Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования   
«Магнитогорский государственный технический университет   
им Г.И. Носова»  
Многопрофильный колледж  
Отделение №2 «Информационные технологии и транспорт»

**КУРСОВОЙ ПРОЕКТ**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

по ПМ.02. Осуществление интеграции программных модулей

МДК.02.02. Инструментальные средства разработки программного продукта

на тему: Разработка приложения по учету и организации рабочего времени сотрудников предприятия на основе Visual Studio

Исполнитель: Дранков Н.А. студент 4 курса, группаИСпП-20-1

Руководитель: Тутарова В.Д, преподаватель МпК

Работа допущена к защите «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа защищена «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г. с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Магнитогорск, 2024

Министерство образования и науки Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Магнитогорский государственный технический университет   
им. Г. И. Носова»  
Многопрофильный колледж

**КУРСОВОЙ ПРОЕКТ**

Тема: Разработка приложения по учету и организации рабочего времени сотрудников предприятия на основе Visual Studio

Студент: Дранков Н.А.

Задание

Исходные данные по заданной предметной области:

Интернет-ресурсы, литературные источники, материалы образовательного портала

1. Выполнить теоретико-информационный анализ предметной области:
   1. Анализ предметной области
   2. Анализ и оценка существующих программных продуктов
   3. Постановка задачи
2. Спроектировать приложение
   1. Моделирование программного обеспечения
   2. Обоснование выбора средств разработки
   3. Разработка базы данных
   4. Проектирование меню и интерфейса программного обеспечения
   5. Спецификация программы
   6. Описание запросов при разработке программного обеспечения
   7. Администрирование программного обеспечения
3. Описать работу приложения
   1. Руководство пользователя
   2. Руководство программисту

Срок сдачи: « » 20 г.

Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / В.Д. Тутарова

« » 2023 г.

Задание получил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Н.А. Дранков

« » 2023г.

Магнитогорск, 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc160804434)

[1. ТЕОРЕТИКО-ИНФОРМАЦИОННЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc160804435)

[1.1. Анализ предметной области 5](#_Toc160804436)

[1.2. Анализ и оценка существующих программных продуктов 6](#_Toc160804437)

[1.3. Постановка задачи 7](#_Toc160804438)

[2. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 9](#_Toc160804439)

[2.1. Моделирование программного обеспечения 9](#_Toc160804440)

[2.2. Обоснование выбора средств разработки 12](#_Toc160804441)

[2.3. Разработка базы данных 13](#_Toc160804442)

[2.4. Проектирование меню и интерфейса программного обеспечения 16](#_Toc160804443)

[2.5. Спецификация программы 18](#_Toc160804444)

[2.6. Описание запросов при разработке программного обеспечения 19](#_Toc160804445)

[2.7. Администрирование программного обеспечения 23](#_Toc160804446)

[3. ОПИСАНИЕ РАБОТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 24](#_Toc160804447)

[3.1. Руководство пользователя 24](#_Toc160804448)

[3.2. Руководство программиста 28](#_Toc160804449)

[3.2.1. Назначение и условия применения программы 28](#_Toc160804450)

[3.2.2. Характеристика программы 28](#_Toc160804451)

[3.2.3. Обращение к программе 29](#_Toc160804452)

[3.2.4. Организация входной информации 29](#_Toc160804453)

[3.2.5. Организация выходной информации 29](#_Toc160804454)

[3.2.6. Сообщения 29](#_Toc160804455)

[ЗАКЛЮЧЕНИЕ 30](#_Toc160804456)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc160804457)

[ПРИЛОЖЕНИЕ А 33](#_Toc160804458)

[Листинг программы 33](#_Toc160804459)

# ВВЕДЕНИЕ

В современном мире эффективное управление задачами является ключевым аспектом успеха в различных областях деятельности. Разработка удобного и функционального приложения для этих целей является важным шагом к оптимизации рабочего процесса.

Цель настоящей работы заключается в исследовании и практической реализации возможностей Windows Presentation Foundation (WPF) и языка разметки XAML для создания упомянутого приложения. WPF обеспечивает возможность разработки современных пользовательских интерфейсов, а XAML является языком разметки, определяющим структуру и внешний вид элементов пользовательского интерфейса.

Основное внимание будет уделено созданию интуитивно понятного интерфейса, функциональности для создания, отслеживания и управления задачами, а также на других важных аспектах управления задачами. Планируется изучить принципы разработки WPF-приложений и лучшие практики в области управления задачами.

Поскольку основной упор в работе делается на использование передовых технологий для повышения производительности и эффективности управления задачами, данная работа направлена на демонстрацию потенциала современных инструментов в решении повседневных задач управления задачами.

Для реализации цели необходимо решить следующие задачи:

1. Выполнить теоретико-информационный анализ предметной области:
   1. Анализ предметной области;
   2. Анализ и оценка существующих программных продуктов
   3. Постановка задачи
2. Спроектировать приложение
   1. Моделирование программного обеспечения
   2. Обоснование выбора средств разработки
   3. Разработка базы данных
   4. Проектирование меню и интерфейса программного обеспечения
   5. Спецификация программы
   6. Описание запросов при разработке программного обеспечения
   7. Администрирование программного обеспечения
3. Описать работу приложения
   1. Руководство пользователя
   2. Руководство программисту.

# Теоретико-информационный анализ предметной области

## Анализ предметной области

Для успешной разработки WPF-приложения, способного эффективно управлять задачами на XAML, необходимо провести анализ предметной области. Важно понять основные принципы управления задачами, потребности пользователей в таком приложении, а также изучить существующие решения на рынке.

Как появился тайм-менеджмент: как явление («стремление повысить продуктивность») он существовал на протяжении всей человеческой истории. Его использовали и в античные времена, и в средневековье, и в эпоху пара. Вполне возможно, что и первобытные люди уже вовсю размышляли о том, как быстрее изготовить каменный топор или эффективнее спрятаться от саблезубого тигра.

Однако само слово «тайм-менеджмент» появилось лишь в середине XX века. В 1959 году американский бизнес-консультант Джеймс Т. Маккей написал книгу, которая так и называлась — The Management of Time. Книга стала бестселлером в США, а термин быстро прижился в деловой литературе.

Впрочем, сегодня многие считают этот термин крайне неудачным

Тайм-менеджмент это уже давно не только про управление временем. В некоторых современных системах планирования (например, в GTD) время почти не упоминается.

Для чего нужен тайм-менеджмент: Тайм-менеджер – это организация времени путём тщательного планирования рабочей нагрузки, чтобы сотрудники могли без труда соблюдать сроки работ. Грамотное управление временем помогает:

Повышать эффективность и производительность: Плохое управление временем приводит к тому, что сотрудники чувствуют себя перегруженными. По результатам [исследования](https://hhcdn.ru/file/17403093.pdf) популярного рекрутингового портала почти половина сотрудников отечественных компаний призналась, что за последний год стали больше уставать на работе. Конечно, причина не только в объёмах задач, но и в информационной повестке. Тем не менее всё вкупе негативно сказывается на нагрузке работников. Тех, кто обладает навыками тайм-менеджмента, справляются с нагрузками легче, поэтому у них повышается эффективность, что, в свою очередь, приводит к повышению производительности. На самом деле, тайм-менеджмент давно уже вышел за рамки «управления временем». Сегодня он охватывает почти все темы, связанные с личной эффективностью. Он включает в себя и методы психологической само регуляции, и стратегии решения проблем, и даже такие философские вопросы, как поиск смысла жизни.;

Сдавать работы в срок. Это наиболее очевидное преимущество эффективного тайм-менеджмента, но также и одно из самых важных, если не самое важное. Тайм-менеджмент помогает сотрудникам укладываться в сроки, что очень важно, когда речь идёт об удовлетворении ожиданий клиентов;

Испытывать меньше стресса и беспокойства. Стресс и тревога заставляют сотрудников не только срывать сроки и выполнять работу некачественно, но и влияют на их здоровье. Работодатели для получения положительного эффекта обязаны следить за тем, чтобы психическое здоровье подчинённых было в норме. Сотрудники, которые изо дня в день испытывают стресс на своём рабочем месте, с большей вероятностью возьмут больничный и найдут другую работу. Тайм-менеджмент позволяет сгладить острые углы, настроить работу на спокойный лад, без авралов и потрясений, что благоприятно сказывается на эмоциональном фоне в коллективе;

Обеспечить более высокое качество работы. При хорошем тайм-менеджменте у сотрудников есть время, необходимое для выполнения работы не только вовремя, но и более высокого качества. Если выполнять задачи с чувством, с толком, с расстановкой, а не в спешке, возникает гораздо меньше ошибок, и появляется возможность найти наиболее подходящие пути и инструменты для решения рабочих вопросов;

Повышение уверенности в себе. Когда сотрудники справляются со своей рабочей нагрузкой, это придаёт им уверенности и позволяет доверять своим способностям. Как следствие, стресс и беспокойство уменьшаются по физиологическим причинам: организм вырабатывает дофамин, буквально окрыляя персонал;

Сокращение прокрастинации и времени простоя. Когда сотрудники знают, как расставлять приоритеты, это уменьшает прокрастинацию. Например, приём «съесть лягушку» сокращает время простоя и помогает выполнять задачи;

Улучшить баланс между работой и личной жизнью. 52% работающих граждан России не могут соблюдать баланс между работой и личной жизнью. Перекос у большинства из опрошенных в сторону карьеры. Однако хороший work-life balance приносит пользу не только самим сотрудникам, но и компании, где они работают. Когда персонал чувствует себя отдохнувшим и имеет возможность перезарядиться, он максимально эффективен в работе;

* принимать лучшие решения. Когда у сотрудников есть время, чтобы сосредоточиться и работать тщательно, это избавляет их от необходимости принимать решения под давлением сроков, ответственности и так далее. Вместо скоропалительных решений, используя грамотный подход к управлению временем, можно принимать обдуманные, обоснованные решения, которые принесут пользу коллективу и бизнесу. Зачем нужно управлять временем В современном мире, где каждая минута на счету, умение хорошо управлять своим временем — это как суперсила. Она открывает двери в мир возможностей в следующих областях:
* достижение целей: Управление временем ускоряет процесс достижения целей, помогая четко определить их и спланировать необходимые шаги и этапы. Это позволяет двигаться к целям более стремительно и эффективно, сохраняя при этом высокий уровень мотивации.
* рост продуктивности: Эффективное планирование и организация рабочего времени помогают избежать прокрастинации, сосредоточив внимание на приоритетных задачах. Такой подход позволяет выполнять больше задач за конкретный промежуток времени, улучшая общую производительность.
* снижение стресса: Четкий план и контроль над временем уменьшают уровень стресса, добавляя уверенности в свои действия. Это помогает избежать ситуаций, когда дела выполняются в спешке и в последний момент.
* баланс работы с личной жизнью: Тайм-менеджмент помогает достичь гармонии между профессиональной деятельностью и личной жизнью, обеспечивая достаточно времени для семьи, друзей и хобби.

Понимание и применение принципов тайм-менеджмента являются ключевыми для достижения успеха в профессиональной сфере и улучшения качества личной жизни. В следующих разделах мы разберем практические стратегии и инструменты, которые помогут вам усовершенствовать свои навыки управления временем.

Инструменты тайм-менеджмента: чтобы помочь сотрудникам достичь успеха в управлении временем и получить все преимущества тайм-менеджмента, существует целый ряд инструментов. Начать можно с банального списка дел – на бумаге или в электронном виде. Когда все задачи, обязательные к выполнению, перед глазами, легче расставить приоритеты и планировать свою занятость. Почти невозможно навести порядок без использования (онлайн) календаря. Календари помогут ориентироваться во времени и напомнят о предстоящих задачах и дедлайнах. Это совершенно очевидно, но что ещё более важно – так это полноценное использование календаря. Большинство людей используют календарь только для записи своих встреч. Но на самом деле это инструмент, который должен помочь лучше контролировать время. Календарь помогает видеть «с высоты птичьего полета» ближайшие несколько недель. Системы учета рабочего времени покажут, как расходуются трудовые часы на самом деле.

