
        foreach (var employeeLoad in
context.EmployeeWorkLoad.GetByPeriodId(1))
        {
            await handler.GenerateAddendum(
                new GenerateAddendumCommand
                {
                    PeriodId = 1,
                    EmployeeId = employeeLoad.EmployeeId,
                    WorkLoad = employeeLoad.WorkLoadHours
                });
        }
        await context.File.ClearStorage(@"wwwroot/files/");
        return Ok();
    }

    [HttpGet("Addendum/{id}")]
    public async Task<FileStreamResult> GetAddendum(int id)
    {
        var addendum = context.File.GetOne(id,
context.Period.GetPeriodByDate(DateTime.Now).PeriodId);
        var adData = addendum.Data;
        var path = @"wwwroot/files/" + addendum.Name;
        using (var stream = new FileStream(path,
        FileMode.OpenOrCreate, FileAccess.Write))
        {
            await stream.WriteAsync(adData, 0, adData.Length);
        }

        var fileStream = System.IO.File.OpenRead(path);
        return File(fileStream, "application
/vnd.openxmlformats-officedocument.wordprocessingml.document");
    }

    [HttpGet("Addendum/Period/{periodid}")]
    public async Task<FileStreamResult> DownloadFolder(int
periodid)
    {
        string path;
        var current = await
context.Period.GetByIdAsync(periodid);
        foreach (var addendum in
context.File.GetAllByPeriodId(periodid))
        {
            var adData = addendum.Data;

```

```

        path = @"wwwroot/files/" + addendum.Name;
        using (var stream = new FileStream(path,
            FileMode.OpenOrCreate, FileAccess.Write))
        {
            await stream.WriteAsync(adData);
        }
    }
    path = @"wwwroot/files/";
    FileStreamResult fileStreamResult;
    var tempPath = Path.Combine(Path.GetTempPath(),
        "temp.zip");

    path = path.Remove(path.Length - 1);
    ZipFile.CreateFromDirectory(path, tempPath,
        CompressionLevel.Fastest, false);
    FileStream fileStreamInput = new FileStream(tempPath,
        FileMode.Open, FileAccess.Read, FileShare.Delete);
    fileStreamResult = new
        FileStreamResult(fileStreamInput, "APPLICATION/octet-stream");
    fileStreamResult.FileNameDownload =
        context.Period.GetDateFromPeriod(current).ToString("Y") + ".zip";

    if (System.IO.File.Exists(tempPath))
    {
        System.IO.File.Delete(tempPath);
        context.File.ClearStorage(path + "/");
    }
    return fileStreamResult;
}
}
}
}

```

Класс контроллера сотрудников EmployeeController:

```

using HRM.Application.Interfaces;
using HRM.Domain;
using Microsoft.AspNetCore.Mvc;

namespace HRM.WebApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class EmployeeController : Controller
    {
        private readonly IUnitOfWork context;

        public EmployeeController(IUnitOfWork context)
        {
            this.context = context;
        }
    }
}

```

```

        [HttpGet]
        public List<Employee> GetActiveEmployee()
        {
            return context.Employee.GetActive().ToList();
        }
    }
}

```

Класс контроллера шаблона дополнительного соглашения

PrefabController:

```

using HRM.Application.Interfaces;
using Microsoft.AspNetCore.Mvc;

namespace HRM.WebApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class PrefabController : Controller
    {
        private readonly IUnitOfWork context;
        public PrefabController(IUnitOfWork context)
        {
            this.context = context;
        }

        [HttpPost]
        public async Task<IActionResult> SetPrefab(IFormFile file)
        {
            byte[] fileBytes;
            using (var ms = new MemoryStream())
            {
                file.CopyTo(ms);
                fileBytes = ms.ToArray();
            }
            var prefab = new Domain.File(); //await
context.File.GetByIdAsync(1);
            prefab.Name = "prefab.docx";
            prefab.Data = fileBytes;
            await context.File.CreateAsync(prefab);
            //await context.Save();
            return Ok();
        }

        [HttpGet]
        public async Task<FileStreamResult> GetPrefab()
        {
            string path = @"wwwroot/files/prefab.docx";

```

```

        //byte[] file =
context.File.GetByIdAsync(2).Result.Data;
        //using (var stream = new FileStream(path,
        FileMode.OpenOrCreate, FileAccess.Write))
        //{
        //    await stream.WriteAsync(file, 0, file.Length);
        //}

        byte[] prefabData = null;
        if (context.File.GetByIdAsync(1).Result is not null)
            prefabData =
context.File.GetByIdAsync(1).Result.Data;

        using (var stream = new FileStream(path,
        FileMode.OpenOrCreate, FileAccess.Write))
        {
            await stream.WriteAsync(prefabData, 0,
prefabData.Length);
        }

        var fileStream = System.IO.File.OpenRead(path);
        return File(fileStream,
"application/vnd.openxmlformats-
officedocument.wordprocessingml.document");
    }
}

```

Класс контроллера статистических данных StatisticsController:

```

using HRM.Application.Interfaces;
using HRM.Domain;
using Microsoft.AspNetCore.Mvc;

namespace HRM.WebApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class StatisticsController : Controller
    {
        private IUnitOfWork context { get; }

        public StatisticsController(IUnitOfWork context)
        {
            this.context = context;
        }

        [HttpGet("Period")]
        public async Task<List<Period>> GetPeriods()
        {
            return await context.Period.GetAllAsync();
        }
    }
}

```

```

    }

    [HttpGet("Period/Current")]
    public Period? GetCurrentPeriod()
    {
        return context.Period.GetPeriodByDate(DateTime.Now);
    }

    [HttpGet("Period/Next")]
    public Period? GetNextPeriod()
    {
        return
context.Period.GetPeriodByDate(DateTime.Now.AddMonths(1));
    }

    [HttpGet("Period/{Id}")]
    public async Task<Period> GetPeriodById(int Id)
    {
        return await context.Period.GetByIdAsync(Id);
    }

    [HttpGet("Period/Previous")]
    public async Task<ActionResult<Period?>>
GetPreviousPeriod()
    {
        var period =
context.Period.GetPeriodByDate(DateTime.Now.AddMonths(-1));
        if (period != null)
            return Ok(period);
        else
            return Ok(null);
    }

    [HttpGet("Department")]
    public async Task<List<Department>> GetDepartments()
    {
        return await context.Department.GetAllAsync();
    }

    [HttpGet("WorkLoad/Department/{Id}")]
    public List<DepartmentWorkLoad>
GetDepartmentWorkLoadsOfPeriod(int Id)
    {
        return
context.DepartmentWorkLoad.GetByPeriodId(Id).ToList();
    }

    [HttpGet("WorkLoad/Employee/{Id}")]
    public List<EmployeeWorkLoad>
GetEmployeeWorkLoadsOfPeriod(int Id)
    {
        return
context.EmployeeWorkLoad.GetByPeriodId(Id).ToList();
    }
}

```



## Приложение 6. Код уровня Presentation (Клиент)

В клиентском приложении производится независимое описание сущностей, однако в данном конкретном случае (в пространстве имен HRM.Desktop.Model), оно полностью совпадает с описанием сущностей на уровне Domain (в пространстве имен HRM.Domain).

Для обращения к серверной части в клиентском приложении формируются запросы (в пространстве имен HRM.Desktop.Commands), соответствующие описанным на уровне Application (в пространстве имен HRM.Application.[модуль]).

Код файла app.xaml:

```
<Application x:Class="HRM.Desktop.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:HRM.Desktop"
    xmlns:col="clr-namespace:System.Collections;assembly=mscorlib"
    xmlns:sys="clr-namespace:System;assembly=mscorlib"
    StartupUri="MainWindow.xaml">
    <Application.Resources>

        <Style x:Key="FocusVisual">
            <Setter Property="Control.Template">
                <Setter.Value>
                    <ControlTemplate>
                        <Rectangle Margin="2"
SnapsToDevicePixels="true" Stroke="{DynamicResource {x:Static
SystemColors.ControlTextBrushKey}}" StrokeThickness="1"
StrokeDashArray="1 2"/>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
        <SolidColorBrush x:Key="Button.Static.Background"
Color="#FFDDDDDD"/>
        <SolidColorBrush x:Key="Button.Static.Border"
Color="#FF707070"/>
        <SolidColorBrush x:Key="Button.MouseOver.Background"
Color="#FFBEE6FD"/>
        <SolidColorBrush x:Key="Button.MouseOver.Border"
Color="#FF3C7FB1"/>
```

```

        <SolidColorBrush x:Key="Button.Pressed.Background"
Color="#FFC4E5F6"/>
        <SolidColorBrush x:Key="Button.Pressed.Border"
Color="#FF2C628B"/>
        <SolidColorBrush x:Key="Button.Disabled.Background"
Color="#FFF4F4F4"/>
        <SolidColorBrush x:Key="Button.Disabled.Border"
Color="#FFADB2B5"/>
        <SolidColorBrush x:Key="Button.Disabled.Foreground"
Color="#FF838383"/>
        <SolidColorBrush x:Key="TextBox.MouseOver.Border"
Color="#FF7EB4EA"/>
        <SolidColorBrush x:Key="TextBox.Focus.Border"
Color="#FF569DE5"/>

        <Style x:Key="{ComponentResourceKey
ResourceId=DataGridSelectAllButtonStyle, TypeInTargetAssembly={x:Type
DataGrid}}" TargetType="{x:Type Button}">
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type Button}">
                        <Grid>
                            <Rectangle x:Name="Border"
Fill="{DynamicResource {x:Static SystemColors.ControlBrushKey}}"
SnapsToDevicePixels="True"/>
                            <Polygon x:Name="Arrow" Fill="Black"
HorizontalAlignment="Right" Margin="8,8,3,3" Opacity="0.15"
Points="0,10 10,10 10,0" Stretch="Uniform" VerticalAlignment="Bottom"/>
                        </Grid>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsMouseOver"
Value="True">
                                <Setter Property="Stroke"
TargetName="Border" Value="{DynamicResource {x:Static
SystemColors.ControlDarkBrushKey}}"/>
                            </Trigger>
                            <Trigger Property="IsPressed"
Value="True">
                                <Setter Property="Fill"
TargetName="Border" Value="{DynamicResource {x:Static
SystemColors.ControlDarkBrushKey}}"/>
                            </Trigger>
                            <Trigger Property="IsEnabled"
Value="False">
                                <Setter Property="Visibility"
TargetName="Arrow" Value="Collapsed"/>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>

```



```

        </Style>
        <Style x:Key="RoundedDataGrid" TargetType="{x:Type
DataGrid}">
            <Setter Property="Background" Value="{DynamicResource
{x:Static SystemColors.ControlBrushKey}}"/>
            <Setter Property="Foreground" Value="{DynamicResource
{x:Static SystemColors.ControlTextBrushKey}}"/>
            <Setter Property="BorderBrush" Value="#FF688CAF"/>
            <Setter Property="BorderThickness" Value="1"/>
            <Setter
Value="VisibleWhenSelected"/>
            <Setter
Property="ScrollViewer.CanContentScroll"
Value="true"/>
            <Setter
Property="ScrollViewer.PanningMode"
Value="Both"/>
            <Setter
Property="Stylus.IsFlicksEnabled"
Value="False"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type
DataGrid}">
                        <Border Background="{TemplateBinding
Background}" BorderBrush="{TemplateBinding BorderBrush}"
BorderThickness="{TemplateBinding BorderThickness}"
Padding="{TemplateBinding Padding}" SnapsToDevicePixels="True"
CornerRadius="10">
                            <ScrollViewer
x:Name="DG_ScrollViewer" Focusable="false" Margin="5">
                                <ScrollViewer.Template>
                                    <ControlTemplate
TargetType="{x:Type ScrollViewer}">
                                        <Grid>

<Grid.ColumnDefinitions>
                                                                    <ColumnDefinition
Width="Auto"/>
                                                                    <ColumnDefinition
Width="*/>
                                                                    <ColumnDefinition
Width="Auto"/>
</Grid.ColumnDefinitions>
                                                                    <Grid.RowDefinitions>
                                                                    <RowDefinition
Height="Auto"/>
                                                                    <RowDefinition
Height="*/>
                                                                    <RowDefinition
Height="Auto"/>
</Grid.RowDefinitions>

```

```

<Button
Command="{x:Static      DataGrid.SelectAllCommand}"      Focusable="false"
Style="{DynamicResource      {ComponentResourceKey
ResourceId=DataGridSelectAllButtonStyle,      TypeInTargetAssembly={x:Type
DataGrid}}}"      Visibility="{Binding      HeadersVisibility,
ConverterParameter={x:Static      DataGridHeadersVisibility.All},
Converter={x:Static      DataGrid.HeadersVisibilityConverter},
RelativeSource={RelativeSource      AncestorType={x:Type      DataGrid}}}"
Width="{Binding      CellsPanelHorizontalOffset,
RelativeSource={RelativeSource      AncestorType={x:Type      DataGrid}}}">/>

<DataGridColumnHeadersPresenter      x:Name="PART_ColumnHeadersPresenter"
Grid.Column="1"      Visibility="{Binding      HeadersVisibility,
ConverterParameter={x:Static      DataGridHeadersVisibility.Column},
Converter={x:Static      DataGrid.HeadersVisibilityConverter},
RelativeSource={RelativeSource      AncestorType={x:Type      DataGrid}}}">/>

<ScrollContentPresenter      x:Name="PART_ScrollContentPresenter"
CanContentScroll="{TemplateBinding      CanContentScroll}"
Grid.ColumnSpan="2" Grid.Row="1"/>