* Принципы планирования рабочего времени: Планированием называют распределение времени для решения краткосрочных и долгосрочных задач. Практически все существующие методы управления временем состоят из трех компонентов: приоритизации, планирования и структурирования.
* Приоритизация. Чтобы выполнить задачу, нужно определить, насколько она срочная, сложная и важная, и только потом приступить к ее выполнению.
* Планирование. Чтобы выполнить задачу, нужно разобраться, когда ее следует сделать и сколько времени на это уйдет.
* Структурирование. Чтобы выполнить задачу, нужно понять, как отслеживать ее выполнение и результаты. Большая часть техник тайм-менеджмента опирается на структурирование и приоритизацию, и только малая часть — сложная комбинация всех трех принципов.

Использование этого инструмента позволяет оптимизировать задачи трудового дня, оптимизировать работу над проектами. При планировании выбирают приоритетные цели, ограничивают время их выполнения, структурируют деятельность команды. Задачи рабочего дня распределяют с учетом целей компании.

* Благодаря грамотно составленному плану можно:
* расставлять приоритеты на день;
* определять, какие задачи можно делегировать;
* оптимизировать работу над проектами;
* выбирать лучшие решения;
* распределять нагрузку между сотрудниками;
* планировать оставшееся в расписании свободное время;
* повышать общую продуктивность компании.

Принципы управления задачами: исследование методик и подходов к управлению задачами поможет определить основные функциональные требования к приложению. Изучение таких концепций, как установка целей, планирование задач, отслеживание прогресса и оценка результатов, поможет определить необходимый функционал.

Потребности пользователей: Важно выявить основные потребности и ожидания потенциальных пользователей приложения. Это позволит создать интерфейс, который будет интуитивно понятен и удобен для использования, а также учесть особенности процесса управления задачами в конкретной области.

Анализ существующих решений: изучение аналогичных приложений на рынке позволит выявить их преимущества и недостатки, что поможет определить уникальные особенности и конкурентные преимущества разрабатываемого WPF-приложения

Одна из основных функций работодателя: учет использования рабочего времени сотрудников на предприятии. От количества отработанных персоналом часов напрямую зависят расходы на заработную плату. Кроме того, контроль дает возможность оценить производительность компании. Не стоит забывать и о том, что эти действия необходимы по действующему законодательству.

В наши дни выполнять эту процедуру стало намного легче благодаря появлению специализированного программного обеспечения. В этой статье будут рассмотрены разные способы и правила проведения мониторинга рабочего процесса, а также перечислены наиболее эффективные решения.

Что такое рабочее время по ТК РФ: Это понятие регламентируется Трудовым кодексом Российской Федерации. Под ним подразумеваются часы, в которые сотрудник исполняет свои непосредственные обязанности. При этом во внимание принимаются положения составленного при приеме в штат договора, а также внутренние правила фирмы. Сверхурочная занятость тоже подпадает под это определение и должна быть дополнительно оплачена. Нормативные акты регулируют и продолжительность работы. По современным правилам она равняется 40 часам.

Понятие режима: график, который применяется для упорядочивания и организации учета рабочего времени персонала на предприятии. Он предусматривает сразу ряд аспектов. Среди них:

* Длительность трудовой недели. Допускается несколько вариантов на усмотрение руководства. Первый – 5 будних дней и 2 выходных. Кроме того, нередко применяется модель 6/1. Можно также прибегнуть к использованию гибкого расписания.
* Протяженность смены (в том числе, для неполной занятости).
* Момент начала и окончания исполнения подчиненным своих непосредственных обязанностей.
* Продолжительность перерывов, их график.
* Число смен, в которое осуществляется производство.
* Группы сотрудников, для которых предусмотрен ненормированный режим.

Понятие учета рабочего времени сложное. Эти пункты прописываются во внутреннем распорядке компании. Если существуют исключения для некоторых специалистов, то они обязаны быть зафиксированы в контракте. Кроме того, значения указанных выше параметров не должны идти вразрез с Трудовым кодексом Российской Федерации. Требования отражены в статье 110. К примеру, там прописано, что человек должен отдыхать в течение минимум 42 часов подряд, вне зависимости от расписания. Не распространяется данное предписание на вахтовый метод. Однако и в этом случае обязателен хотя бы один выходной.

Цели и задачи организации рабочего времени: Правильная (рациональная) организация работы на предприятии: рабочего времени, рабочего места, графика работы, отдыха, перекуров, выходных и т.п. – это очень важная часть общего тайм-менеджмента любой организации (от англ. Time Management, «управление временем»). К чему может привести рационально спланированная политика управления:

* Растёт продуктивность каждого отдельного человека/сотрудника (благодаря распределению времени на вверенные обязанности).
* Снижается общий уровень затрат, оптимизируются расходы на заработную плату.
* Нормализуются трудовые отношения, улучшаются взаимоотношения между отдельными членами коллектива (фактически руководитель может регулировать состав смен и их пересечение между собой, ротацию кадров между разными сменами, взаимозаменяемость сотрудников, наставничество и другие детали).
* У управленца появляется возможность более детального и точного планирования деятельности своей службы/отдела.
* Труд строго нормируется (сотрудники могут планировать свои личные дела с привязкой к выходным и к своему свободному времени).
* Упрощается расчёт заработной платы (к тому же, всегда можно сверить график явок и отработанные часы с тем, что было учтено в официальном табеле при расчётах).

Список задач следует регулярно анализировать. Это нужно делать, чтобы преобразовывать цели в план действий. Например, владельцу бизнеса нужно увеличить прибыль. Для этого он должен самостоятельно или с помощью команды оценить ситуацию на рынке, в компании и решить, какие шаги следует предпринять: повысить качество реализации проектов, привлечь больше клиентов, создать и предлагать новые услуги.

Каковы преимущества тайм-менеджмента на работе? Если проанализировать ваш рабочий день, то можно прийти к выводу, что непредвиденные ситуации могут возникать часто. Допустим, у вас был четкий график на день, но ваши планы были нарушены тремя встречами подряд. Хотя встречи могут быть для вас не полезными, а иногда они не важны и тратят ваше драгоценное время.

Многие авторитетные исследования показывают, что собрания — одна из самых распространенных причин, по которой работники не выполняют свою работу в срок. Проводилось исследование hh.ru в 2018 — 2020 годы, в ходе которого принял участие 2001 сотрудник из российских компаний.

Согласно результатам этого исследования, сотрудники утверждают, что их самыми большими трудностями при выполнении работы являются:

Бесполезные встречи – 57%.

Лишние письма – 53%.

Отвлечение коллегами – 15%.

Преимущества тайм-менеджмента для менеджеров: если вы руководитель группы или менеджер, вы обязаны знать больше о эффективном управлении временем, чтобы помочь своим коллегам оптимально организовать свой рабочий день. Рассмотрим основные преимущества тайм-менеджмента на работе, которые имеют важное значение для менеджеров или руководителей групп.

* Выполнение заданий в срок: в процессе постановки задач для своих подчиненных всегда устанавливайте срок сдачи работ. Если сотрудник понимает, что данную задачу нужно выполнить точно в срок, мозг человека подсознательно стремиться выполнить работы в установленный срок. Это позволит вашим сотрудникам выполнять свою работу быстрее.
* Повышение качества работы: Крайне важно, чтобы вы помогали членам вашей команды организовывать свой рабочий день и расставить приоритеты. Когда у ваших сотрудников есть четкие приоритеты, они могут сосредоточить свое внимание на выполнении важных задач. Сосредоточение внимания на одной задаче позволяет обеспечить высокое качество работы.
* Сокращение задержек и стресса: У всех нас бывают такие дни, когда мы хотим отложить в “долгий ящик” выполнение ряда задач. Но с правильными навыками тайм-менеджмента ваши коллеги будут готовы справятся с нагрузкой и приступят к выполнению заданий. Это позволит справиться с пагубной привычкой откладывать все на потом. Научно доказано, чем меньше прокрастинация на рабочем месте, тем меньше стресса. Это позволит вашей команде равномерно распределить рабочее время и не столкнуться с проблемой, когда скопления большого объема работы, что может вызвать стресс

Преимущества тайм-менеджмента для сотрудников: получение навыков тайм-менеджмента важно и для самих сотрудников. Давайте рассмотрим основные преимущества тайм-менеджмента для сотрудников.

* Повышение производительности: когда работники знают, как выставлены приоритеты, они будут стремиться в первую очередь решать самые важные задачи, не отвлекаясь на второстепенные. Это позволит повысить их профессиональный уровень и производительность.
* Карьерный рост: своевременное выполнение поручений вызывает доверие к сотруднику. Если, отработав в компании несколько лет, сотрудник приобретает репутацию надежного опытного специалиста, то у него появляется больше шансов продвинуться по карьерной лестнице или добиться повышения заработной платы.
* Оптимальный баланс между работой и личной жизнью: вспомните то чувство, когда вы заканчиваете все важные задачи на один день раньше и наслаждаетесь свободным временем. При правильном управлении временем можно достичь баланса между работой и личной жизнью. Мы узнали об основных преимуществах управления временем на работе. Перейдем к действенным методам, которые позволяют улучшить навыки управления временем у ваших сотрудников.

Отвлекающие факторы сводят к нехватке рабочего времени для решения основных задач в срок. К отвлекающим факторам, которые мешают выполнять поставленные задачи можно отнести: громкие разговоры коллег, отвлекающие вопросы, чтение новостей, общение в социальных сетях. Главная цель заключается в том, чтобы выявить ключевые проблемы, а затем соответствующим образом адаптировать свой распорядок дня и свести к минимуму влияние отвлекающих факторов.

Планирование времени сотрудников: сотрудники эффективно выполняют обязанности при составленном плане работы на день, неделю, квартал. Основным задачам отводят до 60% времени, до 20% оставляют на непредвиденные дела и еще 20% — на ситуативные задачи. Планирование рабочего времени и контроль за выполнением задач должны быть регулярными и системными. Результаты сотрудников отслеживают их руководители.

Чтобы повысить производительность сотрудников, необходимо:

* проанализировать эффективность их деятельности; сравнить работу на предприятии и в других организациях;
* внедрить систему тайм-менеджмента в отделах;
* следить за ее применением;
* оценивать, насколько эффективно система позволяет структурировать и оптимизировать рабочее время;

Стандартизация работы позволяет повысить продуктивность специалистов, рационально использовать трудовое время, соблюдать сроки реализации проектов, снижать риски просрочек, репутационных и финансовых потерь компании.

Ошибки планирования: распространенные ошибки при планировании:

* Неправильная расстановка приоритетов. Производительность компании повышает выполнение важных дел. Дела на перспективу, анализ выполняют после решения срочных вопросов.
* Чрезмерное внимание к небольшим задачам. Согласно закону Парето, 20% усилий помогают достигать 80% результата при грамотном определении целей.
* Отсутствует время на отдых. Высокая концентрация дел в течение рабочего дня снижает производительность. Если нет границ между этапами проектов, видами деятельности, у сотрудников и руководителя постепенно снижается концентрация, развивается эмоциональное выгорание.
* Отсутствие контроля за временем решения задач. Прокрастинация, просмотр писем, длинные телефонные переговоры снижают производительность. Необходимо отслеживать эффективность трудового дня, анализировать результаты, оптимизировать задачи.
* Постоянная работа в режиме многозадачности. Регулярное выполнение нескольких проектов, совмещение трудовых обязанностей по разным направлениям снижает способность к концентрации внимания. Продуктивность сотрудников падает. Для сложной умственной деятельности необходима поэтапность. Продуктивно работать над задачей в течение одного временного цикла с перерывами.
* Планировать время работы полезно как владельцу бизнеса, так и его сотрудникам. Тайм-менеджмент дает возможность оптимизировать время, вовремя выполнять поставленные задачи и допускать при этом минимум ошибок.

Преимущества обучения тайм-менеджменту сотрудников: рассмотрим основные преимущества по организации тренингов тайм-менеджмента для ваших сотрудников. Это обучение может быть весьма полезным и для менеджеров.

5 индикаторов слабого тайм-менеджмента:

* Один или несколько пунктов из списка задач постоянно откладываются. Допустим, руководитель отдела маркетинга хочет сам подписывать контракты с партнёрами из Дубая. Для этого ему нужно разобраться в основах иностранных договоров.

Получилась большая задача: найти юриста, согласовать структуру обучения и выделить для него время. Поскольку задача несрочная, она постоянно переносится на завтра. А запись в списке задач отвлекает и заставляет руководителя нервничать.

* После рабочего дня остается состояние неудовлетворённости. По данным Statista, 44% опрошенных офисных сотрудников не удовлетворены тем, как прошёл рабочий день. За 10 лет исследований уровень недовольных вырос с 31% до 43%. Например, руководитель отдела разработки записал задачу: за три дня проверить вёрстку пяти сайтов. Чтобы уложиться в этот срок, руководитель решил тратить по 45 минут на каждый сайт. Первый сайт он проверил за полтора часа, потому что в нём была не одна страница, а три. Второй сайт ― за час, потому что нужно было ответить на письма. Остальные три до конца рабочего дня проверить не успел. Теперь придётся вместо отдыха проверять оставшиеся сайты и переживать, что не успевает в срок.
* Внимание и время тратится на второстепенные задачи. К примеру, на встрече клиент отказался от проекта, потому что на него нет бюджета. Но менеджеру кажется, что настоящая причина в качестве работы. Он придумывает, как улучшить проект, и ставит задачи команде. Это отнимает время — люди работают над несуществующим проектом в ущерб приоритетным.
* Жалобы от клиентов и смежных руководителей. Из-за загруженности руководителя сроки по проекту могут сдвинуться. Например, продажи антивирусов в компании зависят от нескольких отделов: разработчиков, тестировщиков, аналитиков. Если тестировщики поздно сообщат о баге, разработка затянет обновление антивируса. Пользователи останутся без защиты на то время, пока команда заканчивает работу. На руководителя посыплются жалобы от смежных управленцев, будут жалобы и от клиентов — но уже на компанию. А всё из-за того, что он неправильно спланировал время, не выделил и не контролировал промежуточные результаты работы команды.
* Команда устаёт, выполняя обычный набор задач. Отдел разрабатывает интернет-магазины. Команда работает по плану: собрать информацию из брифа, подготовить текст, макет, дизайн и вёрстку. Пока не пришёл новый руководитель, все задачи выполнялись в срок. Но теперь дизайнер вместо одного макета делает пять, а верстальщик в это время сидит без дела. Набор задач остался прежним, но из-за того, что задачи усложнились, команда начала срывать дедлайны.

Проведение анализа предметной области является важным этапом перед началом разработки, поскольку позволяет определить требования к приложению, учесть потребности пользователей и избежать повторения ошибок, сделанных в существующих решениях.

## Анализ и оценка существующих программных продуктов

При разработке WPF-приложения для управления задачами на XAML, ориентированного на корпоративное использование, важным является учет того, что организации в значительной степени используют подобные решения для оптимизации управления временем и задачами. Однако, в силу конфиденциальности информации и уникальных потребностей компаний, подобные программные продукты редко доступны в открытом доступе для оценки и использования. И, хотя существуют онлайн-сервисы для управления задачами, они часто не удовлетворяют специфическим требованиям и управленческим процессам компаний.

Рассмотрим на примере приложения «Учет рабочего времени» версия 3.5

Программа предназначена для учета рабочего времени сотрудников для начисления заработной платы в бухгалтерии организации.

Специалисты по начислению зарплаты, специалисты по труду требуют от руководителей структурных подразделений табели учета рабочего времени, графики дежурств сменного персонала. Руководители в свою очередь требуют эти данные от своих подчиненных (мастеров участков, цехов, бригадиров и т.п.) (Рисунок 1).



Рисунок 1 – Главное меню приложения

Справочник «Подразделения». Несложный справочник, всего два поля данных «подразделение» и «ПО». (ПО вышестоящее структурное подразделение организации, к которому относятся подразделения). Ввод одинаковых записей блокируется программой. Введите в справочник те подразделения, с которыми будете работать (формировать табель или графики) (Рисунок 2).

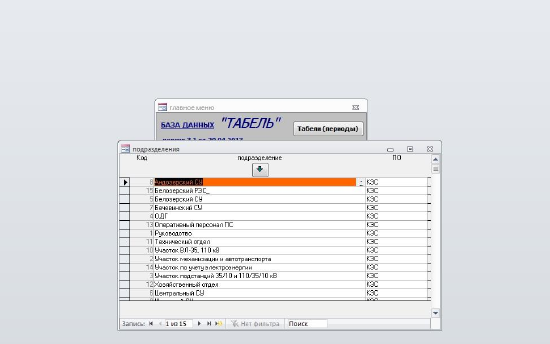


Рисунок 2 – Окно подразделения

В него вводятся все наименования должностей (профессий) сотрудников организации. Ввод одинаковых записей блокируется программой (Рисунок 3).

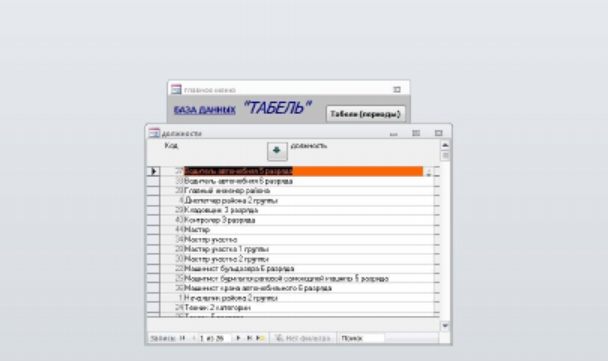


Рисунок 3 – Окно должности

Здесь все просто. Нужно заполнить данные о вашей организации. Данные подставляются в печатные формы Табель Т-13 и графики дежурств. (Рисунок 4)

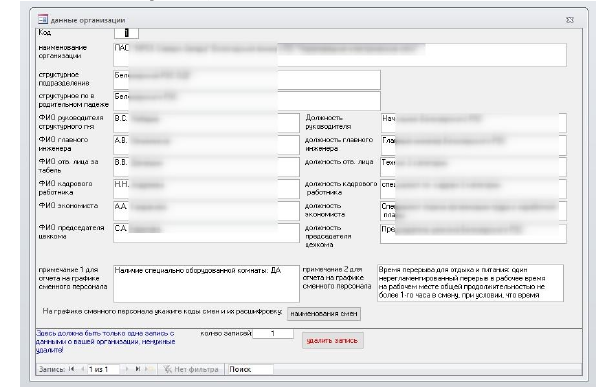


Рисунок 4 – Окно с данными организации

Сюда вводятся записи по кодам вида оплаты, которые также отображаются в печатной форме табеля, влияющие на начисление зарплаты. В этом справочнике уже введены наиболее часто используемые коды, если необходимо добавьте свои коды (Рисунок 5).

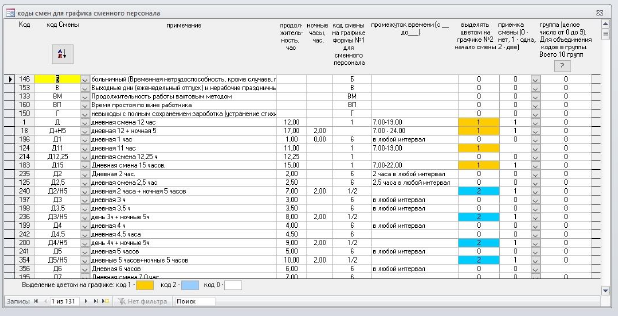


Рисунок 5 – Окно коды смен для графика сменного персонала.

Программа «Учет рабочего времени»

Первой страницей встречающей пользователя будет “Аутентификация системы ИТРИУМ”. В окне аутентификации необходимо ввести имя и пароль оператора (имя и пароль оператора назначает Администратор системы). Чтобы исключить возможность просмотра пароля посторонним лицом, при его вводе символы отображаются в виде звездочек, поэтому будьте внимательны: на экране вы не сможете проверить, правильно ли указан пароль. Нажатием кнопки Клавиатура в окне аутентификации раскрывается виртуальное изображение клавиатуры для ввода имени и пароля с помощью мыши. (Рисунок 6)

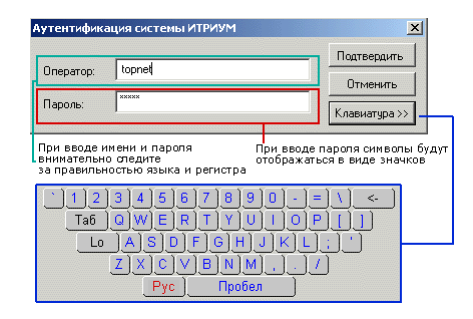


Рисунок 6 – Страница аутентификации пользователя

Основное окно программы состоит из меню, панели инструментов и окна отображения текущего режима. По умолчанию в основном окне программы загружается режим Отчетов (рисунок 7).

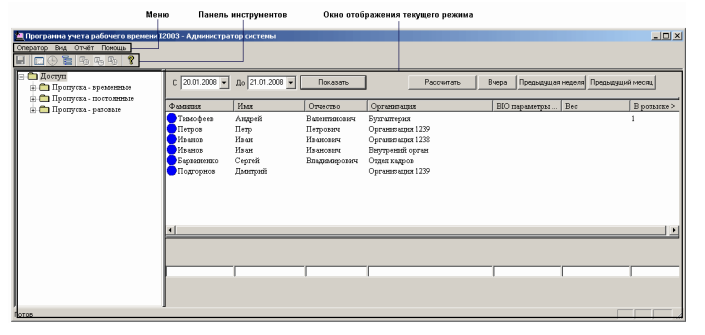


Рисунок 7 – Главное меню приложения

Контекстное меню элемента содержит функции вызова команд (Рисунок 8): Переименовать— переименовать выделенный элемент. Данная опция доступна также при нажатии на кнопку F2 на клавиатуре. Найти — найти элемент в дереве. Вызывается также при нажатии на кнопку F3 на клавиатуре. После вызова данной команды открывается окно поиска, в котором необходимо ввести имя или часть имени искомого элемента (без учета регистра) или идентификатор элемента. При поиске элемента по идентификатору, в данном окне отметьте флаг Поиск по ID. Нажмите на кнопку OK для поиска первого соответствия в дереве элементов. Для поиска последующих соответствий в открывшемся окне с текстом «Искать следующий?» нажмите на кнопку Да. В противном случае, нажмите на кнопку Нет. В случае, если соответствие не найдено, в программе появится окно с сообщением «Элемент системы с таким именем не найден». Отметьте флаг искать только в текущей папке и ее подпапках для поиска элемента только в выбранной папке и ее подпапках. В папках уровнями ниже поиск производиться не будет

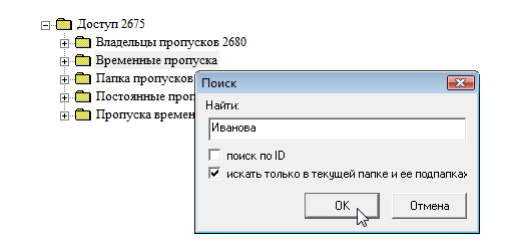


Рисунок 8 – Окно поиск

Заменить — после вызова данной команды на экране появится окно Замена (рисунок 4). Из названия для текущего и всех дочерних элементов удаляется слово или фраза, введенные в поле найти, и заменяется на слово или фразу, введенные в поле и заменить на. Отметьте флаг без учета регистра, чтобы не учитывать регистр букв. Данная опция также доступна при нажатии на кнопку F4 на клавиатуры

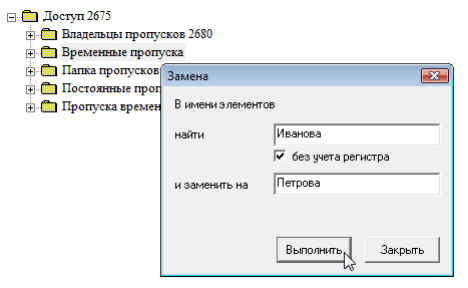


Рисунок 9 – Окно замена

В режиме Отчеты можно формировать отчет, состоящий из множества параметров, например, отработанное время, время нахождения на рабочем месте, опозданиях и т.д. В режиме Отчеты рабочая область основного окна состоит из дерева элементов, списка сотрудников, фильтра и полей временного интервала (Рисунок 10).