<ScrollBar
x:Name="PART_VerticalScrollBar"      Grid.Column="2"
Maximum="{TemplateBinding      ScrollableHeight}"      Orientation="Vertical"
Grid.Row="1"      Value="{Binding      VerticalOffset,      Mode=OneWay,
RelativeSource={RelativeSource      TemplatedParent}}"
ViewportSize="{TemplateBinding      ViewportHeight}"
Visibility="{TemplateBinding      ComputedVerticalScrollBarVisibility}">/>
<Grid Grid.Column="1"
Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition      Width="{Binding
NonFrozenColumnsViewportHorizontalOffset,
RelativeSource={RelativeSource      AncestorType={x:Type      DataGrid}}}">/>

<ColumnDefinition Width="*">/>

</Grid.ColumnDefinitions>

<ScrollBar
x:Name="PART_HorizontalScrollBar"      Grid.Column="1"
Maximum="{TemplateBinding      ScrollableWidth}"      Orientation="Horizontal"
Value="{Binding      HorizontalOffset,      Mode=OneWay,
RelativeSource={RelativeSource      TemplatedParent}}"
ViewportSize="{TemplateBinding      ViewportWidth}"
Visibility="{TemplateBinding      ComputedHorizontalScrollBarVisibility}">/>
</Grid>
</Grid>
</ControlTemplate>
</ScrollViewer.Template>

```

```

                                <ItemsPresenter
SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}"/>
                                </ScrollViewer>
                                </Border>
                                </ControlTemplate>
                                </Setter.Value>
                                </Setter>
                                <Style.Triggers>
                                    <MultiTrigger>
                                        <MultiTrigger.Conditions>
                                            <Condition
Value="true"/>
                                                Property="IsGrouping"
                                </Condition>
Property="VirtualizingPanel.IsVirtualizingWhenGrouping" Value="false"/>
                                        </MultiTrigger.Conditions>
                                        <Setter
Property="ScrollViewer.CanContentScroll" Value="false"/>
                                        </MultiTrigger>
                                    </Style.Triggers>
                                </Style>
                                <Style x:Key="ButtonStyle1" TargetType="{x:Type Button}">
                                    <Setter
Value="{StaticResource FocusVisual}"/>
                                        Property="FocusVisualStyle"
                                    <Setter Property="Background" Value="{StaticResource
Button.Static.Background}"/>
                                    <Setter Property="BorderBrush" Value="{x:Null}"/>
                                    <Setter Property="Foreground" Value="{x:Null}"/>
                                    <Setter Property="BorderThickness" Value="1"/>
                                    <Setter
Value="Center"/>
                                        Property="HorizontalContentAlignment"
                                    <Setter
Value="Center"/>
                                        Property="VerticalContentAlignment"
                                    <Setter Property="Padding" Value="1"/>
                                    <Setter Property="Template">
                                        <Setter.Value>
                                            <ControlTemplate TargetType="{x:Type Button}">
                                                <Border x:Name="border" CornerRadius="15"
BorderBrush="{TemplateBinding
BorderBrush}"
BorderThickness="{TemplateBinding
BorderThickness}"
Background="{TemplateBinding Background}" SnapsToDevicePixels="true">
                                                    <ContentPresenter
x:Name="contentPresenter"
                                                        Focusable="False"
HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}"
Margin="{TemplateBinding
Padding}"
RecognizesAccessKey="True"
SnapsToDevicePixels="{TemplateBinding
SnapsToDevicePixels}"
VerticalAlignment="{TemplateBinding
VerticalContentAlignment}"/>
                                                </Border>
                                                <ControlTemplate.Triggers>
                                                    <Trigger
Value="true">
                                                        Property="IsDefaulted"

```

```

                                <Setter      Property="BorderBrush"
TargetName="border"      Value="{DynamicResource      {x:Static
SystemColors.HighlightBrushKey}}"/>
                                </Trigger>
                                <Trigger      Property="IsMouseOver"
Value="true">
                                <Setter      Property="Opacity"
TargetName="border" Value="80"></Setter>

                                <Setter      Property="Background"
TargetName="border" Value="#FFD4EDF9"/>
                                </Trigger>
                                <Trigger      Property="IsPressed"
Value="true">
                                <Setter      Property="Opacity"
TargetName="border" Value="80"></Setter>
                                <Setter      Property="BorderBrush"
TargetName="border" Value="#FFD4EDF9"/>
                                </Trigger>
                                <Trigger      Property="IsEnabled"
Value="false">
                                <Setter      Property="Opacity"
TargetName="border" Value="80"></Setter>
                                </Trigger>
                                </ControlTemplate.Triggers>
                                </ControlTemplate>
                                </Setter.Value>
                                </Setter>
                                </Style>

                                <Style x:Key="ButtonStyle2" TargetType="{x:Type Button}">
                                    <Setter      Property="FocusVisualStyle"
Value="{StaticResource FocusVisual}"/>
                                    <Setter      Property="Background" Value="{StaticResource
Button.Static.Background}"/>
                                    <Setter      Property="BorderBrush" Value="{StaticResource
Button.Static.Border}"/>
                                    <Setter      Property="Foreground" Value="Black"/>
                                    <Setter      Property="FontSize" Value="18"/>
                                    <Setter      Property="BorderThickness" Value="1"/>
                                    <Setter      Property="HorizontalContentAlignment"
Value="Center"/>
                                    <Setter      Property="VerticalContentAlignment"
Value="Center"/>
                                    <Setter      Property="Padding" Value="1"/>
                                    <Setter      Property="Template">
                                        <Setter.Value>
                                            <ControlTemplate TargetType="{x:Type Button}">
                                                <Border x:Name="border" CornerRadius="7"
BorderBrush="{TemplateBinding      BorderBrush}"

```

```

BorderThickness="{TemplateBinding BorderThickness}"
Background="{TemplateBinding Background}" SnapsToDevicePixels="true">
    <ContentPresenter
        x:Name="contentPresenter" Focusable="False"
        HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
        Margin="{TemplateBinding Padding}" RecognizesAccessKey="True"
        SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}"
        VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>
    </Border>
    <ControlTemplate.Triggers>
        <Trigger Property="IsDefaulted"
            Value="true">
            <Setter Property="BorderBrush"
                TargetName="border" Value="{DynamicResource {x:Static
                    SystemColors.HighlightBrushKey}}"/>
            </Trigger>
        <Trigger Property="IsMouseOver"
            Value="true">
            <Setter Property="BorderBrush"
                TargetName="border" Value="{StaticResource Button.MouseOver.Border}"/>
            </Trigger>
        <Trigger Property="IsPressed"
            Value="true">
            <Setter Property="BorderBrush"
                TargetName="border" Value="{StaticResource Button.Pressed.Border}"/>
            <Setter Property="Background"
                TargetName="border" Value="{StaticResource
                    Button.Pressed.Background}"/>
            </Trigger>
        <Trigger Property="IsEnabled"
            Value="false">
            <Setter Property="Opacity"
                TargetName="border" Value="80"></Setter>
            </Trigger>
    </ControlTemplate.Triggers>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style>

<Style x:Key="UnfocusedTextBlock" TargetType="TextBlock">
    <Setter Property="FontSize" Value="24"></Setter>
    <Setter Property="Foreground"
        Value="#3ca0dc"></Setter>
</Style>

<Style x:Key="FirstStageLabel" TargetType="Label">
    <Setter Property="FontSize" Value="18"></Setter>
    <Setter Property="Foreground" Value="Black"></Setter>
</Style>

```

```

        <ControlTemplate x:Key="LabelRounder" TargetType="Label">
            <Border
                x:Name="border"
                CornerRadius="7"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="{TemplateBinding BorderThickness}"
                Background="{TemplateBinding Background}" SnapsToDevicePixels="true">
                <ContentPresenter
                    x:Name="contentPresenter"
                    Focusable="False"
                    HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"
                    Margin="{TemplateBinding Padding}"
                    RecognizesAccessKey="True"
                    SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}"
                    VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>
            </Border>
        </ControlTemplate>

        <Style x:Key="TextInput" TargetType="TextBox">
            <Setter Property="FontSize" Value="18"></Setter>
            <Setter Property="Foreground" Value="Black"></Setter>
            <Setter Property="Background" Value="White"></Setter>
            <Setter
                Property="BorderBrush"
                Value="#3ca0dc"></Setter>
            <Setter Property="TextWrapping" Value="Wrap"></Setter>
        </Style>

        <ControlTemplate x:Key="Rounder">
            <Border Background="{TemplateBinding Background}"
                x:Name="Bd" BorderBrush="#FF3CA0DC"
                BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="10">
                <ScrollViewer x:Name="PART_ContentHost"/>
            </Border>
            <ControlTemplate.Triggers>
                <Trigger Property="IsEnabled" Value="False">
                    <Setter
                        Property="Background"
                        Value="{DynamicResource {x:Static SystemColors.ControlBrushKey}}"
                        TargetName="Bd"/>
                    <Setter
                        Property="Foreground"
                        Value="{DynamicResource {x:Static SystemColors.GrayTextBrushKey}}"/>
                </Trigger>
                <Trigger Property="Width" Value="Auto">
                    <Setter Property="MinWidth" Value="100"/>
                </Trigger>
                <Trigger Property="Height" Value="Auto">
                    <Setter Property="MinHeight" Value="20"/>
                </Trigger>
            </ControlTemplate.Triggers>
        </ControlTemplate>

        <Style x:Key="PasswordInput" TargetType="PasswordBox">
            <Setter Property="FontSize" Value="18"></Setter>
            <Setter Property="Foreground" Value="Black"></Setter>

```

```

        <Setter                                Property="Background"
Value="Transparent"></Setter>
        <Setter                                Property="BorderBrush"
Value="#3ca0dc"></Setter>
    </Style>
</Application.Resources>
</Application>

```

Код разметки окна MainWindow:

```

<Window x:Class="HRM.Desktop.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:local="clr-namespace:HRM.Desktop"
    mc:Ignorable="d"
    Title="MainWindow"           Height="604.94"           Width="800"
    Loaded="OnWindowLoaded"      WindowStartupLocation="CenterScreen"
    MouseMove="Window_MouseMove" MinHeight="450" MinWidth="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="50"></RowDefinition>
            <RowDefinition Height="*"></RowDefinition>
        </Grid.RowDefinitions>
        <Frame x:Name="MenuFrame" NavigationUIVisibility="Hidden"
Grid.Row="0" IsTabStop="False"/>
        <Frame x:Name="MainFrame" NavigationUIVisibility="Hidden"
Grid.Row="1" IsTabStop="False" Margin="5"></Frame>
    </Grid>
</Window>

```

Код окна MainWindow:

```

using HRM.Desktop.Handlers;
using System.Net.Http;
using System.Windows;
using System.Windows.Input;

namespace HRM.Desktop
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public Timer timer;
        public string currentPage = "";
    }
}

```

```

        public ExceptionsAndMessenges error;
        public WindowSizeController sizeController;
        public readonly HttpClient client = new HttpClient();
        public MainWindow()
        {
            InitializeComponent();
            error = new ExceptionsAndMessenges(this);
            timer = new Timer(this);
            sizeController = new WindowSizeController(this);
            MainFrame.Navigate(new
Pages.AuthorizationPage(this));
        }

        private void OnWindowLoaded(object sender, RoutedEventArgs
e)
        {
            timer.StartupTimer();
        }

        private void Window_MouseMove(object sender,
MouseEventArgs e)
        {
            timer.ResetTimer();
        }
    }
}

```

Код разметки страницы авторизации:

```

<Page x:Class="HRM.Desktop.Pages.AuthorizationPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:HRM.Desktop.Pages"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="AuthorizationPage">

    <Grid>
        <Label Content="Логин:" HorizontalAlignment="Left"
Height="31" Margin="173,84,0,0" VerticalAlignment="Top" Width="98"
Style="{DynamicResource FirstStageLabel}"/>
        <Label Content="Пароль:" HorizontalAlignment="Left"
Height="31" Margin="173,141,0,0" VerticalAlignment="Top" Width="98"
Style="{DynamicResource FirstStageLabel}" />
        <TextBox x:Name="UsernameTB" HorizontalAlignment="Left"
Height="31" Margin="276,84,0,0" TextWrapping="Wrap" Text=""

```



```

VerticalAlignment="Top" Width="311" FontSize="20"
Style="{DynamicResource TextInput}" Template="{StaticResource
Rounded}"/>
        <PasswordBox x:Name="PasswordPB"
HorizontalAlignment="Left" Height="31" Margin="276,140,0,0"
VerticalAlignment="Top" Width="311" FontSize="20"
Style="{DynamicResource PasswordInput}" Background="White"
Template="{StaticResource Rounded}" />
        <CheckBox x:Name="SaveAuthChB" Content="Запомнить меня"
HorizontalAlignment="Left" Margin="276,176,0,0" VerticalAlignment="Top"
Height="20"/>
        <Button Content="Войти" Name="Login_Button"
HorizontalAlignment="Left" Margin="221,212,0,0" VerticalAlignment="Top"
Width="138" Height="29" Click="Login_Button_Click"
Style="{DynamicResource ButtonStyle2}"/>
        <Button Content="Выход" Name="Exit_Button"
HorizontalAlignment="Left" Margin="428,212,0,0" VerticalAlignment="Top"
Width="134" Height="29" Click="Exit_Button_Click"
Style="{DynamicResource ButtonStyle2}"/>
        <Button Content="Регистрация" x:Name="Registration_Button"
HorizontalAlignment="Left" Margin="484,176,0,0"
VerticalContentAlignment="Top" VerticalAlignment="Top" Width="103"
Style="{DynamicResource ButtonStyle2}" FontSize="12"
Click="Registration_Button_Click"/>
    </Grid>
</Page>

```

Код страницы авторизации:

```

using HRM.Desktop.Handlers;
using HRM.Desktop.Model;
using HRM.Desktop.Pages.AdminNavigation;
using HRM.Desktop.Pages.Menu;
using Newtonsoft.Json;
using System;
using System.Net.Http;
using System.Text;
using System.Windows;
using System.Windows.Controls;

namespace HRM.Desktop.Pages
{
    /// <summary>
    /// Логика взаимодействия для AuthorizationPage.xaml
    /// </summary>
    public partial class AuthorizationPage : Page
    {
        readonly Timer timer;
        readonly MainWindow window;
        public AuthorizationPage(MainWindow mainWindow)
        {

```