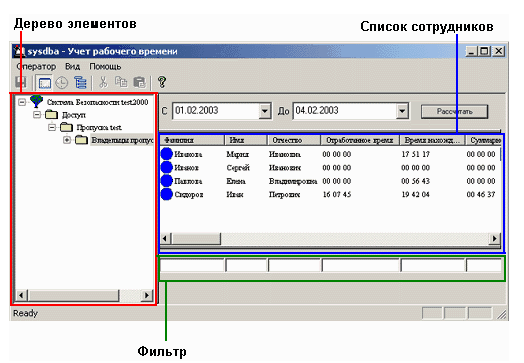


Рисунок 10 – Окно программы в режиме Отчеты

Завершающей на рассмотрение приложение будет “Учет рабочего времени” Версия 1.0 Для запуска редактора выберите в главном меню windows “Учет рабочего времени -> Редактор” (или воспользуйтесь ярлыком на рабочем столе). При первом запуске программы будет предложено ввести название организации. Главное окно программы представлено 2 вкладками – “Организация” и "Режимы рабочего времени"(Рисунок 11):

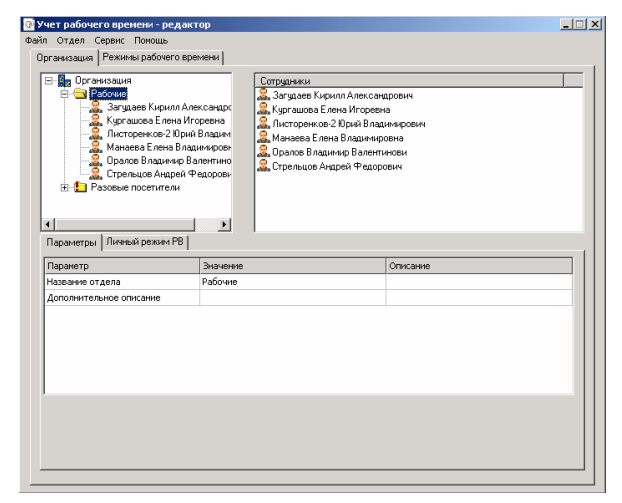


Рисунок 11 – Главное меню приложения “Редактор”

Далее описывается стандартная последовательность действий при работе с редактором. 1. Прежде всего, необходимо добавить сотрудников в организацию. Сотрудники могут быть добавлены только из базы данных проксимити-карт Doors32. Для добавления сотрудников: в меню «Сервис» выберите пункт «Импорт из СКД» – появится диалоговое окно “Импорт сотрудников” (Рисунок 12).

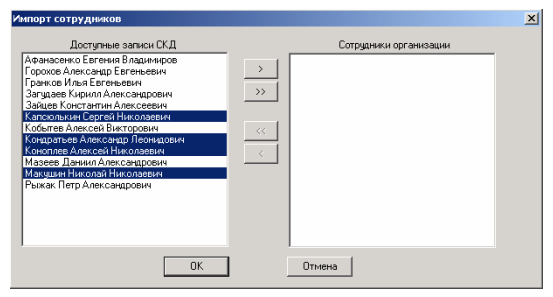


Рисунок 12 – Импорт сотрудников

2. Необходимо создать отделы, отражающие структуру организации. Для этого выберите в меню “Отдел” пункт “Создать” – откроется диалоговое окно “Создание нового отдела организации” (Рисунок 13)

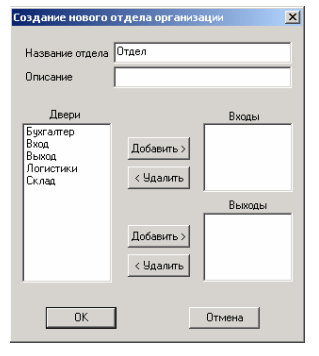


Рисунок 13 – Создание нового отдела организации

Если впоследствии потребуется изменить название / описание отдела или назначение входных / выходных дверей, воспользуйтесь меню “Отдел / Изменить” – появится диалог, аналогичный описанному выше (Рисунок 14).

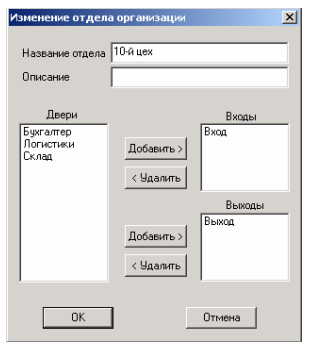


Рисунок 14 – Изменение отдела организации

3. Сотрудников организации следует распределить по отделам (в соответствии со структурой организации). Для этого: перейдите на вкладку “Организация”; · выберите (сделайте клик) название организации в дереве – в списке справа будут показаны все сотрудники; · выберите сотрудников в этом списке и с помощью мыши переместите (drag & drop) их в нужный отдел в дереве организации:

4. Создайте общие режимы рабочего времени. Для этого: · перейдите на вкладку “Режимы рабочего времени” (Рисунок 15)

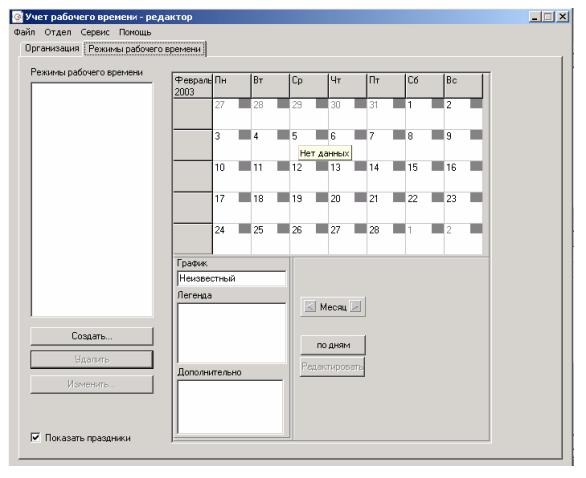


Рисунок 15 – Режимы рабочего времени

Можно сделать вывод что бы приведенные программы имеют удобный и понятный интерфейс богатый функционал, но единственный минус, который можно подметить, это возраст самих программ и начальные версии в которых вероятность попасться на ошибку очень даже велико.

## Постановка задачи

Приложения по учету и организации рабочего времени сотрудников предприятия на основе Visual Studio предназначена для эффективной организации рабочих процессов, управления задачами, а также координации работы команд. Ключевым элементом данной системы является канбан-доска, обеспечивающая гибкую систему структурирования задач и управления процессом выполнения.

Цели и задачи разработки данной системы:

* повышение производительности и эффективности команды разработки;
* создание комфортной среды для совместной работы над задачами;
* организация мониторинга процесса выполнения задач и контроля за их выполнением.

Функциональные требования системы включают:

* Создание неограниченного числа канбан-досок для различных задач;
* Группировку задач по колонкам, отражающим разные этапы выполнения (например, "Готово", "В процессе", "Просрочено");

Функционал создания и редактирования задач включает:

* Добавление новых задач с указанием названия, описания, сроков выполнения, и статуса;
* Редактирование и обновление информации о задачах.

Система также предоставляет возможности совместной работы и синхронизации, такие как:

* Доступ к доске и задачам для команды с различными уровнями доступа;
* Синхронизация между разными устройствами и пользователями.

Другие важные функции системы включают:

* Архивирование завершенных задач для структурированного хранения данных;
* Добавление задач для сотрудника;
* Поиск задач по дате для удобства работы.

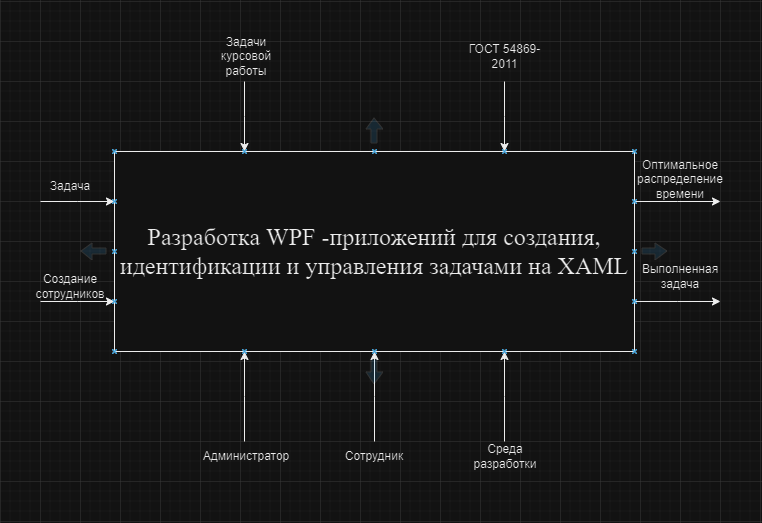
# ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## Моделирование программного обеспечения

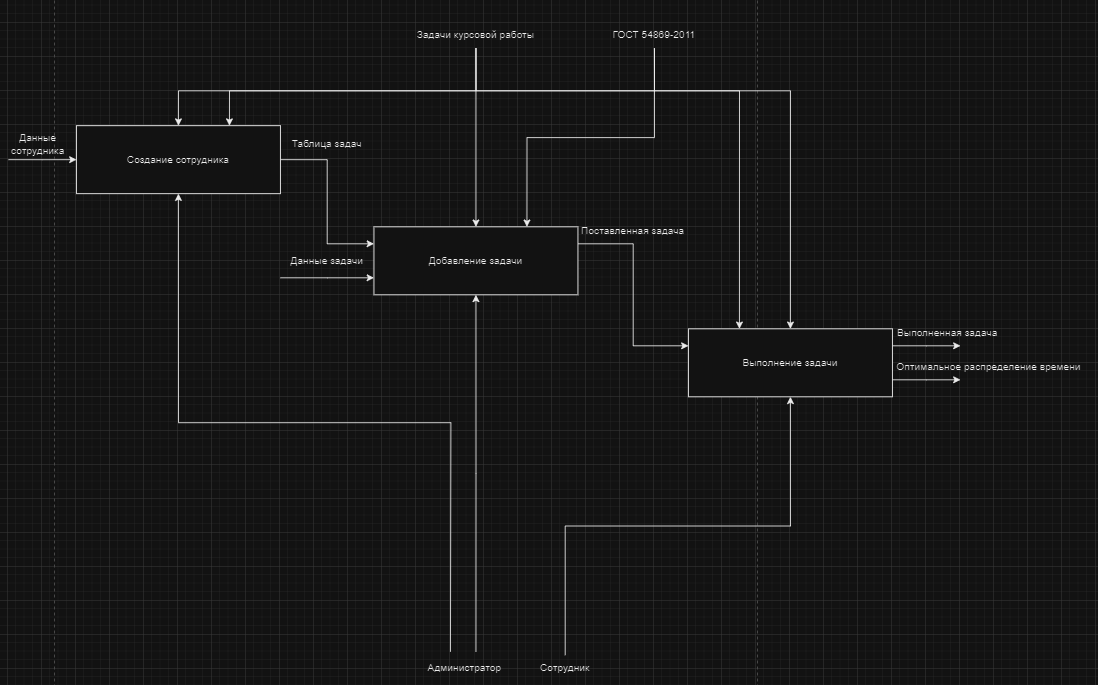
Моделирование программного обеспечения является важным этапом в разработке современных приложений. Разработка приложения по учету и организации рабочего времени сотрудников предприятия на основе Visual Studio представляет собой эффективный способ создания интуитивно понятного и функционального пользовательского интерфейса. Использование технологии XAML позволяет разработчикам легко описывать внешний вид приложения и его поведение, что способствует ускорению процесса разработки и повышению удобства использования программы конечными пользователями.

Разработка WPF-приложения включает в себя создание различных элементов пользовательского интерфейса, таких как окна, кнопки, меню, таблицы и другие. Использование XAML позволяет легко стилизовать и размещать эти элементы на экране, обеспечивая их взаимодействие с пользователем. Кроме того, для работы со сложными данными в приложении часто применяются шаблоны данных и привязка данных, что позволяет эффективно управлять информацией, отображаемой в приложении.

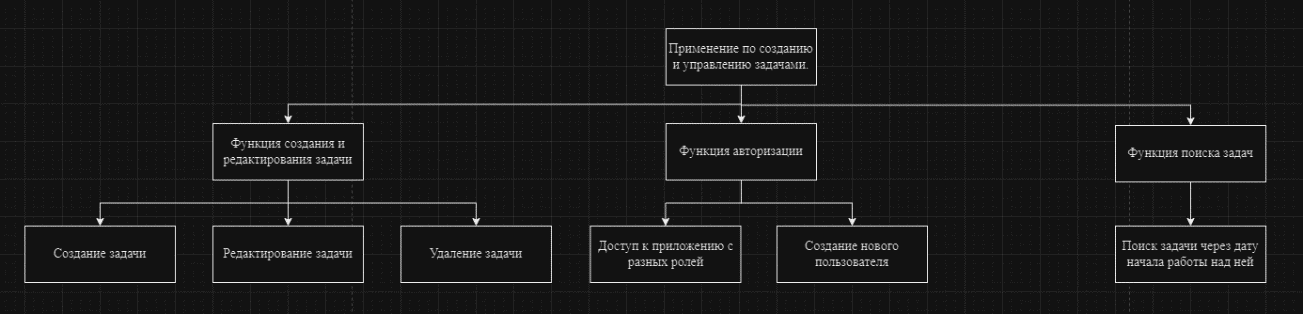
IDEF - это специальная методология моделирования процессов и систем, которая используется для анализа, проектирования и документирования информационных систем. Она включает в себя набор графических и текстовых обозначений, позволяющих стандартизировать процессы моделирования и повысить понимание системы как со стороны разработчиков, так и пользователей. IDEF0: Диаграммы функционального моделирования, используемые для описания функций, обмена данными и управления процессами в системе (Рисунок 16).

  
Рисунок 16 – Диаграмма IDEF0

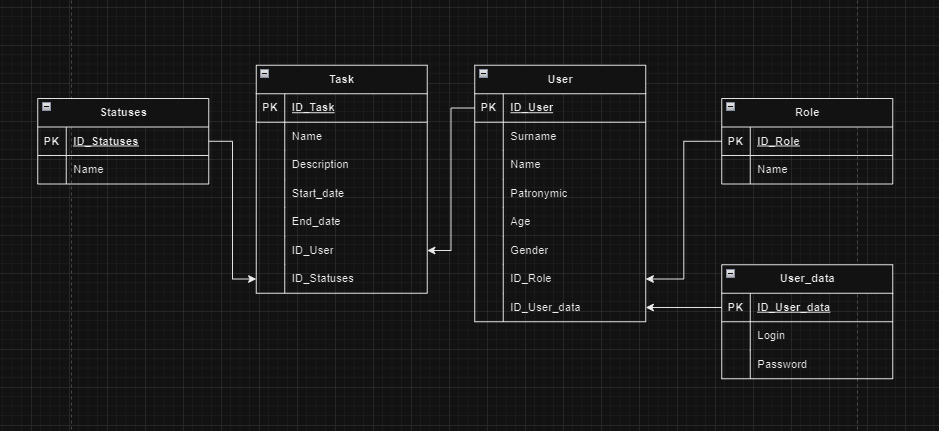
Декомпозиция в разработке программного обеспечения представляет собой процесс разбиения сложной системы на более простые и понятные компоненты. Цель декомпозиции - упростить процесс разработки, управления и поддержки программного продукта. Путем декомпозиции проект структурируется на более мелкие и управляемые части, что облегчает понимание его структуры и функциональности (Рисунок 17).

  
Рисунок 17 – Декомпозиция IDEF0

WBS (Work Breakdown Structure) - это иерархическая декомпозиция работы, которая представляет собой иерархическую структуру задач, необходимых для выполнения проекта. WBS помогает управлять проектом, разбивая его на более мелкие и управляемые элементы работы (Рисунок 18).

  
Рисунок 18 – Диаграмма WBS

ER-диаграмма (Entity-Relationship diagram) - это графическое представление структуры данных, которое позволяет описать сущности (entities) и их взаимосвязи в базе данных. ER-диаграммы широко используются в проектировании баз данных для моделирования информационных систем (Рисунок 19).

  
Рисунок 19 – ER-диаграмма

## Обоснование выбора средств разработки

Разработка приложения по учету и организации рабочего времени сотрудников предприятия на основе Visual Studio предполагает обоснование выбора соответствующих средств разработки. В данном контексте, выбор средств разработки обуславливается следующими факторами:

* Производительность и эффективность: Средства разработки WPF-приложений, основанные на технологии XAML, обеспечивают высокую производительность и эффективность при создании пользовательского интерфейса и взаимодействии с данными.
* Гибкость и масштабируемость: Использование XAML позволяет создавать гибкие и масштабируемые приложения, способные адаптироваться к изменяющимся требованиям бизнеса.
* Поддержка сообществом и обновлений: Выбор средств разработки, пользующихся активной поддержкой сообщества разработчиков и регулярными обновлениями, обеспечивает стабильную и долгосрочную поддержку проекта.

При выборе SQL Server Management Studio (SSMS) в качестве средства разработки баз данных обоснование основано на следующих преимуществах:

* Нативная поддержка SQL Server: SSMS разработан специально для работы с SQL Server, что обеспечивает нативную интеграцию и полный спектр возможностей для администрирования, оптимизации и управления базами данных на платформе SQL Server.
* Мощные инструменты управления: SSMS предоставляет широкий функционал для создания, изменения и управления базами данных, таблицами, представлениями, хранимыми процедурами и другими объектами базы данных.
* Интуитивно понятный интерфейс: Удобный и интуитивно понятный пользовательский интерфейс SSMS делает работу с базами данных более эффективной и позволяет разработчикам и администраторам легко выполнять разнообразные задачи.

## Разработка базы данных

SQL (Structured Query Language) - это язык программирования, используемый для управления реляционными базами данных. Он предоставляет возможность создания, изменения, удаления и извлечения данных из базы данных. Вот некоторые ключевые аспекты и возможности SQL:

* Операторы: SQL включает в себя различные операторы для выполнения различных операций с данными, такие как SELECT (выборка данных), INSERT (добавление данных), UPDATE (обновление данных) и DELETE (удаление данных).
* Клаузы: SQL также включает в себя клаузы для уточнения запросов, например, WHERE (условие выборки), ORDER BY (сортировка результатов), GROUP BY (группировка данных), HAVING (условие группировки) и другие, что делает запросы гибкими и мощными.
* Функции: SQL предоставляет разнообразные функции для обработки данных в запросах, такие как арифметические операции, функции агрегирования (SUM, AVG, COUNT и т.д.), функции преобразования данных и многие другие.
* Транзакции: SQL поддерживает транзакции, что позволяет выполнять несколько операций как единое целое, обеспечивая целостность данных и откат изменений в случае возникновения ошибок.
* Управление доступом: SQL позволяет управлять доступом к данным через механизмы авторизации и разграничения прав доступа, обеспечивая безопасность данных и конфиденциальность информации.
* Создание и управление объектами базы данных: SQL позволяет создавать и управлять структурой базы данных, включая таблицы, индексы, представления, хранимые процедуры и другие объекты, необходимые для организации данных и их эффективного использования.
* Начать следует с создания базы данных для хранения и взаимодействия с данными (Рисунок 20).

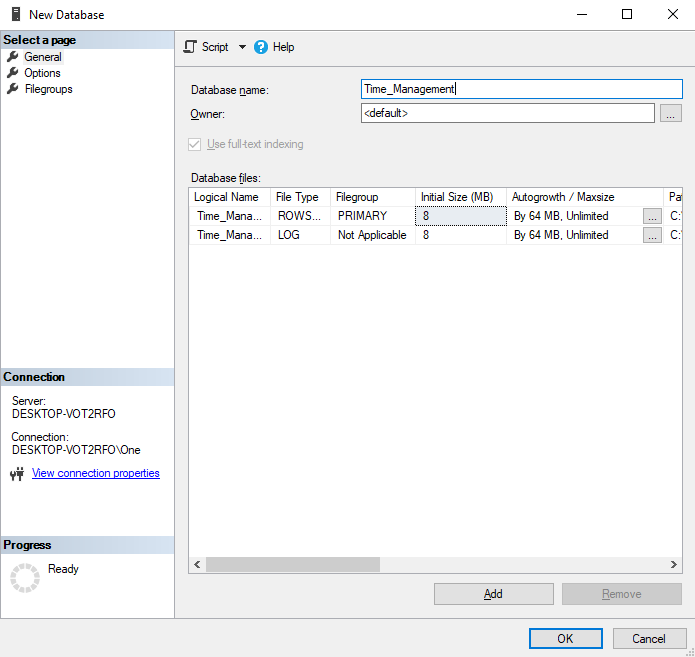


Рисунок 20 – Создание базы данных Time\_Managment

Далее понадобятся таблицы в которых будут хранятся данные, понадобятся таблицы: User – таблица в которой будут хранятся данные сотрудников (Рисунок 21). User\_data – таблица для хранения персональных данных (Рисунок 22). Role – таблица где будут хранится должности сотрудников (Рисунок 23). Task – таблица хранения задач сотрудников (Рисунок 24). Statuses – таблица с данными статусов (Рисунок 25).

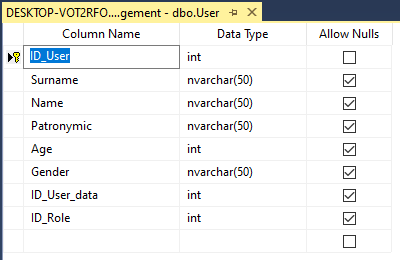


Рисунок 21 – Данные таблицы User

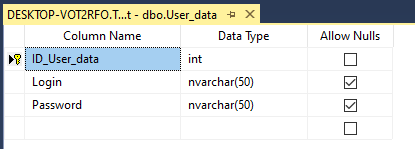


Рисунок 22 – Данные таблицы User\_data

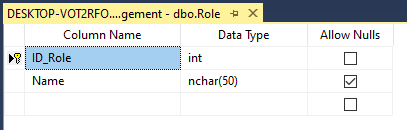


Рисунок 23 – Данные таблицы Role

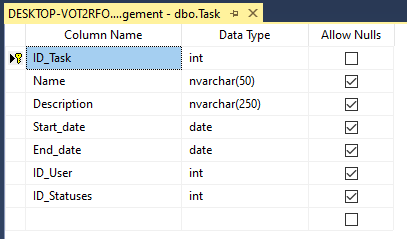


Рисунок 24 – Данные таблицы Task

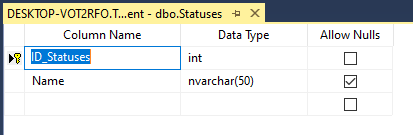


Рисунок 25 – Данные таблицы Statuses

Для работы программы понадобится заполнить данными таблицы.

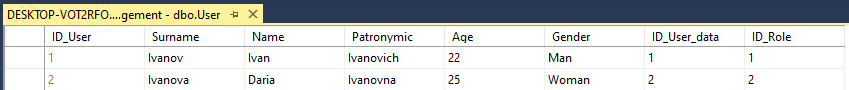


Рисунок 26 – Заполнение таблицы User

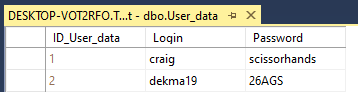


Рисунок 27 – Заполнение таблицы User\_data

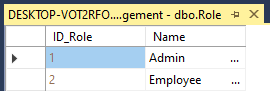


Рисунок 28 – Заполнение таблицы Role

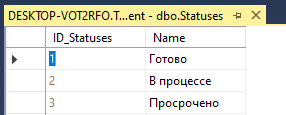


Рисунок 29 – Заполнение таблицы Statuses

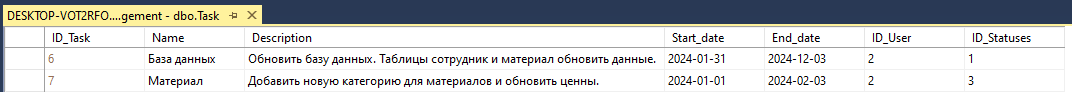


Рисунок 30 – Заполнение таблицы Task

Последним шагом составление диаграммы (Рисунок 31). Для установки связей.