```

        window = mainWindow;
        InitializeComponent();
        timer = window.timer;
        window.currentPage = "AuthorizationPage";

window.sizeController.CheckWindowSize("AuthorizationPage");
        window.Title = "Вход";

        UsernameTB.Text =
Properties.Settings.Default.UserName;
        PasswordPB.Password =
Properties.Settings.Default.Password;
    }
    private void Login_Button_Click(object sender,
RoutedEventArgs e)
    {
        string login = UsernameTB.Text;
        string pass = PasswordPB.Password;

        if (login == "" || pass == "")
        {
            window.error.LackInputExeption();
            return;
        }
        var loginContent = new
StringContent(JsonConvert.SerializeObject(new Commands.LoginCommand {
Password = pass, Username = login }), Encoding.UTF8,
"application/json");
        var loginResponse = window.client.PostAsync(new
Uri("https://localhost:44355/api/Authorization/Login"),
loginContent).Result;
        int responseContentCode =
(int)JsonConvert.DeserializeObject(loginResponse.Content.ReadAsStringAsync().Result, typeof(int));
        if (responseContentCode != 0)
        {
            var authContent = new
StringContent(JsonConvert.SerializeObject(new Commands.LoginCommand {
Password = pass, Username = login }), Encoding.UTF8,
"application/json");
            var employeeResponse = window.client.PostAsync(new
Uri("https://localhost:44355/api/Authorization/Authorize"),
authContent).Result;
            var responseContentEmployee =
(Employee)JsonConvert.DeserializeObject(employeeResponse.Content.ReadAsStringAsync().Result, typeof(Employee));
            if ((bool)SaveAuthChB.IsChecked)
            {
                Properties.Settings.Default.UserName = login;
                Properties.Settings.Default.Password = pass;
            }
        }
    }

```

```

        if (responseContentCode == 3)
        {
            window.MenuFrame.Navigate(new
AdminMenu(window));
            window.MainFrame.Navigate(new
AdminIndex(window));
        }
        else if (responseContentCode == 2)
        {
            window.MenuFrame.Navigate(new
AdminMenu(window, responseContentEmployee));
            window.MainFrame.Navigate(new
AdminIndex(window, responseContentEmployee));
        }

        else if (responseContentCode == 1)
        {

//window.MenuFrame.NavigationService.Navigate(new      UserMenu(window,
responseContentEmployee));
        }
        else
            window.error.AuthorizationError();
    }

    private void Exit_Button_Click(object sender,
RoutedEventArgs e)
    {
        window.Close();
    }

    private void Registration_Button_Click(object sender,
RoutedEventArgs e)
    {
        window.MainFrame.Navigate(new
RegistrationPage(window));
    }
}

```

Код разметки страницы регистрации:

```

<Page x:Class="HRM.Desktop.Pages.RegistrationPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

```

```

xmlns:local="clr-namespace:HRM.Desktop.Pages"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800"
Title="RegistrationPage">

    <Grid>
        <Label Content="Логин:" HorizontalAlignment="Center"
Height="31" VerticalAlignment="Top" Style="{DynamicResource
FirstStageLabel}" Margin="0,-11,0,0"/>
        <Label Content="Пароль:" HorizontalAlignment="Center"
Height="31" Margin="0,63,0,0" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}" />
        <TextBox x:Name="UsernameTB" HorizontalAlignment="Center"
Height="31" Margin="0,20,0,0" TextWrapping="Wrap" Text=""
VerticalAlignment="Top" Width="312" FontSize="20"
Style="{DynamicResource TextInput}" Template="{StaticResource
Rounded}" />
        <PasswordBox x:Name="PasswordPB"
HorizontalAlignment="Center" Height="31" Margin="0,95,0,0"
VerticalAlignment="Top" Width="312" FontSize="20"
Style="{DynamicResource PasswordInput}" Template="{StaticResource
Rounded}" Background="White" />
        <Button Content="Продолжить" x:Name="Registrate_Button"
HorizontalAlignment="Left" Margin="221,315,0,0" VerticalAlignment="Top"
Width="138" Height="29" Style="{DynamicResource ButtonStyle2}"
Click="Registrate_Button_Click"/>
        <Button Content="Назад" x:Name="Back_Button"
HorizontalAlignment="Left" Margin="428,315,0,0" VerticalAlignment="Top"
Width="134" Height="29" Style="{DynamicResource ButtonStyle2}"
Click="Back_Button_Click"/>
        <Label Content="Повторите пароль:"
HorizontalAlignment="Center" Height="31" Margin="0,140,0,0"
VerticalAlignment="Top" Style="{DynamicResource FirstStageLabel}" />
        <PasswordBox x:Name="PasswordPB_Copy"
HorizontalAlignment="Center" Height="31" Margin="0,172,0,0"
VerticalAlignment="Top" Width="312" FontSize="20"
Style="{DynamicResource PasswordInput}" Template="{StaticResource
Rounded}" Background="White" />
        <Label Content="Идентификатор"
HorizontalAlignment="Center" Height="31" Margin="0,220,0,0"
VerticalAlignment="Top" Style="{DynamicResource FirstStageLabel}" />
        <TextBox x:Name="VerificationTB"
HorizontalAlignment="Center" Height="31" Margin="0,251,0,0"
TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="312"
FontSize="20" Style="{DynamicResource TextInput}"
Template="{StaticResource Rounded}" />
    </Grid>
</Page>

```

Код разметки меню:

```

<Page x:Class="HRM.Desktop.Pages.Menu.AdminMenu"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:HRM.Desktop.Pages.Menu"
    mc:Ignorable="d"
    d:DesignHeight="50" d:DesignWidth="800"
    Title="AdminMenu" MaxHeight="50" >

    <Grid Background="#FFD4EDF9">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="80"></ColumnDefinition>
            <ColumnDefinition Width="0"></ColumnDefinition>
            <ColumnDefinition Width="80"></ColumnDefinition>
            <ColumnDefinition Width="80"></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition Width="55*"/>
            <ColumnDefinition Width="145"/>
            <ColumnDefinition Width="52"></ColumnDefinition>
            <ColumnDefinition Width="52"/>
        </Grid.ColumnDefinitions>

        <Grid Grid.Column="0">
            <Rectangle RadiusX="15" RadiusY="15" Fill="White"
Stroke="#FF3CA0DC" Margin="2"/>
            <Button x:Name="HomeButton" Background="Transparent"
Margin="5" Foreground="{x:Null}" BorderBrush="{x:Null}"
Style="{DynamicResource ButtonStyle1}" Click="HomeButton_Click">
                <StackPanel>
                    <Image
Source="/HRM.Desktop;component/Images/ButtonImages/home.png"
Stretch="Uniform" Width="40" />
                </StackPanel>
            </Button>
        </Grid>
        <Grid Grid.Column="2">
            <Rectangle RadiusX="15" RadiusY="15" Fill="White"
Stroke="#FF3CA0DC" Margin="2"/>
            <Button x:Name="DepartmentButton" Margin="5"
Background="Transparent" Foreground="{x:Null}" BorderBrush="{x:Null}"
Style="{DynamicResource ButtonStyle1}" Click="DepartmentButton_Click">
                <StackPanel>
                    <Image Stretch="Uniform"
Source="/HRM.Desktop;component/Images/ButtonImages/group.png"
Height="33"/>
                </StackPanel>
            </Button>

```

```

        </Grid>
        <Grid Grid.Column="3">
            <Rectangle RadiusX="15" RadiusY="15" Fill="White"
Stroke="#FF3CA0DC" Margin="2"/>
            <Button x:Name="DocumentsButton"
Background="Transparent" Margin="5" BorderBrush="{x:Null}"
Foreground="{x:Null}" Style="{DynamicResource ButtonStyle1}"
Click="DocumentsButton_Click">
                <StackPanel>
                    <Image Stretch="Uniform"
Source="/HRM.Desktop;component/Images/ButtonImages/folder.png"
Height="42"/>
                </StackPanel>
            </Button>
        </Grid>

        <Button x:Name="SettingsButton" Grid.Column="7" Margin="5"
BorderBrush="{x:Null}" Foreground="{x:Null}" Style="{DynamicResource
ButtonStyle1}" Click="SettingsButton_Click">
            <Button.Background>
                <ImageBrush
ImageSource="/HRM.Desktop;component/Images/ButtonImages/settings.png"
Stretch="Uniform"/>
            </Button.Background>
        </Button>

        <Label x:Name="NameLabel" Grid.Column="6"
Content="{Binding: Name}" VerticalContentAlignment="Center"
HorizontalAlignment="Right" Style="{DynamicResource
FirstStageLabel}" ></Label>
        <Button x:Name="LogOutButton" Grid.Column="8" Width="40"
Margin="0,5,7,5" HorizontalAlignment="Right" Foreground="{x:Null}"
BorderBrush="#FF3CA0DC" BorderThickness="5" Style="{DynamicResource
ButtonStyle1}" Click="LogOutButton_Click">
            <Button.Resources>
                <Style TargetType="Border">
                    <Setter Property="CornerRadius" Value="15"/>
                </Style>
            </Button.Resources>
            <Button.Background>
                <ImageBrush
ImageSource="/HRM.Desktop;component/Images/ButtonImages/logout.png"
Stretch="Uniform"/>
            </Button.Background>
        </Button>

    </Grid>
</Page>

```

Код меню:

```
using HRM.Desktop.Model;
using HRM.Desktop.Pages.AdminNavigation;
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace HRM.Desktop.Pages.Menu
{
    /// <summary>
    /// Логика взаимодействия для AdminMenu.xaml
    /// </summary>
    public partial class AdminMenu : Page
    {
        readonly MainWindow window;
        Employee loggedUser;
        public AdminMenu(MainWindow mainWindow, Employee user)
        {
            InitializeComponent();
            window = mainWindow;
            loggedUser = user;

            NameLabel.Content = user.Passport.Name == null ?
loggedUser.Authorizations.FirstOrDefault().Username
loggedUser.Passport.Name;
        }
        public AdminMenu(MainWindow mainWindow)
        {
            InitializeComponent();
            window = mainWindow;
        }

        private void LogOutButton_Click(object sender,
RoutedEventArgs e)
        {
            loggedUser = null;
            window.MainFrame.Navigate(new
AuthorizationPage(window));
            window.MenuFrame.Navigate(null);
        }

        private void HomeButton_Click(object sender,
RoutedEventArgs e)
        {
            window.MainFrame.Navigate(new
AdminNavigation.AdminIndex(window));
        }

        private void DepartmentButton_Click(object sender,
RoutedEventArgs e)
```

```

        {
            window.MainFrame.Navigate(new
EmployeeAchievementsPage(window));
        }

        private void DocumentsButton_Click(object sender,
RoutedEventArgs e)
        {
            window.MainFrame.Navigate(new
DocumentDownloadPage());
        }

        private void SettingsButton_Click(object sender,
RoutedEventArgs e)
        {
            window.MainFrame.Navigate(new SettingsPage(window));
        }
    }
}

```

Код разметки страницы AdminIndex:

```

<Page x:Class="HRM.Desktop.Pages.AdminNavigation.AdminIndex"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:local="clr-
namespace:HRM.Desktop.Pages.AdminNavigation"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800"
Title="AdminIndex">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="40" MinHeight="36"
MaxHeight="36"/>
            <RowDefinition Height="89*"/>
            <RowDefinition Height="61*"/>
            <RowDefinition Height="30*"/>
        </Grid.RowDefinitions>
    </Grid>

```



```

        <Label x:Name="HLabel" Margin="15,0,0,0" Grid.Row="0"
Height="32" VerticalAlignment="Center" HorizontalAlignment="Left"
Width="217" Style="{DynamicResource FirstStageLabel}"></Label>

```

```

        <Button x:Name="CurrentStatisticsButton"
Background="Transparent" Grid.Column="0" Grid.Row="1" Width="150"
Margin="59,0,58,10" HorizontalAlignment="Center"
Style="{DynamicResource ButtonStyle1}" Height="150"
VerticalAlignment="Bottom" Click="CurrentStatisticsButton_Click">

```

```

        <StackPanel>
            <Image
Source="/HRM.Desktop;component/Images/ButtonImages/statistics.png"/>
        </StackPanel>

```

```

    </Button>
    <TextBlock Grid.Column="0" Grid.Row="2"
Margin="10,15,10,0" VerticalAlignment="Top" Text="Посмотреть статистику
текущего периода" Height="106" TextWrapping="Wrap"
TextAlignment="Center" HorizontalAlignment="Center" Width="247"
Style="{DynamicResource UnfocusedTextBlock}"/>

```

```

        <Button x:Name="CurrentDocumentsButton"
Background="Transparent" Grid.Column="1" Grid.Row="1"
Margin="60,0,57,10" Style="{DynamicResource ButtonStyle1}"
HorizontalAlignment="Center" Width="150" Height="150"
VerticalAlignment="Bottom" Click="CurrentDocumentsButton_Click">

```

```

        <StackPanel>
            <Image
Source="/HRM.Desktop;component/Images/ButtonImages/document.png"/>
        </StackPanel>

```

```

    </Button>
    <TextBlock Grid.Column="1" Grid.Row="2"
Margin="10,15,10,0" VerticalAlignment="Top" Text="Дополнительные
соглашения текущего периода" Height="106" AllowDrop="True"
TextWrapping="Wrap" TextAlignment="Center" HorizontalAlignment="Center"
Width="247" Style="{DynamicResource UnfocusedTextBlock}"/>

```