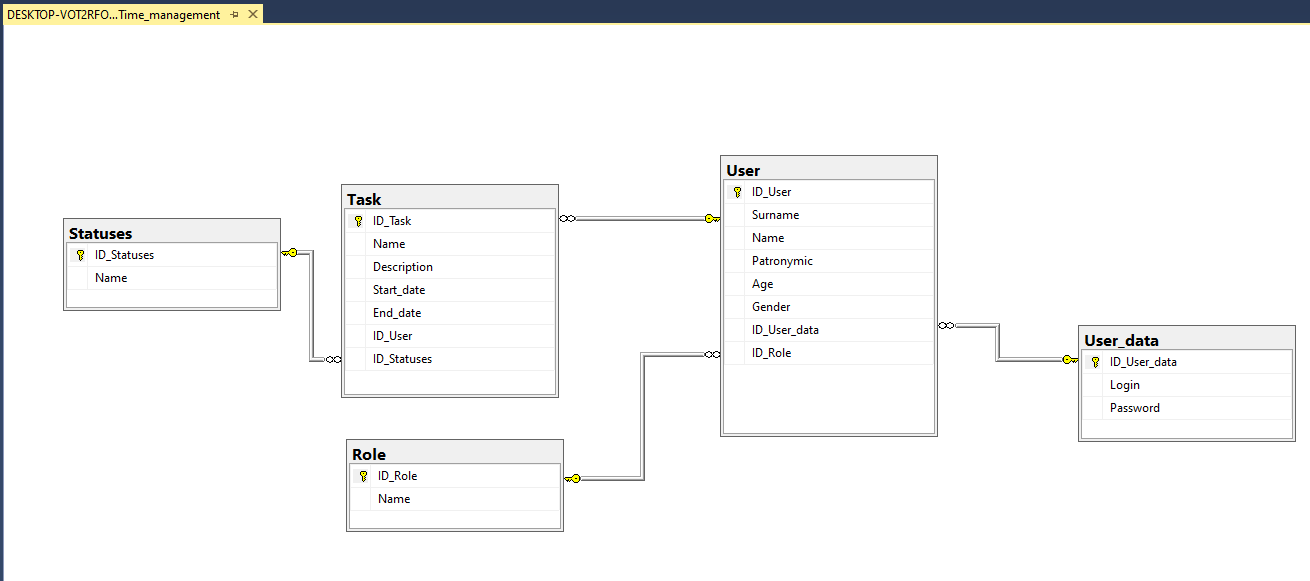
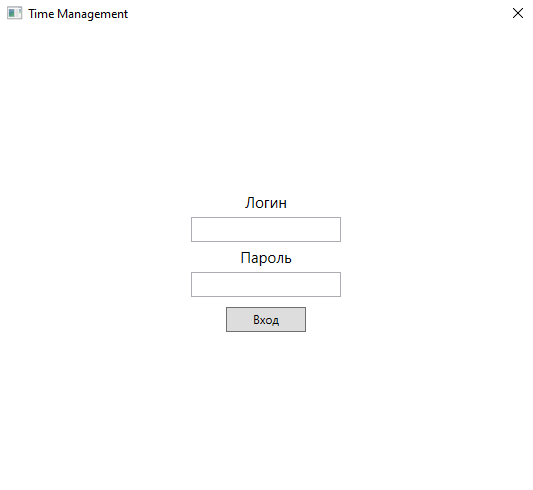


Рисунок 31 – Диаграмма в SQL Server Management Studio

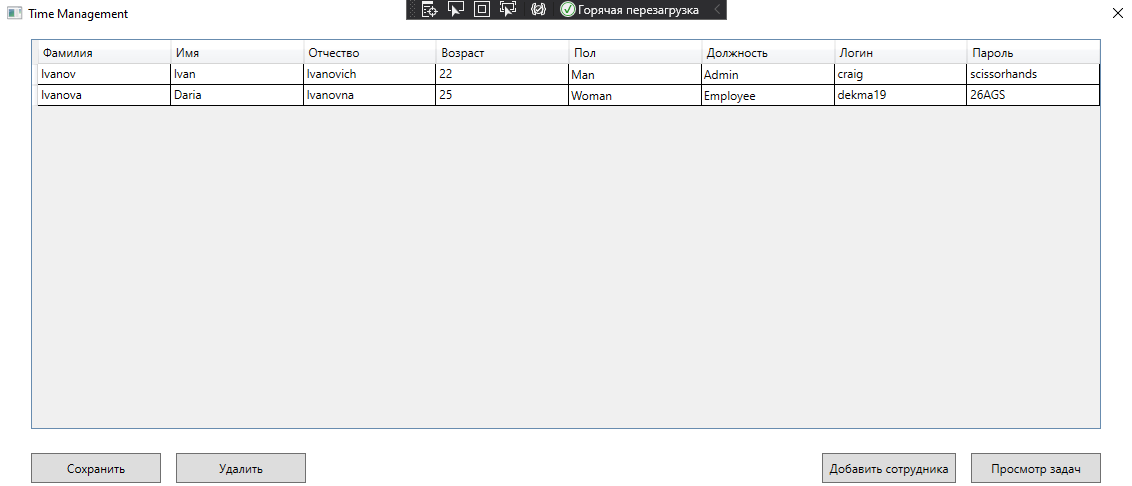
## Проектирование меню и интерфейса программного обеспечения

При проектировании меню и интерфейса программного обеспечения, включая разработку WPF-приложения для создания, отслеживания и управления задачами на XAML, следует уделить особое внимание архитектуре пользовательского интерфейса. Грамотное построение меню и пользовательского интерфейса способствует удобству использования приложения, повышает эффективность работы пользователя и улучшает пользовательский опыт.

Первой встречающей страницей будет вход где пользователь будет вводить свои данные (Рисунок 32).

  
Рисунок 32 – Страница входа

Следующая страница главное меню администратора на ней находятся четыре кнопки и одна таблица с данными сотрудников. Кнопка сохранить отвечает за сохранения внесенных изменений в поля сотрудников. Кнопка удалить, удаляет данные сотрудников. Добавить сотрудника перенаправляет на страницу с полями заполнения данных нового сотрудника и последняя кнопка на странице просмотр задач предназначена для перехода на страницу с ознакомлением задач сотрудников (Рисунок 33).

  
Рисунок 33 – Главное меню администратора

Страница с добавлением сотрудника имеет четыре поля, два комбинированных списка и две кнопки, добавить отвечает за добавление самого пользователя, а кнопка назад нужна для возвращения в главное меню (Рисунок 34).

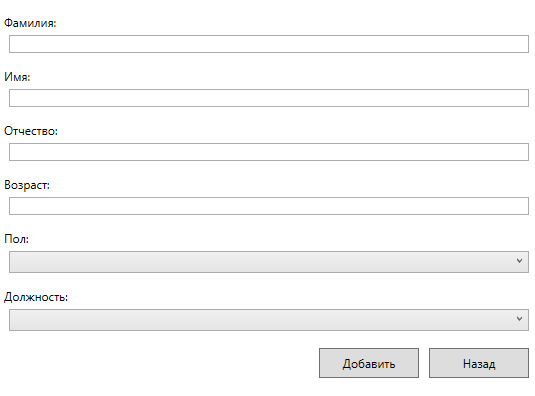


Рисунок 34 – Страница добавление сотрудника

Страница просмотр задач содержит: один комбинированный список выбора сотрудника одну таблицу с отображением задач сотрудников, кнопка просмотр для отображения задач в таблице, кнопка сохранить для сохранения внесенных изменений в задачу, кнопка добавить переход на страницу с добавлением сотруднику задач, и кнопка назад которая вернет администратора на главную страницу. (Рисунок 35)

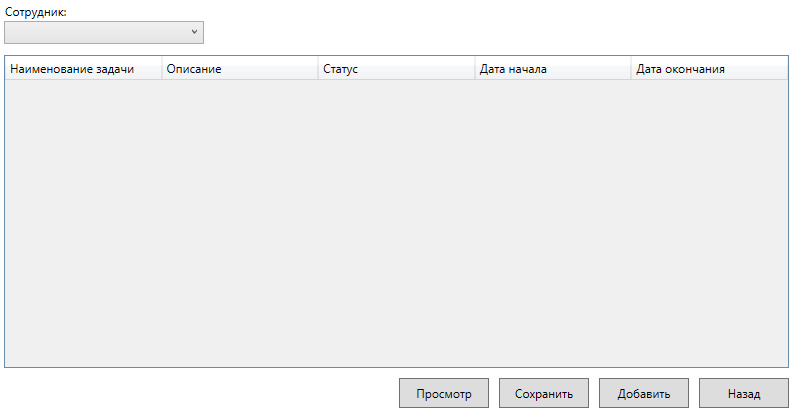


Рисунок 35 – Страница просмотра задач сотрудника

На странице с добавлением задачи сотруднику есть два комбинированных списка, два поля, два календаря и две кнопки, которые нужны для подтверждения внесённых данных и возвращения на прошлую страницу (Рисунок 36).

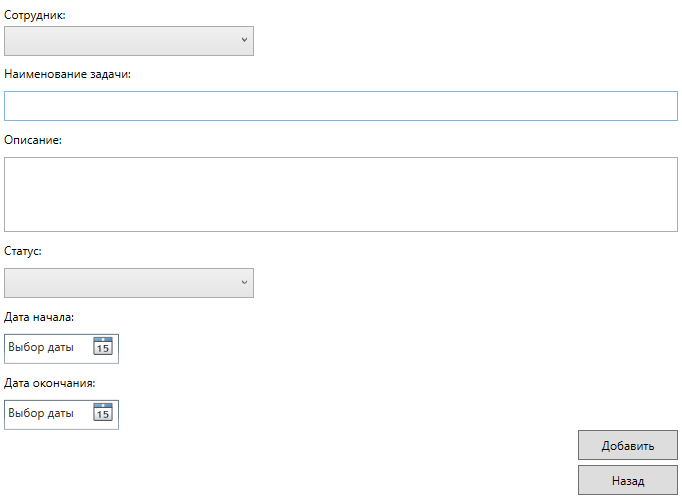


Рисунок 36 – Страница добавление задач для сотрудника

Далее будут представлены страницы сотрудника. На странице находится один календарь, четыре кнопки и одна таблица (Рисунок 37).

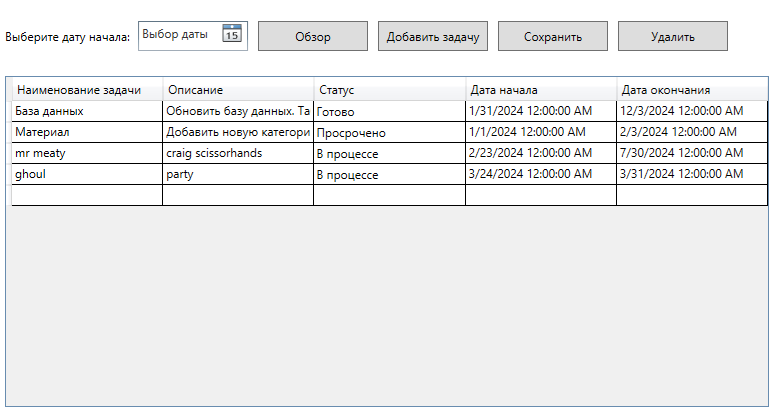


Рисунок 38 – Главное меню сотрудника

Нажав на кнопку добавить задачу откроется страница с добавлением задач для сотрудника. На странице находится два поля, два календаря и две кнопки. (Рисунок 39)

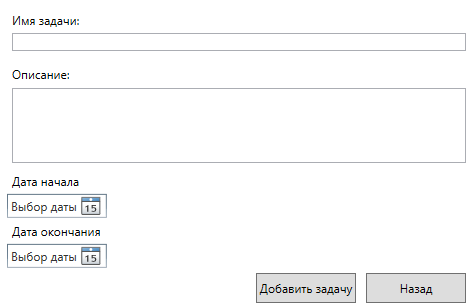


Рисунок 39 – Страница добавление задачи

## Спецификация программы

Программа предназначена для организации рабочих процессов в компаниях которым требуется составлять задания для ускорения работы сотрудников.

Цели и задачи:

* повышение эффективной работы сотрудников компании;
* повышение удобства коллективной разработки программного обеспечения.
* редактирование и контроль задач.

Функциональные требования

* Создание неограниченного числа таблиц для разных задач;
* Введение статусов задач для отображения этапа задачи (“Готово” “В процессе” “Просроченно”);
* Добавление выбранному сотруднику задачи;
* Редактирование и обновление информации о задачах.
* Синхронизация между пользователями.

Используемые технологии

* WPF;
* C#;
* SQL;

## Описание запросов при разработке программного обеспечения

При разработке программного обеспечения, в частности WPF-приложения для создания, отслеживания и управления задачами с использованием XAML, необходимо провести анализ требований и выделить ключевые запросы к базе данных. Описанные запросы должны обеспечивать эффективное выполнение операций над задачами, обеспечивать плавное взаимодействие с базой данных и обеспечивать необходимый функционал приложения.

Пример ViewTaskWindow:

1. FillComboBoxWithEmployees(): - Запрос для выборки ФИО сотрудников, имеющих определенную роль (ID\_Role = 2) из таблицы [User].

private void FillComboBoxWithEmployees()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT CONCAT(Surname, ' ', Name, ' ', Patronymic) AS FullName FROM [User] WHERE ID\_Role = 2";

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dt = new DataTable();

dataAdapter.Fill(dt);

if (dt.Rows.Count > 0)

{

userComboBox.ItemsSource = dt.DefaultView;

}

else

{

MessageBox.Show("Нет сотрудников с заданным ID\_Role");

}

}

}

2. FillComboBoxWithStatuses(): - Запрос для получения и заполнения комбо-бокса со статусами из таблицы Statuses.

private void FillComboBoxWithStatuses()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT ID\_Statuses, Name FROM Statuses";

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dt = new DataTable();

dataAdapter.Fill(dt);

if (dt.Rows.Count > 0)

{

foreach (DataRow row in dt.Rows)

{

StatusViewModel status = new StatusViewModel

{

ID = (int)row["ID\_Statuses"],

Name = row["Name"].ToString()

};

Statuses.Add(status);

}

}

}

}

3. ButtonView\_Click(): - Запрос для выборки задач, связанных с выбранным сотрудником на основе его ФИО.

private void ButtonView\_Click(object sender, RoutedEventArgs e)

{

if (userComboBox.SelectedItem != null)

{

Tasks.Clear();

DataRowView dataRow = (DataRowView)userComboBox.SelectedItem;

string selectedEmployeeFullName = dataRow["FullName"].ToString();

string queryTasks = "SELECT Task.ID\_Task, Task.Name, Task.Description, Task.Start\_date, Task.End\_date, Task.ID\_Statuses FROM Task INNER JOIN [User] ON Task.ID\_User = [User].ID\_User WHERE CONCAT([User].Surname, ' ', [User].Name, ' ', [User].Patronymic) = @EmployeeFullName";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(queryTasks, connection);

command.Parameters.AddWithValue("@EmployeeFullName", selectedEmployeeFullName);

SqlDataAdapter dataAdapter = new SqlDataAdapter(command);

DataTable dtTasks = new DataTable();

dataAdapter.Fill(dtTasks);

if (dtTasks.Rows.Count > 0)

{

foreach (DataRow row in dtTasks.Rows)

{

TaskViewModel task = new TaskViewModel

{

ID = Convert.ToInt32(row["ID\_Task"]),

Name = row["Name"].ToString(),

Description = row["Description"].ToString(),

Start\_date = Convert.ToDateTime(row["Start\_date"]),

End\_date = Convert.ToDateTime(row["End\_date"]),

Status = Statuses.FirstOrDefault(s => s.ID == Convert.ToInt32(row["ID\_Statuses"]))

};

Tasks.Add(task);

}

taskDataGrid.ItemsSource = Tasks;

}

else

{

MessageBox.Show("У выбранного сотрудника нет задач.");

}

}

}

else

{

MessageBox.Show("Пожалуйста, выберите сотрудника.");

}

}

4. ButtonSave\_Click(): - Запросы для обновления задач в базе данных. Происходит циклическое обновление данных о каждой задаче в соответствии с внесенными изменениями.

private void ButtonSave\_Click(object sender, RoutedEventArgs e)

{

if (userComboBox.SelectedItem == null)

{

MessageBox.Show("Выберите пользователя перед сохранением изменений.");

return;

}

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlTransaction transaction = connection.BeginTransaction())

{

foreach (TaskViewModel task in Tasks)

{

string queryUpdate = "UPDATE Task SET Name = @Name, Description = @Description, Start\_date = @Start\_date, End\_date = @End\_date, ID\_Statuses = @ID\_Statuses WHERE ID\_Task = @ID\_Task";

using (SqlCommand command = new SqlCommand(queryUpdate, connection, transaction))

{

command.Parameters.AddWithValue("@Name", task.Name);

command.Parameters.AddWithValue("@Description", task.Description);

command.Parameters.AddWithValue("@Start\_date", task.Start\_date);

command.Parameters.AddWithValue("@End\_date", task.End\_date);

command.Parameters.AddWithValue("@ID\_Statuses", task.Status.ID);

command.Parameters.AddWithValue("@ID\_Task", task.ID);

command.ExecuteNonQuery();

}

}

transaction.Commit();

FillDataGrid();

MessageBox.Show("Изменения успешно сохранены в базе данных.");

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при сохранении изменений: {ex.Message}");

}

}

5. FillDataGrid(): - Запрос для заполнения таблицы задач данными из базы данных на основе роли сотрудника.

private void FillDataGrid()

{

try

{

Tasks.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Task.ID\_Task, Task.Name, Task.Description, Task.Start\_date, Task.End\_date, Task.ID\_Statuses FROM Task INNER JOIN [User] ON Task.ID\_User = [User].ID\_User WHERE [User].ID\_Role = 2";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

TaskViewModel task = new TaskViewModel

{

ID = Convert.ToInt32(reader["ID\_Task"]),

Name = reader["Name"].ToString(),

Description = reader["Description"].ToString(),

Start\_date = Convert.ToDateTime(reader["Start\_date"]),

End\_date = Convert.ToDateTime(reader["End\_date"]),

Status = Statuses.FirstOrDefault(s => s.ID == Convert.ToInt32(reader["ID\_Statuses"]))

};

Tasks.Add(task);

}

taskDataGrid.ItemsSource = Tasks;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при заполнении DataGrid: {ex.Message}");

}

}

6. DeleteSelectedTask(): - Запрос для удаления выбранной задачи из базы данных.

private void DeleteSelectedTask()

{

if (taskDataGrid.SelectedItem is TaskViewModel)

{

int selectedIndex = taskDataGrid.SelectedIndex;

if (taskDataGrid.Items.Count > 0 && selectedIndex >= 0 && selectedIndex < taskDataGrid.Items.Count)

{

TaskViewModel task = taskDataGrid.Items.GetItemAt(selectedIndex) as TaskViewModel;

int taskId = task.ID;

MessageBoxResult result = MessageBox.Show("Вы хотите удалить запись?", "Подтверждение удаления", MessageBoxButton.YesNo);

if (result == MessageBoxResult.Yes)

{

string queryDelete = "DELETE FROM Task WHERE ID\_Task = @TaskID";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(queryDelete, connection))

{

command.Parameters.AddWithValue("@TaskID", taskId);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Выбранная задача успешно удалена из базы данных.");

FillDataGrid();

}

else

{

MessageBox.Show("Не удалось удалить выбранную задачу из базы данных.");

}

}

}

}

else if (result == MessageBoxResult.No)

{

MessageBox.Show("Запись не была удалена.");

}

}

else

{

MessageBox.Show("Пожалуйста, выберите задачу для удаления.");

}

}

else

{

MessageBox.Show("Пожалуйста, выберите задачу для удаления.");

}

}

## Администрирование программного обеспечения

Управление программным обеспечением (ПО) включает в себя координацию процессов установки, технического обслуживания и поддержки прикладных программ в рамках информационной системы.

Одной из ключевых частей администрирования ПО является обеспечение безопасного доступа пользователей к приложению. Для этого в приложении реализованы механизмы авторизации пользователей. Авторизация позволяет контролировать доступ пользователя к определенным данным в приложении. Имея у администратора возможность создавать и удалять сотрудников это обеспечивает избегать посторонних пользователей.

Также важный аспект администрирования ПО – это управление правами пользователей.

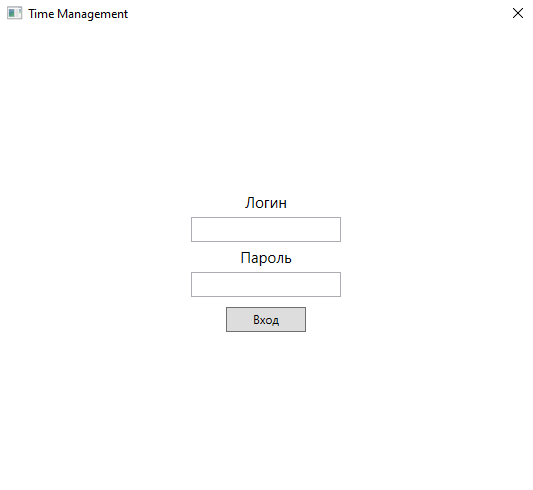
В системе не предусмотрены отдельные роли.

# описание работы программного обеспечения

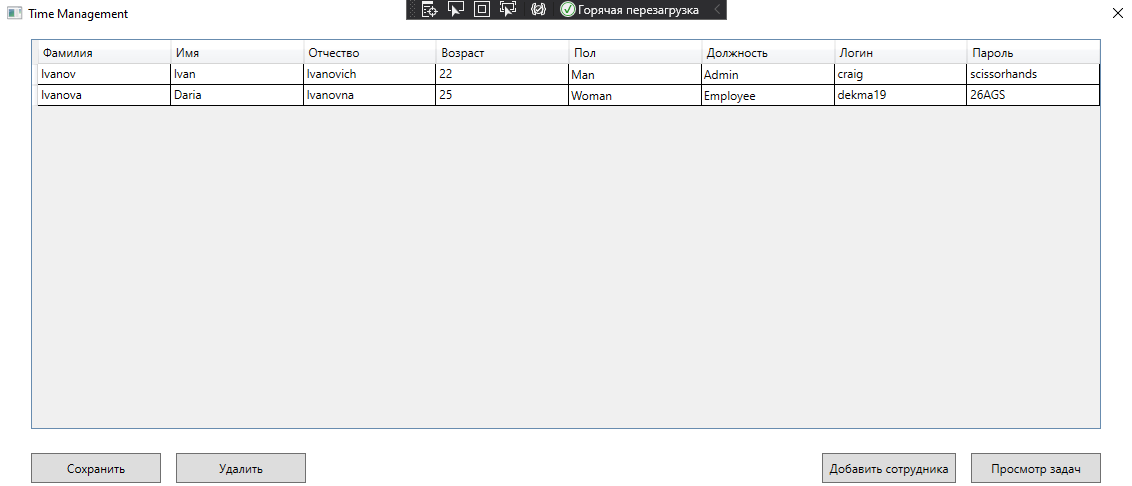
## Руководство пользователя

Данное руководство предназначено для знакомства пользователей с основными возможностями и функционалом веб-приложения, а также для обучения оптимальным способам использования его возможностей.

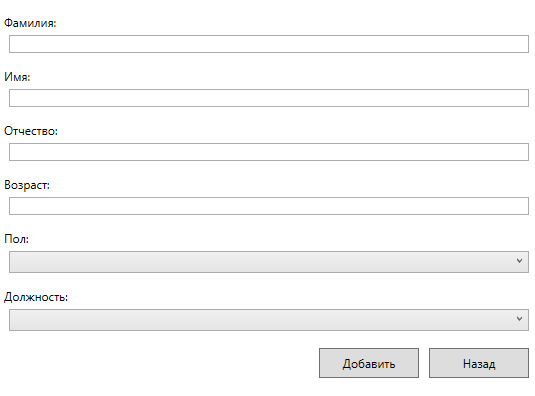
Первое что встречает пользователя это страница входа, которая в зависимости распределит его либо в администраторы, либо в сотрудники (Рисунок 40).

  
Рисунок 40 – Страница вход

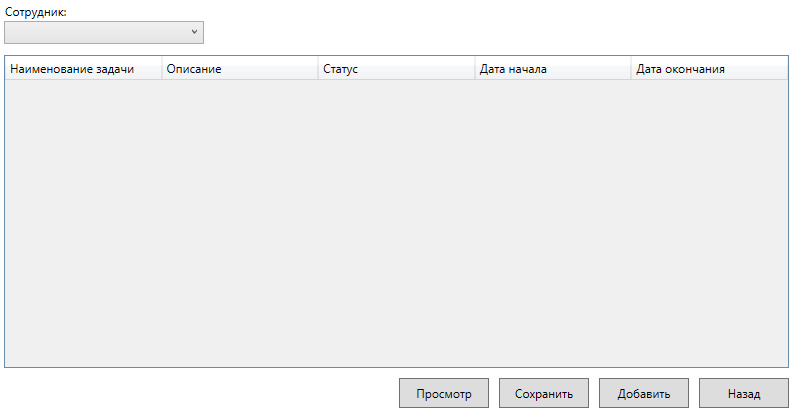
В основном меню администратора размещены четыре кнопки и таблица с данными о сотрудниках. Кнопка "Сохранить" предназначена для сохранения внесенных изменений в информацию о сотрудниках. Кнопка "Удалить" используется для удаления данных о сотрудниках. "Добавить сотрудника" перенаправляет на страницу с формой для заполнения информации о новом сотруднике, а кнопка "Просмотр задач" предназначена для перехода на страницу с описанием задач, назначенных сотрудникам (Рисунок 41).