```

        <Button x:Name="NextPeriodOrganizerButton"
Background="Transparent" Grid.Column="2" Grid.Row="1" Width="150"
Margin="59,0,57,10" HorizontalAlignment="Center" Height="150"
VerticalAlignment="Bottom" Style="{DynamicResource ButtonStyle1}"
Click="NextPeriodOrganizerButton_Click">

```

```

        <StackPanel>
            <Image
Source="/HRM.Desktop;component/Images/ButtonImages/stoncs.png"/>
        </StackPanel>

```

```

    </Button>
    <TextBlock Grid.Column="2" Grid.Row="2"
Margin="10,15,10,0" VerticalAlignment="Top" Text="Установить курс на
следующий период" Height="106" AllowDrop="True" TextWrapping="Wrap"

```

```

TextAlignment="Center"      HorizontalAlignment="Center"      Width="246"
Style="{DynamicResource UnfocusedTextBlock}"/>
    </Grid>
</Page>

```

Код страницы AdminIndex:

```

using HRM.Desktop.Model;
using System.Windows;
using System.Windows.Controls;

namespace HRM.Desktop.Pages.AdminNavigation
{
    /// <summary>
    /// Логика взаимодействия для AdminIndex.xaml
    /// </summary>
    public partial class AdminIndex : Page
    {
        readonly MainWindow window;
        Employee loggedInUser;
        public AdminIndex(MainWindow mainWindow, Employee user)
        {
            InitializeComponent();
            window = mainWindow;
            loggedInUser = user;
            window.currentPage = "AdminPages";
            window.sizeController.CheckWindowSize("AdminIndex");
            HLabel.Content = $"Здравтвуйте
{(user.Passport.Name)}";
        }
        public AdminIndex(MainWindow mainWindow)
        {
            InitializeComponent();
            window = mainWindow;
            window.currentPage = "AdminPages";
            window.sizeController.CheckWindowSize("AdminIndex");
        }

        private void CurrentStatisticsButton_Click(object sender,
RoutedEventArgs e)
        {
            window.MainFrame.Navigate(new
CurrentStatisticsPage(window));
        }

        private void CurrentDocumentsButton_Click(object sender,
RoutedEventArgs e)
        {
            window.error.CheckSettingsSaved();
            window.MainFrame.Navigate(new
DocumentDownloadPage());
        }
    }
}

```

```

        }

        private void NextPeriodOrganizerButton_Click(object
sender, RoutedEventArgs e)
        {
            window.MainFrame.Navigate(new
CreateNextPeriodPage(window));
        }
    }
}

```

Код разметки страницы статистики:

```

<Page
x:Class="HRM.Desktop.Pages.AdminNavigation.CurrentStatisticsPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:lvc="clr-
namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"
    xmlns:local="clr-
namespace:HRM.Desktop.Pages.AdminNavigation"
    mc:Ignorable="d"
    d:DesignHeight="400" d:DesignWidth="800"
    Title="CurrentStatisticsPage">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="3*"/>
            <RowDefinition Height="17*"/>
            <RowDefinition Height="3*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <ComboBox x:Name="PeriodCB" FontSize="18" Grid.Row="0"
Grid.Column="0" HorizontalAlignment="Center" Margin="11,10,10,0"
VerticalAlignment="Top" Height="48" Width="246"
HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
Background="#FF3CA0DC" SelectionChanged="PeriodCB_SelectionChanged"/>
        <Label x:Name="CompanyWorkTimeLabel" Grid.Row="1"
Grid.Column="0" Margin="5,0" Content="Цель на текущий период:"
VerticalAlignment="Top" Height="70" HorizontalContentAlignment="Center"

```

```

VerticalContentAlignment="Center" Background="{x:Null}"
Style="{DynamicResource FirstStageLabel}"/>
        <lvc:CartesianChart Name="PeriodLineChart"
Series="{Binding SeriesCollection}" LegendLocation="Right"
Grid.Column="0" Grid.Row="1" Margin="0,47,0,0" FontSize="14">
        <lvc:CartesianChart.AxisY>
            <lvc:Axis Title="Занятость"></lvc:Axis>
        </lvc:CartesianChart.AxisY>
        <lvc:CartesianChart.AxisX>
            <lvc:Axis Title="Период" Labels="{Binding
PeriodChartLabels}"></lvc:Axis>
        </lvc:CartesianChart.AxisX>
    </lvc:CartesianChart>
    <Label x:Name="PeriodStatsLabel" Grid.Row="2"
Grid.Column="0" Margin="5" Content=":"
HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
Background="#FF3CA0DC" Style="{DynamicResource FirstStageLabel}"
Template="{StaticResource LabelRounded}"/>

        <Label x:Name="DepartmentStatsLabel" Grid.Row="0"
Grid.Column="1" Content="Статистика отделов:"
HorizontalContentAlignment="Center" Margin="10,10,10,0"
VerticalContentAlignment="Top" Height="48" Width="247"
HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
Style="{DynamicResource FirstStageLabel}" Background="#FF3CA0DC"
Template="{StaticResource LabelRounded}"/>
        <lvc:PieChart Name="DepartmentPie" Grid.Row="1"
Grid.Column="1" LegendLocation="Bottom" FontSize="14"></lvc:PieChart>
        <Label x:Name="STDDepartmentLabel" Grid.Row="2"
Grid.Column="1" Margin="5" Content=":"
HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
Background="#FF3CA0DC" Style="{DynamicResource FirstStageLabel}"
Template="{StaticResource LabelRounded}"/>

        <Label x:Name="EmployeeStatsLabel" Grid.Row="0"
Grid.Column="2" Content="Статистика сотрудников:"
HorizontalContentAlignment="Center" Margin="10,10,10,0"
VerticalContentAlignment="Top" Height="48" Width="246"
HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
Style="{DynamicResource FirstStageLabel}" Background="#FF3CA0DC"
Template="{StaticResource LabelRounded}"/>
        <DataGrid x:Name="EmployeeDG" Style="{StaticResource
RoundedDataGrid}" Grid.Row="1" Grid.Column="2" Margin="5,0"
Background="#FFEEF9FF" AutoGenerateColumns="False"
CanUserAddRows="False" FontSize="14">
            <DataGrid.Columns>
                <DataGridTextColumn Header="id"
Visibility="Hidden" Binding="{Binding
EmployeeId}"></DataGridTextColumn>

```

```

                <DataGridTextColumn Header="Сотрудник" Width="2*"
Binding="{Binding Employee.Passport.Inits}"></DataGridTextColumn>
                <DataGridTextColumn Header="Занятость" Width="2*"
Binding="{Binding WorkLoadHours}"></DataGridTextColumn>
            </DataGrid.Columns>
        </DataGrid>
        <Label x:Name="STDEmployeeLabel" Grid.Row="2"
Grid.Column="3" Margin="5" Content=":"
HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
Background="#FF3CA0DC" Style="{DynamicResource FirstStageLabel}"
Template="{StaticResource LabelRounded}"/>

    </Grid>
</Page>

```

Код страницы статистики:

```

using HRM.Desktop.Model;
using LiveCharts;
using LiveCharts.Defaults;
using LiveCharts.Wpf;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Controls;

namespace HRM.Desktop.Pages.AdminNavigation
{
    /// <summary>
    /// Логика взаимодействия для CurrentStatisticsPage.xaml
    /// </summary>
    public partial class CurrentStatisticsPage : Page
    {
        readonly MainWindow window;
        Period previousPeriod;
        Period currentPeriod;
        List<Period> periodList;
        public CurrentStatisticsPage(MainWindow mainWindow)
        {
            InitializeComponent();
            window = mainWindow;
            window.currentPage = "PeriodStatisticsPage";

            window.sizeController.CheckWindowSize("PeriodStatisticsPage");
            GeneratePage();

            DataContext = this;
        }
        private void GeneratePage()
        {

```

```

        FillPeriodsCB();
        GetPeriodContent();
        GetDepartmentContent();
        GetEmployeeContent();
    }
    private void FillPeriodsCB()
    {
        var periodsResponse = window.client.GetAsync(new
Uri("https://localhost:44355/api/Statistics/Period")).Result;
        var periodsResponseContent =
(List<Period>)JsonConvert.DeserializeObject(periodsResponse.Content.Re
adAsStringAsync().Result, typeof(List<Period>));
        periodList = periodsResponseContent;
        List<string> PeriodCBsource = new List<string>();
        foreach (var i in periodsResponseContent)
        {
            PeriodCBsource.Add(new DateTime(i.Year, i.Month,
1).Date.ToString("Y"));
        }
        PeriodCB.ItemsSource = PeriodCBsource;

        PeriodCB.SelectedIndex = PeriodCB.Items.Count - 1;
    }
    public SeriesCollection PeriodSeriesCollection { get; set;
}

    public string[] PeriodChartLabels { get; set; }
    public Func<double, string> YFormatter { get; set; }
    private void GetPeriodContent()
    {
        var periodResponse = window.client.GetAsync(new
Uri("https://localhost:44355/api/Statistics/Period/Current")).Result;
        var periodResponseContent =
(Period)JsonConvert.DeserializeObject(periodResponse.Content.ReadAsStr
ingAsync().Result, typeof(Period));

        var previousPeriodResponse =
window.client.GetAsync(new
Uri("https://localhost:44355/api/Statistics/Period/Previous")).Result;
        var previousPeriodResponseContent =
(Period)JsonConvert.DeserializeObject(previousPeriodResponse.Content.R
eadAsStringAsync().Result, typeof(Period));

        currentPeriod = periodResponseContent;
        CompanyWorkTimeLabel.Content = $"Цель на текущий
период: {currentPeriod.TotalWorkLoadHours}";
        previousPeriod = previousPeriodResponseContent;
        if (previousPeriod != null)
        {
            int diff = currentPeriod.TotalWorkLoadHours -
previousPeriod.TotalWorkLoadHours;
            if (diff == 0)

```

```

        PeriodStatsLabel.Content = "Столько же часов,
сколько и в прошлом месяце";
    else
        PeriodStatsLabel.Content = $"На {(diff > 0 ?
diff + " часов больше, чем в прошлом месяце" : -diff + " часов меньше,
чем в прошлом месяце")}";
    }
    else
        PeriodStatsLabel.Content = "Это первый
зафиксированный период";
        PeriodCB.SelectedItem = new
DateTime(currentPeriod.Year,
currentPeriod.Month,
1).Date.ToString("Y");

        PeriodSeriesCollection = new SeriesCollection();
        var PeriodList = periodList.Where(x => x.PeriodId <=
currentPeriod.PeriodId).ToArray();
        PeriodSeriesCollection.Add(new LineSeries() { Values =
new
ChartValues<double>(PeriodList.Select(x
=>
(Double)(x.TotalWorkLoadHours)).ToArray()), PointGeometry =
DefaultGeometries.Circle, Title = null });
        //YFormatter = value => value.ToString("");
        PeriodChartLabels = new string[PeriodList.Length];

        for (int i = 0; i < PeriodList.Length; i++)
        {
            PeriodChartLabels[i] = "" + new
DateTime(PeriodList[i].Year,
PeriodList[i].Month,
1).Date.ToString("Y");
        }

        PeriodLineChart.Series = PeriodSeriesCollection;
    }
    private void GetPeriodContent(Period period)
    {
        currentPeriod = period;
        CompanyWorkTimeLabel.Content = $"Цель на период:
{currentPeriod.TotalWorkLoadHours}";
        if (previousPeriod != null)
        {
            int diff = currentPeriod.TotalWorkLoadHours -
previousPeriod.TotalWorkLoadHours;
            if (diff == 0)
                PeriodStatsLabel.Content = "Столько же часов,
сколько и в прошлом месяце";
            else
                PeriodStatsLabel.Content = $"На {(diff > 0 ?
diff + " часов больше, чем в прошлом месяце" : -diff + " часов меньше,
чем в прошлом месяце")}";
        }
        else

```

```

        PeriodStatsLabel.Content = "Это первый  
зафиксированный период";

```

```

        PeriodSeriesCollection = new SeriesCollection();
        var PeriodList = periodList.Where(x => x.PeriodId <=
currentPeriod.PeriodId).ToArray();
        PeriodSeriesCollection.Add(new LineSeries() { Values =
new ChartValues<double>(PeriodList.Select(x =>
(Double)(x.TotalWorkLoadHours)).ToArray()), PointGeometry =
DefaultGeometries.Circle, Title = null });
        //YFormatter = value => value.ToString("");
        PeriodChartLabels = new string[PeriodList.Length];
        for (int i = 0; i < PeriodList.Length; i++)
        {
            PeriodChartLabels[i] = "" + new
DateTime(PeriodList[i].Year, PeriodList[i].Month,
1).Date.ToString("Y");
        }
        PeriodLineChart.Series = PeriodSeriesCollection;
    }
    public SeriesCollection DepartmentSeriesCollection { get;
set; }

    private void GetDepartmentContent()
    {
        var currentDepWorkLoadResponse =
window.client.GetAsync(new
Uri("https://localhost:44355/api/Statistics/WorkLoad/Department/" +
currentPeriod.PeriodId)).Result;
        var periodResponseContent =
(List<DepartmentWorkLoad>)JsonConvert.DeserializeObject(currentDepWork
LoadResponse.Content.ReadAsStringAsync().Result,
typeof(List<DepartmentWorkLoad>));

        var current_Department_Work_Loads =
periodResponseContent;
        DepartmentSeriesCollection = new SeriesCollection();
        foreach (var dep in current_Department_Work_Loads)
        {
            DepartmentSeriesCollection.Add(new PieSeries() {
Title = dep.Department.Direction, Values = new
ChartValues<ObservableValue> { new ObservableValue(dep.WorkLoad) } });
        }
        DepartmentPie.Series = DepartmentSeriesCollection;
        var STDDepartmentWorkLoad =
current_Department_Work_Loads.Sum(x => x.WorkLoad);
        STDDepartmentLabel.Content = $"Средняя занятость
отдела:{STDDepartmentWorkLoad / (current_Department_Work_Loads.Count ==
0 ? 1 : current_Department_Work_Loads.Count)}";

```



```

    }
    private void GetEmployeeContent()
    {
        var currentEmpWorkLoadResponse =
window.client.GetAsync(new
Uri("https://localhost:44355/api/Statistics/WorkLoad/Employee/"
currentPeriod.PeriodId)).Result;
        var periodResponseContent =
(List<EmployeeWorkLoad>)JsonConvert.DeserializeObject(currentEmpWorkLo
adResponse.Content.ReadAsStringAsync().Result,
typeof(List<EmployeeWorkLoad>));

        var current_Employee_Work_Loads =
periodResponseContent;
        EmployeeDG.ItemsSource = current_Employee_Work_Loads;
        var STDEmployeeWorkLoad =
current_Employee_Work_Loads.Sum(x => x.WorkLoadHours);
        STDEmployeeLabel.Content = $"Средняя занятость
сотрудника:{STDEmployeeWorkLoad / (current_Employee_Work_Loads.Count ==
0 ? 1 : current_Employee_Work_Loads.Count)}";
    }

    private void PeriodCB_SelectionChanged(object sender,
SelectionChangedEventArgs e)
    {
        int Id = PeriodCB.SelectedIndex + 1;
        var periodResponse = window.client.GetAsync(new
Uri(@"https://localhost:44355/api/Statistics/Period/" + Id)).Result;
        var periodResponseContent =
(Period)JsonConvert.DeserializeObject(periodResponse.Content.ReadAsStr
ingAsync().Result, typeof(Period));

        var previousPeriodResponse =
window.client.GetAsync(new
Uri(@"https://localhost:44355/api/Statistics/Period/" + (Id -
1))).Result;
        var previousPeriodResponseContent =
(Period)JsonConvert.DeserializeObject(previousPeriodResponse.Content.R
eadAsStringAsync().Result, typeof(Period));
        previousPeriod = previousPeriodResponseContent;
        currentPeriod = periodResponseContent;
        GetPeriodContent(currentPeriod);
        GetDepartmentContent();
        GetEmployeeContent();
    }
}
}

```

Код разметки страницы распределения рабочей нагрузки:

```

    <Page
x:Class="HRM.Desktop.Pages.AdminNavigation.CreateNextPeriodPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-
namespace:HRM.Desktop.Pages.AdminNavigation"
    xmlns:model="clr-namespace:HRM.Desktop.Model"
    mc:Ignorable="d"
    d:DesignHeight="400" d:DesignWidth="800"
    Title="CreateNextPeriodPage">
    <Page.Resources>
        <local:DepartmentCollection x:Key="DepartmentList">
            <model:Department DepartmentId="1" Direction="Tex
поддержка"/>
            <model:Department DepartmentId="2"
Direction="Обслуживание"/>
        </local:DepartmentCollection>
    </Page.Resources>

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="4*"/>
            <RowDefinition Height="3*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
        </Grid.RowDefinitions>

        <Label x:Name="NextPeriodLabel" Grid.Row="0"
Margin="50,10,0,0" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" Style="{DynamicResource
FirstStageLabel}"/>

        <Rectangle Grid.Row="1" Grid.Column="0" Margin="5"
Stroke="#FF3CA0DC" RadiusX="20" RadiusY="20"
Fill="#FFEEF9FF"/></Rectangle>
        <Label Content="Нагрузка в часax: "
HorizontalAlignment="Left" Margin="13,20,0,0" Grid.Row="1"
VerticalAlignment="Top" Style="{DynamicResource FirstStageLabel}"/>
        <Label x:Name="LWorkLoad" Content="{Binding
ElementName=WorkLoadSlider,Path=Value}" HorizontalAlignment="Left"

```

```

Height="34" Margin="171,20,0,0" Grid.Row="1" VerticalAlignment="Top"
Width="86" Background="{x:Null}" BorderBrush="#FF3CA0DC"
Style="{DynamicResource FirstStageLabel}"/>
        <Button x:Name="CreatePeriodButton" Content="Сформировать"
Grid.Column="2" Grid.Row="1" Width="150" Height="30" FontSize="18"
Margin="58,110,58,10" Background="#FF3CA0DC" Foreground="Black"
BorderBrush="{x:Null}" Click="CreatePeriodButton_Click"
Style="{DynamicResource ButtonStyle2}" />

        <DataGrid Style="{DynamicResource RoundedDataGrid}"
x:Name="DepartmentContextDG" Grid.Row="1" Grid.Column="1" Margin="5"
AutoGenerateColumns="False" CanUserAddRows="True"
Background="#FFEEF9FF">
            <DataGrid.Columns>
                <DataGridTextColumn Header="Id" Binding="{Binding
DepartmentId}" Visibility="Hidden"/>
                <DataGridComboBoxColumn Header="Отдел" Width="2*"
ItemsSource="{StaticResource DepartmentList}"
DisplayMemberPath="Direction" TextBinding="{Binding DepartmentTitle}"
/>
                <DataGridTextColumn Header="Нагрузка"
Binding="{Binding StaticHours}" Width="*/>
            </DataGrid.Columns>
        </DataGrid>

        <Button x:Name="StatisticsButton" Background="Transparent"
IsEnabled="False" Grid.Row="2" VerticalAlignment="Center"
HorizontalAlignment="Center" Style="{DynamicResource ButtonStyle1}"
Width="100" Height="100" Click="StatisticsButton_Click">
            <StackPanel>
                <Image
Source="/HRM.Desktop;component/Images/ButtonImages/statistics.png"/>
            </StackPanel>
        </Button>
        <Label Grid.Row="3" Grid.Column="0" Content="Распределение
часов" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" Style="{DynamicResource
FirstStageLabel}" Foreground="#FF3CA0DC"/>

        <Button x:Name="DocumentsButton" Background="Transparent"
IsEnabled="False" Grid.Column="1" Grid.Row="2"
VerticalAlignment="Center" HorizontalAlignment="Center"
Style="{DynamicResource ButtonStyle1}" Width="100" Height="100"
Click="DocumentsButton_Click" >
            <StackPanel>
                <Image
Source="/HRM.Desktop;component/Images/ButtonImages/document.png"/>
            </StackPanel>
        </Button>

```

```

        <Label Grid.Row="3" Grid.Column="1" Content="Доп.
соглашения" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" Style="{DynamicResource
FirstStageLabel}" Foreground="#FF3CA0DC"/>
        <Slider x:Name="WorkLoadSlider" Margin="10,60,10,0"
Grid.Row="1" VerticalAlignment="Top" TickPlacement="BottomRight"/>
    </Grid>
</Page>

```

Код страницы распределения рабочей нагрузки:

```

using HRM.Desktop.Commands;
using HRM.Desktop.Model;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Windows;
using System.Windows.Controls;

namespace HRM.Desktop.Pages.AdminNavigation
{
    /// <summary>
    /// Логика взаимодействия для CreateNextPeriodPage.xaml
    /// </summary>
    public partial class CreateNextPeriodPage : Page
    {
        readonly MainWindow window;
        public List<DistributionOption> options;
        Period currentPeriod;
        Period newPeriod;
        public CreateNextPeriodPage(MainWindow mainWindow)
        {
            InitializeComponent();
            window = mainWindow;
            GetPeriodContext();

            options = new List<DistributionOption>();
            //options.Add(new DistributionOption());
            DepartmentContextDG.ItemsSource = options;
            var activeEmployeeResponse =
window.client.GetAsync(new
Uri("https://localhost:44355/api/Employee")).Result;
            var activeEmployeeResponseContent =
(List<Employee>)JsonConvert.DeserializeObject(activeEmployeeResponse.C
ontent.ReadAsStringAsync().Result, typeof(List<Employee>));

```

```

        int EmployeeCount =
activeEmployeeResponseContent.Count;
        WorkLoadSlider.Minimum = EmployeeCount * 4 * 22; //4
        часа в день, 22 рабочих дня в среднем в месяц
        WorkLoadSlider.Maximum = EmployeeCount * 10 * 22; //10
        часов
        WorkLoadSlider.TickFrequency = EmployeeCount * 22 *
0.25; //Интервал по 15 минут рабочего времени в день
        WorkLoadSlider.SmallChange = EmployeeCount * 22 * 0.25;
        WorkLoadSlider.IsSnapToTickEnabled = true;
    }
    private int WorkDays(Period period)
    {
        int workDays = DateTime.DaysInMonth(period.Year,
period.Month);
        for (int i = 1; i < DateTime.DaysInMonth(period.Year,
period.Month) - 1; i++)
        {
            DateTime dt = new DateTime(period.Year,
period.Month, i);
            if (dt.DayOfWeek == DayOfWeek.Saturday ||
dt.DayOfWeek == DayOfWeek.Sunday)
                workDays--;
        }
        return workDays;
    }
    private void GetPeriodContext()
    {
        var currentPeriodResponse = window.client.GetAsync(new
Uri("https://localhost:44355/api/Statistics/Period/Current")).Result;
        var currentPeriodResponseContent =
(Period)JsonConvert.DeserializeObject(currentPeriodResponse.Content.Re
adAsStringAsync().Result, typeof(Period));

        var nextPeriodResponse = window.client.GetAsync(new
Uri("https://localhost:44355/api/Statistics/Period/Next")).Result;
        var nextPeriodResponseContent =
(Period)JsonConvert.DeserializeObject(nextPeriodResponse.Content.ReadA
sAsStringAsync().Result, typeof(Period));

        currentPeriod = currentPeriodResponseContent;
        newPeriod = nextPeriodResponseContent;
        if (newPeriod == null)
        {
            newPeriod = new Period();
            newPeriod.Year = currentPeriod.Month == 12 ?
currentPeriod.Year + 1 : currentPeriod.Year;
            newPeriod.Month = currentPeriod.Month == 12 ? 1 :
currentPeriod.Month + 1;
        }
    }

```

```

        else
        {
            StatisticsButton.IsEnabled = true;
            DocumentsButton.IsEnabled = true;
            CreatePeriodButton.IsEnabled = false;
        }
        WorkLoadSlider.Value = newPeriod.TotalWorkLoadHours ==
0 ? currentPeriod.TotalWorkLoadHours : newPeriod.TotalWorkLoadHours;
        NextPeriodLabel.Content = new DateTime(newPeriod.Year,
newPeriod.Month, 1).Date.ToString("Y");
    }

    private void StatisticsButton_Click(object sender,
RoutedEventArgs e)
    {
        window.MainFrame.Navigate(new
CurrentStatisticsPage(window));
    }

    private void CreatePeriodButton_Click(object sender,
RoutedEventArgs e)
    {
        int WorkLoad = (int)Math.Round(WorkLoadSlider.Value);

        if (options.Count != 0)
        {
            var departmnetResponse =
window.client.GetAsync(new
Uri("https://localhost:44355/api/Statistics/Department")).Result;

            List<Department> departments =
(List<Department>)JsonConvert.DeserializeObject(departmnetResponse.Con
tent.ReadAsStringAsync().Result, typeof(List<Department>));
            foreach (var opt in options)
            {
                opt.DepartmentId = departments.Where(x =>
x.Direction == opt.DepartmentTitle).Select(x =>
x.DepartmentId).FirstOrDefault();

                var stringContent = new
StringContent(JsonConvert.SerializeObject(new
Commands.CreateDistributionCommand { MonthlyHours = WorkLoad, Options =
options }), Encoding.UTF8, "application/json");
                var distributionResponse =
window.client.PostAsync(new
Uri("https://localhost:44355/api/Distribution"), stringContent).Result;
                if (distributionResponse.StatusCode !=
System.Net.HttpStatusCode.OK)
                    window.error.DBError();
            }
        }
    }
}

```

```

        else
        {
            var stringContent = new
StringContent(JsonConvert.SerializeObject(new
Commands.CreateDistributionCommand { MonthlyHours = WorkLoad, Options =
null })), Encoding.UTF8, "application/json");
            var distributionResponse =
window.client.PostAsync(new
Uri("https://localhost:44355/api/Distribution"), stringContent).Result;
            if (distributionResponse.StatusCode !=
System.Net.HttpStatusCode.OK)
                window.error.DBError();
        }
    }

    private void DocumentsButton_Click(object sender,
RoutedEventArgs e)
    {
        window.error.CheckSettingsSaved();
    }
}

public class DepartmentCollection :
ObservableCollection<Department>
{
}
}

```

Код разметки страницы настроек системы:

```

<Page x:Class="HRM.Desktop.Pages.AdminNavigation.SettingsPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-
namespace:HRM.Desktop.Pages.AdminNavigation"
    mc:Ignorable="d"
    d:DesignHeight="400" d:DesignWidth="800"
    Title="SettingsPage">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition Height="7*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>

```

```

        </Grid.ColumnDefinitions>
        <Rectangle Grid.Column="0" Margin="10" Stroke="#FF3CA0DC"
Grid.RowSpan="2" RadiusX="20" RadiusY="20" Fill="#FFEEF9FF"/>
        <TextBlock Grid.Column="0" Grid.Row="0"
Margin="15,15,15,0" VerticalAlignment="Top" Text="Данные о компании"
TextAlignment="Center" Style="{DynamicResource UnfocusedTextBlock}"
Height="32"/>

        <Rectangle Grid.Column="1" Margin="10" Stroke="#FF3CA0DC"
Grid.RowSpan="2" RadiusX="20" RadiusY="20" Fill="#FFEEF9FF"/>
        <TextBlock Grid.Column="1" Grid.Row="0"
Margin="15,15,15,0" VerticalAlignment="Top" Text="Формирование
документов" TextAlignment="Center" Style="{DynamicResource
UnfocusedTextBlock}" Height="32"/>
        <Label Content="Название" HorizontalAlignment="Left"
Margin="15,10,0,0" Grid.Row="1" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}"/>
        <Label Content="Директор" HorizontalAlignment="Left"
Margin="15,49,0,0" Grid.Row="1" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}"/>
        <Label Content="Адрес" HorizontalAlignment="Left"
Margin="15,88,0,0" Grid.Row="1" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}"/>
        <Label Content="ИИН" HorizontalAlignment="Left"
Margin="15,127,0,0" Grid.Row="1" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}"/>
        <Label Content="КПП" HorizontalAlignment="Left"
Margin="15,166,0,0" Grid.Row="1" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}"/>
        <Label Content="БИК" HorizontalAlignment="Left"
Margin="15,205,0,0" Grid.Row="1" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}"/>
        <Label Content="P/C" HorizontalAlignment="Left"
Margin="15,10,0,0" Grid.Row="1" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}" Grid.Column="1"/>
        <Label Content="Банк" HorizontalAlignment="Left"
Margin="15,88,0,0" Grid.Row="1" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}" Grid.Column="1"/>
        <Label Content="K/C" HorizontalAlignment="Left"
Margin="15,49,0,0" Grid.Row="1" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}" Grid.Column="1"/>

        <TextBox x:Name="CompanyNameTB"
VerticalContentAlignment="Center" Height="34" Margin="118,10,27,0"
Grid.Row="1" TextWrapping="NoWrap" VerticalAlignment="Top"
Style="{DynamicResource TextInput}" Template="{StaticResource
Rounded}"/>
        <TextBox x:Name="DirectorTB"
VerticalContentAlignment="Center" Height="34" Margin="118,49,27,0"
Grid.Row="1" TextWrapping="NoWrap" VerticalAlignment="Top"

```



```

Style="{DynamicResource TextInput}" Template="{StaticResource
Rounded}" />
        <TextBox x:Name="AddressTB"
VerticalContentAlignment="Center" Height="34" Margin="118,88,27,0"
Grid.Row="1" TextWrapping="NoWrap" VerticalAlignment="Top"
Style="{DynamicResource TextInput}" Template="{StaticResource
Rounded}" />
        <TextBox x:Name="INNTB" VerticalContentAlignment="Center"
Height="34" Margin="118,127,27,0" Grid.Row="1" TextWrapping="NoWrap"
VerticalAlignment="Top" Style="{DynamicResource TextInput}"
Template="{StaticResource Rounded}" />
        <TextBox x:Name="KPPTB" VerticalContentAlignment="Center"
Height="34" Margin="118,166,27,0" Grid.Row="1" TextWrapping="NoWrap"
VerticalAlignment="Top" Style="{DynamicResource TextInput}"
Template="{StaticResource Rounded}" />
        <TextBox x:Name="BIKTB" VerticalContentAlignment="Center"
Height="34" Margin="118,205,27,0" Grid.Row="1" TextWrapping="NoWrap"
VerticalAlignment="Top" Style="{DynamicResource TextInput}"
Template="{StaticResource Rounded}" />
        <TextBox x:Name="PaymentAccTB"
VerticalContentAlignment="Center" Height="34" Margin="120,10,25,0"
Grid.Row="1" TextWrapping="NoWrap" VerticalAlignment="Top"
Style="{DynamicResource TextInput}" Template="{StaticResource Rounded}"
Grid.Column="1" />
        <TextBox x:Name="CorrAccTB"
VerticalContentAlignment="Center" Height="34" Margin="120,49,25,0"
Grid.Row="1" TextWrapping="NoWrap" VerticalAlignment="Top"
Style="{DynamicResource TextInput}" Template="{StaticResource Rounded}"
Grid.Column="1" />
        <TextBox x:Name="BankTB" VerticalContentAlignment="Center"
Height="34" Margin="120,88,25,0" Grid.Row="1" TextWrapping="NoWrap"
VerticalAlignment="Top" Style="{DynamicResource TextInput}"
Template="{StaticResource Rounded}" Grid.Column="1" />
        <Label Content="Путь к файлам" HorizontalAlignment="Left"
Margin="15,127,0,0" Grid.Row="1" VerticalAlignment="Top"
Style="{DynamicResource FirstStageLabel}" Grid.Column="1" />
        <TextBox x:Name="FilePathTB"
VerticalContentAlignment="Center" Height="34" Margin="145,127,25,0"
Grid.Row="1" TextWrapping="NoWrap" VerticalAlignment="Top"
Style="{DynamicResource TextInput}" Template="{StaticResource Rounded}"
Grid.Column="1" />
        <Button x:Name="OpenDialogButton" Grid.Column="1"
HorizontalAlignment="Right" Margin="0,127,25,0" Grid.Row="1"
VerticalAlignment="Top" Width="30" Height="34" Style="{DynamicResource
ButtonStyle2}" Click="OpenDialogButton_Click" />
        <Button x:Name="SaveButton" Content="Сохранить"
Grid.Column="1" HorizontalAlignment="Left" Margin="275,0,0,26"
Grid.Row="1" VerticalAlignment="Bottom" Width="100" Height="28"
Style="{DynamicResource ButtonStyle2}" Click="SaveButton_Click" />

```

```

        <Button x:Name="BackButton" Content="Назад"
Grid.Column="1" HorizontalAlignment="Left" Margin="170,0,0,26"
Grid.Row="1" VerticalAlignment="Bottom" Width="100" Height="28"
Style="{DynamicResource ButtonStyle2}" Click="BackButton_Click"/>
    </Grid>
</Page>

```

Код страницы настроек системы:

```

using HRM.Desktop.Model;
using Newtonsoft.Json;
using System;
using System.Net.Http;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Forms;

namespace HRM.Desktop.Pages.AdminNavigation
{
    /// <summary>
    /// Логика взаимодействия для SettingsPage.xaml
    /// </summary>
    public partial class SettingsPage : Page
    {
        MainWindow window;
        CompanyData companyData;
        public SettingsPage(MainWindow mainWindow)
        {
            InitializeComponent();
            window = mainWindow;

            var companyDataResponse = window.client.GetAsync(new
Uri("https://localhost:44355/api/CompanyData")).Result;
            var companyDataResponseContent =
(CompanyData)JsonConvert.DeserializeObject(companyDataResponse.Content
.ReadAsStringAsync().Result, typeof(CompanyData));
            companyData = companyDataResponseContent;
            CompanyNameTB.Text = companyData.CompanyName;
            DirectorTB.Text = companyData.DirectorName;
            AddressTB.Text = companyData.CompanyAddress;
            INNTB.Text = companyData.INN;
            KPPTB.Text = companyData.KPP;
            BIKTB.Text = companyData.BIK;
            PaymentAccTB.Text = companyData.PAcc;
            CorrAccTB.Text = companyData.CAcc;
            BankTB.Text = companyData.Bank;
        }
        private void BackButton_Click(object sender,
RoutedEventArgs e)
        {

```

```

        window.MainFrame.GoBack();
    }

    private void SaveButton_Click(object sender,
RoutedEventArgs e)
    {
        companyData.CompanyName = CompanyNameTB.Text;
        companyData.DirectorName = DirectorTB.Text;
        companyData.CompanyAddress = AddressTB.Text;
        companyData.INN = INNTB.Text;
        companyData.KPP = KPPTB.Text;
        companyData.BIK = BIKTB.Text;
        companyData.PAcc = PaymentAccTB.Text;
        companyData.CAcc = CorrAccTB.Text;
        companyData.Bank = BankTB.Text;
        var companyContent = new
StringContent(JsonConvert.SerializeObject(
        new Commands.FillCompanyDataCommand
        {
            Bank = companyData.Bank,
            BIK = companyData.BIK,
            PAcc = companyData.PAcc,
            INN = companyData.INN,
            KPP = companyData.KPP,
            CAcc = companyData.CAcc,
            CompanyAddress = companyData.CompanyAddress,
            CompanyName = companyData.CompanyName,
            DirectorName = companyData.DirectorName
        })), Encoding.UTF8, "application/json");
        var companyDataResponse = window.client.PostAsync(new
Uri("https://localhost:44355/api/Statistics/CompanyData"),
companyContent).Result;
    }

    private void OpenDialogButton_Click(object sender,
RoutedEventArgs e)
    {
        FolderBrowserDialog fileDialog = new
FolderBrowserDialog();
        fileDialog.ShowDialog();
        if (fileDialog.SelectedPath != "")
            FilePathTB.Text = fileDialog.SelectedPath;
    }
}

```

Код разметки страницы работы с документами:

```

<Page
x:Class="HRM.Desktop.Pages.AdminNavigation.DocumentDownloadPage"

```

```

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-
namespace:HRM.Desktop.Pages.AdminNavigation"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="DocumentDownloadPage">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="2*"/>
            <ColumnDefinition Width="5*"/>
        </Grid.ColumnDefinitions>
        <Rectangle Grid.Column="0" Margin="10,10,10,10"
Stroke="#FF3CA0DC" RadiusX="20" RadiusY="20" Fill="#FFEEF9FF"/>

            <ComboBox x:Name="PeriodCB" HorizontalAlignment="Center"
Margin="0,53,0,0" VerticalAlignment="Top" Width="190" Height="35"
SelectionChanged="FilterChanged" FontSize="18"/>
            <ComboBox x:Name="DepartmentCB"
HorizontalAlignment="Center" Margin="0,124,0,0" VerticalAlignment="Top"
Width="190" Height="35" SelectionChanged="FilterChanged"
FontSize="18"/>
            <Label Content="Период" HorizontalAlignment="Center"
Margin="0,19,0,0" VerticalAlignment="Top" Style="{StaticResource
FirstStageLabel}"/>
            <Label Content="Отдел" HorizontalAlignment="Center"
Margin="0,90,0,0" VerticalAlignment="Top" Style="{StaticResource
FirstStageLabel}"/>
            <ComboBox x:Name="FormatCB" HorizontalAlignment="Center"
Margin="0,190,0,0" VerticalAlignment="Top" Width="190" Height="35"
FontSize="18"/>
            <Label Content="Формат" HorizontalAlignment="Center"
Margin="0,156,0,0" VerticalAlignment="Top" Style="{StaticResource
FirstStageLabel}"/>
            <Button Content="Скачать" HorizontalAlignment="Center"
Margin="0,0,0,52" Width="190" Style="{StaticResource ButtonStyle2}"
BorderBrush="#FF3CA0DC" Background="#FFD4EDF9" Height="31"
VerticalAlignment="Bottom"/>
            <DataGrid x:Name="FilesDG" Style="{StaticResource
RoundedDataGrid}" HorizontalAlignment="Left" Height="344"
Margin="10,96,10,0" VerticalAlignment="Top" AutoGenerateColumns="False"
CanUserAddRows="False" Grid.Column="1" FontSize="16"
Background="#FFEEF9FF">
                <DataGrid.Columns>

```

```

        <DataGridTextColumn                                Header="Период"
Binding="{Binding MounthOfYear}" Width="2*"/>
        <DataGridTextColumn                                Header="Отдел"
Binding="{Binding DepartmentDirection}" Width="2*"/>
        <DataGridTextColumn                                Header="Сотрудник"
Binding="{Binding EmployeeInits}" Width="2*"/>
        <DataGridCheckBoxColumn                            Header="Загрузить"
Width="*"/>
    </DataGrid.Columns>
</DataGrid>
    <Label                                Content="Найдено                файлов:"
HorizontalAlignment="Left" Margin="36,19,0,0" VerticalAlignment="Top"
Style="{StaticResource FirstStageLabel}" Grid.Column="1"/>

</Grid>
</Page>

```

Код страницы работы с документами:

```

using System.Collections.Generic;
using System.Linq;
using System.Windows.Controls;

namespace HRM.Desktop.Pages.AdminNavigation
{
    /// <summary>
    /// Логика взаимодействия для DocumentDownloadPage.xaml
    /// </summary>
    public partial class DocumentDownloadPage : Page
    {
        List<DGSource> DGValues;
        public DocumentDownloadPage()
        {
            InitializeComponent();
            FormatCB.ItemsSource = new List<string>() { "DOC",
"DOCX", "PDF", "XML" };
            PeriodCB.ItemsSource = new List<string>() { "за все
время", "апрель 2022", "май 2022" };
            DepartmentCB.ItemsSource = new List<string>() { "все
отделы", "Техподдержка", "Обслуживание" };
            PeriodCB.SelectedIndex = 0;
            DepartmentCB.SelectedIndex = 0;
            FormatCB.SelectedIndex = 0;
            DGValues = new List<DGSource>()
            {
                new DGSource()
                {
                    MounthOfYear = "апрель 2022",
                    DepartmentDirection = "Техподдержка",
                    EmployeeInits = "Яскунова М.Д."
                },
            },
        }
    }
}

```

```

        new DGSource()
        {
            MounthOfYear = "май 2022",
            DepartmentDirection = "Техподдержка",
            EmployeeInits = "Яскунова М.Д."
        },
        new DGSource()
        {
            MounthOfYear = "апрель 2022",
            DepartmentDirection = "Обслуживание",
            EmployeeInits = "Пузанов Ф.С."
        },
        new DGSource()
        {
            MounthOfYear = "май 2022",
            DepartmentDirection = "Обслуживание",
            EmployeeInits = "Яскунова М.Д."
        },
        new DGSource()
        {
            MounthOfYear = "апрель 2022",
            DepartmentDirection = "Техподдержка",
            EmployeeInits = "Ширяев Ф.А."
        },
        new DGSource()
        {
            MounthOfYear = "май 2022",
            DepartmentDirection = "Техподдержка",
            EmployeeInits = "Ширяев Ф.А."
        },
        new DGSource()
        {
            MounthOfYear = "апрель 2022",
            DepartmentDirection = "Обслуживание",
            EmployeeInits = "Цицина Ю.Ф."
        },
        new DGSource()
        {
            MounthOfYear = "май 2022",
            DepartmentDirection = "Обслуживание",
            EmployeeInits = "Цицина Ю.Ф."
        }
    }.OrderBy(x => x.MounthOfYear)
    .OrderBy(x => x.DepartmentDirection)
    .OrderBy(x => x.EmployeeInits).ToList();
    FilesDG.ItemsSource = DGValues;
}

private void FilterChanged(object sender,
SelectionChangedEventArgs e)
{

```

```

        var tempValues = DGValues;
        if (PeriodCB.SelectedIndex > 0)
            tempValues = tempValues.Where(x => x.MounthOfYear
== PeriodCB.SelectedItem.ToString()).ToList();
            if(DepartmentCB.SelectedIndex > 0)
                tempValues
tempValues.Where(x=>x.DepartmentDirection
DepartmentCB.SelectedItem.ToString()).ToList();
                FilesDG.ItemsSource = tempValues;
            }
        }
    internal class DGSource
    {
        public string MounthOfYear { get; set; }
        public string DepartmentDirection { get; set; }
        public string EmployeeInits { get; set; }
    }
}

```

Код разметки страницы достижений сотрудников:

```

<Page
x:Class="HRM.Desktop.Pages.AdminNavigation.EmployeeAchievementsPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-
namespace:HRM.Desktop.Pages.AdminNavigation"
    xmlns:model="clr-namespace:HRM.Desktop.Model"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="EmployeeAchievementsPage">

    <Grid>
        <DataGrid x:Name="AchievementsDG" Style="{StaticResource
RoundedDataGrid}" Margin="10,58,10,10" AutoGenerateColumns="False"
CanUserAddRows="True" FontSize="14"
RowEditEnding="AchievementsDG_RowEditEnding" Background="#FFEEF9FF">
            <DataGrid.Columns>
                <DataGridComboBoxColumn Width="*"
Header="Сотрудник" ItemsSource="{StaticResource EmployeeList}"
DisplayMemberPath="Passport.Inits" SelectedValuePath="EmployeeId"
SelectedValueBinding="{Binding EmployeeId}"/>
                <DataGridTextColumn Width="2*" Header="Описание"
Binding="{Binding Description}"/>
                <DataGridTextColumn Width="*" Header="Премия"
Binding="{Binding Reward}"/>
            </DataGrid.Columns>
        </DataGrid>
    </Grid>

```

```

        </DataGrid.Columns>
    </DataGrid>
    <ComboBox          x:Name="PeriodCB"          FontSize="18"
HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top"
Height="48"          Width="246"          HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" Background="#FF3CA0DC"/>

    </Grid>
</Page>

```

Код страницы достижений сотрудников:

```

using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Net.Http;
using System.Text;
using System.Threading;
using System.Windows.Controls;
using HRM.Desktop.Commands;
using HRM.Desktop.Model;
using Newtonsoft.Json;

namespace HRM.Desktop.Pages.AdminNavigation
{
    /// <summary>
    /// Логика взаимодействия для EmployeeAchievementsPage.xaml
    /// </summary>
    public partial class EmployeeAchievementsPage : Page
    {
        public List<PersonalAchievement> achievements;
        private readonly MainWindow window;
        private List<Period> periodList;
        private List<Employee> employeeList;
        public EmployeeAchievementsPage(MainWindow window)
        {
            var employeeResponse = window.client.GetAsync(new
Uri("https://localhost:44355/api/Employee")).Result;
            var employeeResponseContent =
(List<Employee>)JsonConvert.DeserializeObject(employeeResponse.Content
.ReadAsStringAsync().Result, typeof(List<Employee>));
            employeeList = employeeResponseContent;
            this.Resources.Add("EmployeeList", employeeList);
            InitializeComponent();
            this.window = window;
            var achievementsResponse = window.client.GetAsync(new
Uri("https://localhost:44355/api/Achievements")).Result;
            var achievementsResponseContent =
(List<PersonalAchievement>)JsonConvert.DeserializeObject(achievementsRe
sponse.Content.ReadAsStringAsync().Result,
typeof(List<PersonalAchievement>));

```



```

        achievements = achievementsResponseContent;
        AchievementsDG.ItemsSource = achievements;

        var periodsResponse = window.client.GetAsync(new
Uri("https://localhost:44355/api/Statistics/Period")).Result;
        var periodsResponseContent =
(List<Period>)JsonConvert.DeserializeObject(periodsResponse.Content.Re
adAsStringAsync().Result, typeof(List<Period>));
        periodList = periodsResponseContent;
        List<string> PeriodCBsource = new List<string>();
        foreach (var i in periodList)
        {
            PeriodCBsource.Add(new DateTime(i.Year, i.Month,
1).Date.ToString("Y"));
        }
        PeriodCB.ItemsSource = PeriodCBsource;

        PeriodCB.SelectedIndex = PeriodCB.Items.Count - 1;
    }
    private void AchievementsDG_RowEditEnding(object sender,
DataGridRowEditEndingEventArgs e)
    {
        Timer t = new Timer(new TimerCallback(SaveAchiev));
        t.Change(1000, 0);
    }
    private void SaveAchiev(object state)
    {
        ((Timer)state).Dispose();
        foreach (var achiev in achievements)
        {
            if (achiev.PeriodId == 0)
                Dispatcher.Invoke(() => achiev.PeriodId =
PeriodCB.SelectedIndex + 1); //Обращение к UI элементу в дополнительном
потоке

            var stringContent = new
StringContent(JsonConvert.SerializeObject(
new AddOrUpdatePersonalAchievementCommand()
{
                AchievementId = achiev.AchievementId,
                Description = achiev.Description,
                EmployeeId = achiev.EmployeeId,
                PeriodId = achiev.PeriodId,
                Reward = achiev.Reward
            })), Encoding.UTF8, "application/json");
            var achievementsResponse =
window.client.PostAsync(new
Uri("https://localhost:44355/api/Achievements"), stringContent).Result;

```

```

        var getAchievementsResponse =
window.client.GetAsync(new
Uri("https://localhost:44355/api/Achievements")).Result;
        var achievementsResponseContent =
(List<PersonalAchievement>)JsonConvert.DeserializeObject(getAchievement
sResponse.Content.ReadAsStringAsync().Result,
typeof(List<PersonalAchievement>));

        achievements = achievementsResponseContent;
        AchievementsDG.ItemsSource = achievements;
    }
}
}
public class EmployeeCollection :
ObservableCollection<Employee>
{
}
}

```

### Вспомогательные классы

Класс обработчиков исключений ExceptionsAndMessenges:

```

using System.Windows;

namespace HRM.Desktop.Handlers
{
    public class ExceptionsAndMessenges
    {
        readonly MainWindow window;
        public ExceptionsAndMessenges(MainWindow mainWindow)
        {
            window = mainWindow;
        }
        public void TimerExceptions()
        {
            if (window.currentPage == "AuthorizationPage" ||
window.currentPage == "")
            {
                MessageBox.Show("Причина: длительное бездействие",
"Выход из системы", MessageBoxButton.OK, MessageBoxImage.Information);
                window.Close();
            }
            else
            {
                window.Height = 450;
                window.MainFrame.Navigate(new
Pages.AuthorizationPage(window));
                window.MenuFrame.Navigate(null);
                window.timer.timer.Start();
            }
        }
    }
}

```

```

        MessageBox.Show("Причина: длительное бездействие",
"Выход из системы", MessageBoxButton.OK, MessageBoxImage.Information);
    }
}
public void LackInputExeption()
{
    MessageBox.Show("Все поля должны быть заполнены",
"Ошибка ввода данных", MessageBoxButton.OK, MessageBoxImage.Warning);
}
public void AuthorizationError()
{
    MessageBox.Show("Пользователь с такими данными не
найден", "Ошибка авторизации", MessageBoxButton.OK,
MessageBoxImage.Error);
}
public void DBError()
{
    MessageBox.Show("База данных послала вас на юх",
"Оыпкка стоп 0x000000000000", MessageBoxButton.OK,
MessageBoxImage.Error);
}
public void CheckSettingsSaved()
{
    if (Properties.Settings.Default.CompanyName == "" ||
        Properties.Settings.Default.DirectorName == "" ||
        Properties.Settings.Default.CompanyAddress == ""
||
        Properties.Settings.Default.INN == "" ||
        Properties.Settings.Default.KPP == "" ||
        Properties.Settings.Default.BIK == "" ||
        Properties.Settings.Default.PAcc == "" ||
        Properties.Settings.Default.CAcc == "" ||
        Properties.Settings.Default.Bank == "" ||
        Properties.Settings.Default.SaveFilePath == "")
    {
        if (MessageBoxResult.Yes == MessageBox.Show("В
настройках приложения заполнены не все данные, необходимые для
продолжения. Желаете заполнить их сейчас?", "Необходимые данные не
заполнены", MessageBoxButton.YesNo, MessageBoxImage.Error))
        {
            window.MainFrame.Navigate(new
Pages.AdminNavigation.SettingsPage(window));
        }
    }
}
public void FileCreactionFailed()
{
    MessageBox.Show("Ошибка в процессе создания файлов.
Закройте файл с шаблоном и попробуйте снова ", "Ошибка в процессе
создания файлов", MessageBoxButton.OK, MessageBoxImage.Error);
}

```

```

    }
    public void FormatFileLoss()
    {
        MessageBox.Show("Файл с шаблоном потерян", "Нет
файла", MessageBoxButton.OK, MessageBoxImage.Information);
    }
}

```

Класс обработчик времени бездействия Timer:

```

using System;
using System.Windows.Threading;

namespace HRM.Desktop.Handlers
{
    public class Timer
    {
        readonly MainWindow window;
        public Timer(MainWindow mainWindow)
        {
            window = mainWindow;
        }
        public DispatcherTimer timer = new DispatcherTimer(); //
Таймер афк
        public void StartupTimer()
        {
            timer.Tick += new EventHandler(timerTick);
            timer.Interval = new TimeSpan(0, 1, 0);
            timer.Start();
        }
        public void ResetTimer()
        {
            timer.Stop();
            timer.Interval = new TimeSpan(0, 1, 0);
            timer.Start();
        }
        private void timerTick(object sender, EventArgs e)
        {
            window.error.TimerExeptions();
        }
    }
}

```

Класс обработчик размеров окна приложения WindowSizeController:

```

using System.Collections.Generic;
using System.Linq;

namespace HRM.Desktop.Handlers
{

```

```

public class WindowSizeController
{
    MainWindow window;
    public WindowSizeController(MainWindow mainWindow)
    {
        window = mainWindow;
    }
    List<PageSize> pageslist = new List<PageSize>() { new
    PageSize("AuthorizationPage", false, 800, 450), new
    PageSize("AdminIndex", true, 1200, 650) };
    public void CheckWindowSize(string windowTitle)
    {
        PageSize currentPage = pageslist.Where(x =>
x.WindowTitle == windowTitle).FirstOrDefault();
        if (currentPage == null)
            return;
        if (currentPage.IsChangeble == false)
            window.ResizeMode =
System.Windows.ResizeMode.NoResize;
        else
            window.ResizeMode =
System.Windows.ResizeMode.CanResize;

        if (currentPage.DefaultHeight != 0)
        {
            window.Height = currentPage.DefaultHeight;
            window.WindowStartupLocation =
System.Windows.WindowStartupLocation.CenterScreen;
        }
        if (currentPage.DefaultWidth != 0)
        {
            window.Width = currentPage.DefaultWidth;
            window.WindowStartupLocation =
System.Windows.WindowStartupLocation.CenterScreen;
        }
    }
}
class PageSize
{
    public PageSize(string Title, bool CanChange, double Width,
double Height)
    {
        WindowTitle = Title;
        IsChangeble = CanChange;
        DefaultHeight = Height;
        DefaultWidth = Width;
    }
    public PageSize(string Title, bool CanChange)
    {
        WindowTitle = Title;
        IsChangeble = CanChange;
    }
}

```

```
    }  
  
    public string WindowTitle;  
    public bool IsChangeble;  
    public double DefaultWidth;  
    public double DefaultHeigth;  
    }  
}
```

## Приложение 7. Код тестирования алгоритмов

Код контекста тестовой базы данных:

```
using HRM.Domain;
using HRM.Persistence;
using Microsoft.EntityFrameworkCore;
using System;
using System.IO;

namespace HRM.Tests.Common
{
    public class HRMContextFactory
    {
        public static HRMdbContext Create()
        {
            var option = new DbContextOptionsBuilder<HRMdbContext>()
                .UseInMemoryDatabase(Guid.NewGuid().ToString())
                .Options;
            var context = new HRMdbContext(option);
            context.Database.EnsureCreated();

            context.ContactData.AddRange(
                new ContactData
                {
                    ContactDataId = 1,
                    Email = "email@mail.em",
                    PhoneNumber = "89998441515"
                },
                new ContactData
                {
                    ContactDataId = 2,
                    Email = "email@mail.em",
                    PhoneNumber = "89998441515"
                },
                new ContactData
                {
                    ContactDataId = 3,
                    Email = "email@mail.em",
                    PhoneNumber = "89998441515"
                }
            );
            context.PassportInfos.AddRange(
                new PassportInfo
                {
                    PassportId = 1,
                    PassportSerial = 9131,
                    PassportNumber = 193413,
```

```

        Name = "Name",
        Surname = "Surname",
        State = "State",
        Country = "Country",
        City = "City",
        Street = "Street",
        House = 1,
        Buinding = 1,
        Apartment = 1
    },
    new PassportInfo
    {
        PassportId = 2,
        PassportSerial = 9131,
        PassportNumber = 193413,
        Name = "Name",
        Surname = "Surname",
        State = "State",
        Country = "Country",
        City = "City",
        Street = "Street",
        House = 1,
        Buinding = 1,
        Apartment = 1
    },
    new PassportInfo
    {
        PassportId = 3,
        PassportSerial = 9131,
        PassportNumber = 193413,
        Name = "Name",
        Surname = "Surname",
        State = "State",
        Country = "Country",
        City = "City",
        Street = "Street",
        House = 1,
        Buinding = 1,
        Apartment = 1
    });
context.Candidates.AddRange(
    new Candidate
    {
        ContactDataId = 1,
        PassportId = 1,
        CandidateId = 1,
        Education = "Edu",
        ExpirienseYears = 1
    },
    new Candidate
    {

```



```

        ContactDataId = 2,
        PassportId = 2,
        CandidateId = 2,
        Education = "Edu",
        ExpirienseYears = 1
    },
    new Candidate
    {
        ContactDataId = 3,
        PassportId = 3,
        CandidateId = 3,
        Education = "Edu",
        ExpirienseYears = 1
    });
context.Departments.AddRange(
    new Department
    {
        DepartmentId = 1,
        EmployeeCount = 1,
        Direction = "Direction",
        BasicMoneyPerHour = 300,
        TotalMoneyPerHour = 300
    },
    new Department
    {
        DepartmentId = 2,
        EmployeeCount = 1,
        Direction = "Direction",
        BasicMoneyPerHour = 300,
        TotalMoneyPerHour = 300
    });
context.Interviews.AddRange(
    new Interview
    {
        InterviewId = 1,
        CandidateId = 1,
        Date = DateTime.Now,
        IsPassed = true
    },
    new Interview
    {
        InterviewId = 2,
        CandidateId = 2,
        Date = DateTime.Now,
        IsPassed = true
    });
context.Employees.AddRange(
    new Employee
    {
        EmployeeId = 1,
        DepartmentId = 1,

```

```

        InterviewId = 1,
        PassportId = 1,
        ContactDataId = 1,
        Active = true,
        AuthorizationCode = Guid.NewGuid().ToString()
    },
    new Employee
    {
        EmployeeId = 2,
        DepartmentId = 2,
        InterviewId = 2,
        PassportId = 2,
        ContactDataId = 2,
        Active = true,
        AuthorizationCode = Guid.NewGuid().ToString()
    });
context.Periods.Add(
    new Period
    {
        PeriodId = 1,
        Month = 1,
        Year = 2022,
        TotalWorkLoadHours = 240
    });
context.DepartmentWorkLoads.AddRange(
    new DepartmentWorkLoad
    {
        ScheduleId = 1,
        DepartmentId = 1,
        PeriodId = 1,
        WorkLoad = 120,
        IsEqualOrMore = false,
        WorkedHours = 100
    },
    new DepartmentWorkLoad
    {
        ScheduleId = 2,
        DepartmentId = 2,
        PeriodId = 1,
        WorkLoad = 120,
        WorkedHours = 120,
        IsEqualOrMore = true
    });
context.EmployeeWorkLoads.AddRange(
    new EmployeeWorkLoad
    {
        AddendumId = 1,
        EmployeeId = 1,
        PeriodId = 1,
        WorkLoadHours = 120,
        WorkedHours = 100
    }

```

```

    },
    new EmployeeWorkLoad
    {
        AddendumId = 2,
        EmployeeId = 2,
        PeriodId = 1,
        WorkLoadHours = 120,
        WorkedHours = 140
    });
context.CompanyData.Add(
    new CompanyData
    {
        DirectorName = "Директор",
        CompanyName = "ООО Компания",
        CompanyAddress = "Адресс",
        Bank = "Sberbank",
        BIK = "1234567890",
        CAcc = "1234567890",
        PAcc = "1234567890",
        INN = "1234567890",
        KPP = "1234567890"
    });
byte[] fileBytes;
FileStream file = new
FileStream(@"wwwroot/files/prefab.docx", FileMode.Open,
FileAccess.Read);
using (var ms = new MemoryStream())
{
    file.CopyTo(ms);
    fileBytes = ms.ToArray();
}
context.Files.Add(
    new Domain.File
    {
        Id = 1,
        Name = "prefab.docx",
        Data = fileBytes
    });
context.SaveChanges();
return context;
}
public static void Destroy(HRMDBContext context)
{
    context.Database.EnsureDeleted();
    context.Dispose();
}
}
}

```

Код основы тестовой команды:

```

using HRM.Application.Interfaces;
using HRM.Persistence;
using System;

namespace HRM.Tests.Common
{
    public class TestCommandBase : IDisposable
    {
        protected readonly HRMDBContext Context;
        protected readonly IUnitOfWork unitOfWork;

        public TestCommandBase()
        {
            Context = HRMContextFactory.Create();
            unitOfWork = new UnitOfWork(Context);
        }
        public void Dispose()
        {
            HRMContextFactory.Destroy(Context);
        }
    }
}

```

Тестирование модуля распределения рабочей нагрузки

Класс            тестирования            создания            рабочей            нагрузки

CreateDistributionCommandHandlerTests:

```

using HRM.Application.WorkLoadDistribution;
using HRM.Application.WorkLoadDistribution.CreateDistribution;
using HRM.Tests.Common;
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Threading.Tasks;
using Xunit;

namespace HRM.Tests.Commands.DistributionCommandHandlerTests
{
    public class CreateDistributionCommandHandlerTests
:TestCommandBase
    {
        [Fact]
        public async Task
CreateDistributionCommandHandlerTests_Success1()
        {
            //Arrange
            var handler = new
CreateDistributionCommandHandler(unitOfWork);
            var monthlyHours = 260;

```

```

        //Act
        await handler.Distribute(
            new CreateDistributionCommand
            {
                MonthlyHours = monthlyHours,
                Options = null
            });

        //Assert
        Assert.NotNull(await
Context.Periods.SingleOrDefaultAsync(period =>
    period.TotalWorkLoadHours == monthlyHours));
        Assert.NotNull(await
Context.EmployeeWorkLoads.SingleOrDefaultAsync(load =>
    load.EmployeeId == 1 &&
    load.WorkLoadHours == 130));
    }

    [Fact]
    public async Task
CreateDistributionCommandHandlerTests_Success2()
    {
        //Arrange
        var handler = new
CreateDistributionCommandHandler(unitOfWork);
        var monthlyHours = 0;
        var options = new List<DistributionOption>();
        options.Add(new DistributionOption
        {
            DepartmentId = 1,
            StaticHours = 130
        });
        options.Add(new DistributionOption
        {
            DepartmentId = 2,
            StaticHours = 140
        });

        //Act
        await handler.Distribute(
            new CreateDistributionCommand
            {
                MonthlyHours = monthlyHours,
                Options = options
            });

        //Assert
        Assert.NotNull(await
Context.Periods.SingleOrDefaultAsync(period =>
    period.TotalWorkLoadHours == 270));

```

```

        Assert.NotNull(await
Context.EmployeeWorkLoads.SingleOrDefaultAsync(load =>
    load.EmployeeId == 1 &&
    load.WorkLoadHours == 130));

        Assert.NotNull(await
Context.EmployeeWorkLoads.SingleOrDefaultAsync(load =>
    load.EmployeeId == 2 &&
    load.WorkLoadHours == 140));
    }
}
}

```

Класс тестирования редактирования распределения рабочей нагрузки

UpdateDistributionCommandHandlerTests:

```

using HRM.Application.WorkLoadDistribution;
using HRM.Application.WorkLoadDistribution.UpdateDistribution;
using HRM.Tests.Common;
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Threading.Tasks;
using Xunit;

namespace HRM.Tests.Commands.DistributionCommandHandlerTests
{
    public class UpdateDistributionCommandHandlerTests :
TestCommandBase
    {
        [Fact]
        public async Task
UpdateDistributionCommandHandlerTests_Success1()
        {
            //Arrange
            var handler = new
UpdateDistributionCommandHandler(unitOfWork);
            var monthlyHours = 300;

            //Act
            await handler.UpdateDistribution(
                new UpdateDistributionCommand
                {
                    PeriodId = 1,
                    MonthlyHours = monthlyHours,
                    Options = null
                });

            //Assert
            Assert.NotNull(await
Context.Periods.SingleOrDefaultAsync(period =>

```

```

        period.TotalWorkLoadHours == monthlyHours));
        Assert.NotNull(await
Context.EmployeeWorkLoads.SingleOrDefaultAsync(load =>
        load.EmployeeId == 1 &&
        load.WorkLoadHours == 150));
    }

    [Fact]
    public async Task
UpdateDistributionCommandHandlerTests_Success2()
    {
        //Arrange
        var handler = new
UpdateDistributionCommandHandler(unitOfWork);
        var monthlyHours = 0;

        //Act
        await handler.UpdateDistribution(
            new UpdateDistributionCommand
            {
                PeriodId = 1,
                MonthlyHours = monthlyHours,
                Options = new List<DistributionOption>
                {
                    new DistributionOption
                    {
                        DepartmentId = 1,
                        StaticHours = 140
                    },
                    new DistributionOption
                    {
                        DepartmentId = 2,
                        StaticHours = 160
                    }
                }
            });

        //Assert
        Assert.NotNull(await
Context.Periods.SingleOrDefaultAsync(period =>
        period.TotalWorkLoadHours == 300));
        Assert.NotNull(await
Context.EmployeeWorkLoads.SingleOrDefaultAsync(load =>
        load.EmployeeId == 1 &&
        load.WorkLoadHours == 140));
        Assert.NotNull(await
Context.EmployeeWorkLoads.SingleOrDefaultAsync(load =>
        load.EmployeeId == 2 &&
        load.WorkLoadHours == 160));
    }
}

```

```
}
```

Код тестирования создания дополнительных соглашений

GenerateAddendumCommandHandlerTests:

```
using HRM.Application.WorkLoadDistribution.GenerateAddendum;
using HRM.Tests.Common;
using System.Linq;
using System.Threading.Tasks;
using Xunit;

namespace HRM.Tests.Commands.DistributionCommandHandlerTests
{
    public class GenerateAddendumCommandHandlerTests :
TestCommandBase
    {
        [Fact]
        public async Task
GenerateAddendumCommandHandlerTests_Success()
        {
            //Arrange
            var handler = new
GenerateAddendumCommandHandler(unitOfWork);

            //Act
            foreach (var employeeLoad in
unitOfWork.EmployeeWorkLoad.GetByPeriodId(1))
            {
                await handler.GenerateAddendum(
                    new GenerateAddendumCommand
                    {
                        PeriodId = 1,
                        EmployeeId = employeeLoad.EmployeeId,
                        WorkLoad = employeeLoad.WorkLoadHours
                    });
            }

            //Assert
            var res = unitOfWork.File.GetAllByPeriodId(1);
            Assert.Equal(2, res.Count());
            Assert.NotNull(res.FirstOrDefault().Data);
        }
    }
}
```

Тестирование модуля управления персоналом

Класс тестирования увольнения сотрудников:

```
using HRM.Application.Dismissing;
using HRM.Tests.Common;
```



```

using System.Threading.Tasks;
using Xunit;

namespace HRM.Tests.Commands.HRManagement
{
    public class DismissingCommandHandlerTests : TestCommandBase
    {
        [Fact]
        public async Task DismissingCommandHandlerTests_Success()
        {
            //Arrange
            var handler = new DismissingCommandHandler(unitOfWork);
            var employeeId = 1;
            var reason = "else";
            var Payments = 0;

            //Act
            await handler.Dismiss(
                new DismissingCommand
                {
                    EmployeeId = employeeId,
                    Reason = reason,
                    Payments = Payments
                });

            //Assert
            Assert.NotNull(unitOfWork.Dismissal.FirstAsync());
            Assert.Null(await unitOfWork.Employee.GetByIdAsync(1));
        }
    }
}

```

Класс                      тестирования                      найма                      кандидатов

InterviewingCommandHandlerTests:

```

using HRM.Application.Interviewing;
using HRM.Tests.Common;
using Microsoft.EntityFrameworkCore;
using System;
using System.Threading.Tasks;
using Xunit;

namespace HRM.Tests.Commands.HRManagement
{
    public class InterviewingCommandHandlerTests : TestCommandBase
    {
        [Fact]

```

```

        public async Task
InterviewingCommandHandlerTests_Success()
        {
            //Arrange
            var handler = new
InterviewingCommandHandler(unitOfWork);
            var id = 3;
            var isPassed = true;
            var passDate = DateTime.Now;
            var departmentId = 1;

            //Act
            await handler.TakeInterview(
new InterviewingCommand
            {
                CandidateId = id,
                IsPassed = isPassed,
                PassDate = passDate,
                DepartmentId = departmentId
            });

            //Assert
            var res = await
Context.Interviews.SingleOrDefaultAsync(interview =>
                interview.CandidateId == id &&
                interview.IsPassed == isPassed &&
                interview.Date == passDate);
            Assert.NotNull(res);
            Assert.NotNull(
                await
Context.Employees.SingleOrDefaultAsync(employee =>
                employee.InterviewId == res.InterviewId));
        }
    }
}

```

Класс      тестирования      расчета      заработной      платы

MonthResultSalaryHandlerTests:

```

using HRM.Application.Salary;
using HRM.Tests.Common;
using System.Threading.Tasks;
using Xunit;

namespace HRM.Tests.Commands.HRManagement
{
    public class MonthResultSalaryHandlerTests : TestCommandBase
    {
        /// <summary>

```

```

        /// employee 1 отработал 100 часов из 120 в отделе со
        ставкой 300р/час
        /// employee 2 отработал 140 часов из 120 в отделе со
        ставкой 300р/час
        /// </summary>
        /// <returns>employee 2 получит зарплату только за 120
        часов</returns>
        [Fact]
        public async Task MonthResultSalaryHandlerTests_Success()
        {
            //Arrange
            var handler = new
MonthResultSalaryHandler(unitOfWork);

            //Act
            await handler.Execute(null);

            //Assert
            Assert.Equal(30000,
unitOfWork.EmployeeWorkLoad.GetByIdAsync(1).Result.ResultSalary);
            Assert.Equal(36000,
unitOfWork.EmployeeWorkLoad.GetByIdAsync(2).Result.ResultSalary);
        }
    }
}

```