  
Рисунок 41 – Главное меню администратора

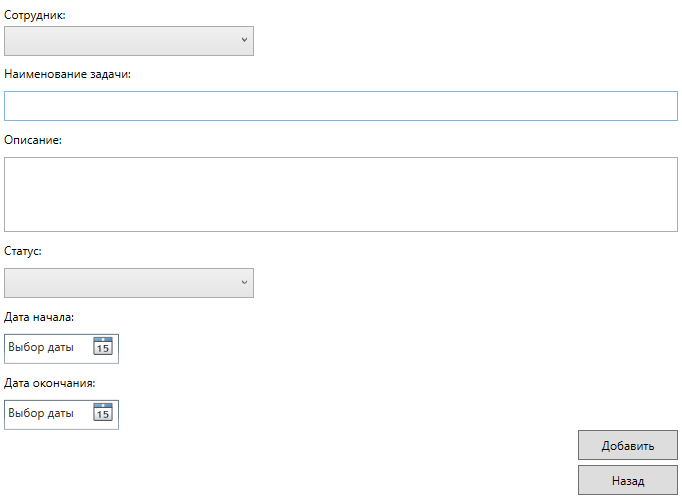
На странице добавления сотрудника присутствуют четыре поля ввода, два выпадающих списка и две кнопки. Кнопка "Добавить" предназначена для внесения нового пользователя, а кнопка "Назад" используется для возврата в главное меню (Рисунок 42).

  
Рисунок 42 – Добавить сотрудника

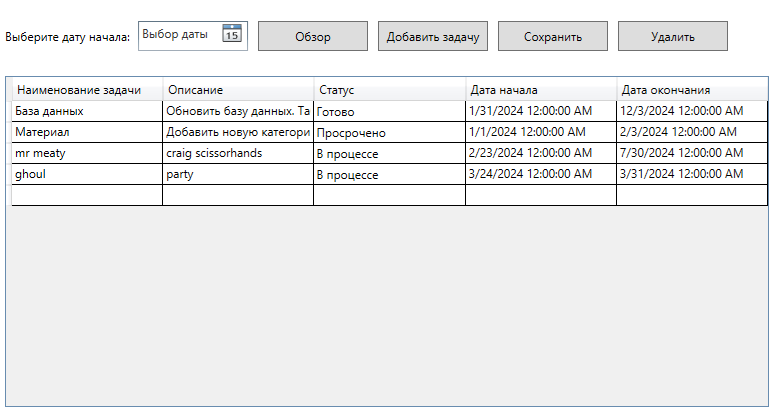
На странице просмотра задач присутствует: один выпадающий список для выбора сотрудника, одна таблица с задачами сотрудников, кнопка "Просмотр" для обновления задач в таблице, кнопка "Сохранить" для сохранения изменений в задаче, кнопка "Добавить" для перехода к добавлению задачи сотруднику, и кнопка "Назад", возвращающая администратора на главную страницу (Рисунок 43).

  
Рисунок 44 – Просмотр задач

На странице добавления задачи сотруднику находятся два выпадающих списка, два поля для ввода, два календаря и две кнопки: одна для подтверждения введенных данных и другая для возврата на предыдущую страницу (Рисунок 45).

  
Рисунок 45 – Добавление задач сотруднику

У сотрудника главное меню (Рисунок 46) отображается иначе, будут представлены страницы сотрудника. На странице присутствует один календарь, четыре кнопки и одна таблица

  
Рисунок 46 – Главное меню сотрудника

После нажатия кнопки "Добавить задачу" пользователю будет предоставлена страница для добавления новых задач для сотрудника. На данной странице размещены два поля для ввода, два календаря и две кнопки для управления процессом (Рисунок 47).

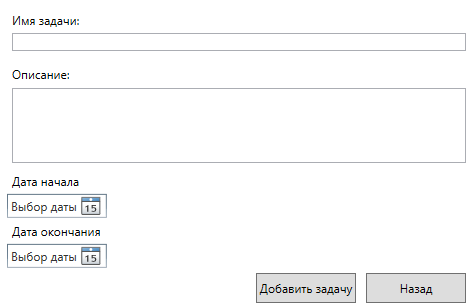


Рисунок 47 – Добавить задачу

## Руководство программиста

## Назначение и условия применения программы

WPF-приложение предназначено для удобного и эффективного управления задачами в рамках компании. Оно позволяет создавать задачи, редактировать их администратору для сотрудников, отслеживать и контролировать выполнение задач, а также оценивать состоянии задачи.

Функции, выполняемые программой

* Создание задач и проектов.
* Отслеживание выполнения задач.
* Контроль сроков выполнения задач.
* Добавление и удаление задач администратором для сотрудников.

Условия, необходимые для выполнения программы

* Наличие базы данных.
* Наличие учетных записей пользователей.

## Характеристика программы

* Интуитивно понятный интерфейс: обеспечивает удобную навигацию и понимание функционала программы.
* Расширенные функциональные возможности: позволяют пользователям создавать, отслеживать и управлять задачами с высокой эффективностью.
* Гибкость настроек: позволяет настраивать приоритеты, сроки выполнения, отображение задач и другие параметры в соответствии с потребностями пользователя.
* Отслеживание прогресса: предоставляет возможность контролировать выполнение задач и оценивать свой прогресс в рамках поставленных целей.
* Совместимость с технологией XAML: обеспечивает использование преимуществ этой технологии для создания гибкого и отзывчивого пользовательского интерфейса.

## Обращение к программе

Для запуска программы необходимо выполнить следующие действия:

* Запустите ярлык с приложением;
* Вводите свои данные;

## Организация входной информации

Данные для создания, изменения в программу поступают напрямую через SQL запросы, которые уже взаимодействуют с данными в БД.

## Организация выходной информации

Вывод данных происходит схожим образом через SQL запросы, которые будут взаимодействовать с базой данных.

## Сообщения

При успешном создании задачи, сотрудника и задачи для сотрудника появляется сообщение об успешном создании.

В случае неполного заполнения полей, программа потребует заполнить все обязательные поля.

При возникновении ошибок по каким-либо причинам, вместо страницы появится сообщение об этой причине.

# ЗАКЛЮЧЕНИЕ

В результате разработки WPF-приложения для создания, отслеживания и управления задачами на платформе XAML был достигнут значительный прогресс в области организации рабочего процесса и повышения производительности пользователей. Приложение представляет собой инновационный инструмент, спроектированный с учетом современных требований к управлению задачами и использования технологии XAML для создания гибкого и отзывчивого пользовательского интерфейса.

Основными характеристиками разработанного приложения являются интуитивно понятный интерфейс, расширенные функциональные возможности, гибкость в настройке задач, отслеживание прогресса выполнения и совместимость с технологией XAML. Пользователи получают возможность эффективно создавать, отслеживать и управлять задачами, что способствует улучшению организации рабочего процесса и достижению целей.

Важным аспектом разработки приложения была организация входной и выходной информации, что позволяет пользователям правильно структурировать задачи и анализировать результаты работы. Система безопасности приложения обеспечивает защиту конфиденциальности данных и надежность использования.

Разработанное WPF-приложение представляет собой важный инструмент для повышения производительности и эффективности управления задачами на платформе XAML. Его практическое применение может помочь пользователям достичь успеха в реализации проектов и управлении задачами в современной информационной среде.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гагарина, Л Г. Технология разработки программного обеспечения [Электронный ресурс] : учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 400 с. — Режим доступа: <https://new.znanium.com/read?id=336552>
2. Гагарина, Л .Г. Введение в архитектуру программного обеспечения [Электронный ресурс] : учебное пособие / Л. Г. Гагарина, А. Р. Федоров, П. А. Федоров. — Москва : ФОРУМ : ИНФРА-М, 2018. — 320 с. — Режим доступа: <https://new.znanium.com/read?id=315623>
3. Черткова, Е. А. Программная инженерия. Визуальное моделирование программных систем [Электронный ресурс] : учебник для среднего профессионального образования / Е. А. Черткова. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 147 с. — (Профессиональное образование). — ISBN 978-5-534-09823-5. — Режим доступа: <https://www.biblio-online.ru/bcode/441255>
4. Кундышева, Е. С. Математические методы и модели в экономике [Электронный ресурс] : учебник для бакалавров / Кундышева Е. С.; Под ред. Суслаков Б. А. - Москва :Дашков и К, 2017. - 286 с. ISBN 978-5-394-02488-7 - Режим доступа: <https://new.znanium.com/read?id=300594>
5. Новиков, А. И. Экономико-математические методы и модели [Электронный ресурс] : учебник для бакалавров / А. И. Новиков. - Москва :Дашков и К, 2017. - 532 с. ISBN 978-5-394-02615-7 - Режим доступа: <https://new.znanium.com/read?id=300712>
6. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности [Электронный ресурс] : учебное пособие / Г. Н. Федорова. — М. :КУРС : ИНФРА-М, 2019. — 336 с. (Среднее Профессиональное Образование). - Режим доступа: <https://new.znanium.com/read?id=330691>
7. Интуит Национальный открытый университет курс Язык UML 2 в анализе и проектировании программных систем и бизнес-процессов [Электронный ресурс]. – Режим доступа: <http://www.intuit.ru/studies/courses/480/336/info>, свободный.– Загл. с экрана. Яз. рус.3.3 Учебно-методическое обеспечение самостоятельной работы обучающихся
8. Ананьева, Т. Н. Стандартизация, сертификация и управление качеством программного обеспечения: Учебное пособие / Ананьева Т.Н., Новикова Н.Г., Исаев Г.Н. - М.:НИЦ ИНФРА-М, 2016. - 232 с.: [Электронный ресурс], Режим доступа: <http://znanium.com/bookread2.php?book=541003>
9. Шустова, Л.И. Базы данных: учебник / Л.И. Шустова, О.В. Тараканов. - Москва: ИНФРА-М, 2017. -- 304 с. + Доп, материалы [Электронный ресурс]. -- Режим доступа: https://new.znanium.com/catalog/document?id=205388 (Дата обращения: 01.06.2023).
10. Основы языка гипертекстовой разметки HTML и CSS [электронный ресурс] : учебное пособие /Сост. С. М. Наместников. – Ульяновск: УлГТУ, 2014. // Режим доступа: http://sernam.ru/book\_html.php
11. Quick Start // react: [сайт]. – 2023. – URL: https://medium.com/@mal7othify/lists-using-lazycolumn-in-jetpack-compose-c70c39805fbc – Текст: электронный.
12. Getting Started// vitejs: [сайт]. – 2023. – URL: <https://vitejs.dev/guide> – Текст: электронный.
13. React.js: понятное руководство для начинающих // habr: [сайт]. – 2018. – URL: <https://habr.com/ru/companies/ruvds/articles/428077> – Текст: электронный.
14. Документация NestJS // nestjs: [сайт]. – 2022. – URL: https://nestjs.ru (дата обращения: 22.02.2023). – Текст: электронный.
15. Изучаем PostgreSQL. // habr: [сайт]. – 2022. – URL: <https://habr.com/ru/companies/otus/articles/706346> – Текст: электронный.
16. Курсы по TypeScript: [Электронный ресурс] // URL: https://code-basics.com/ru/languages/typescript
17. Золотухина, Е. Б. Управление жизненным циклом информационных систем (продвинутый курс) [Электронный ресурс]: конспект лекций / Е. Б. Золотухина, С. А. Красникова, А. С. Вишня. – Москва: КУРС, НИЦ ИНФРА-М, 2017. – 119 с.: ISBN 978-5-906818-36-2–Режим доступа: <https://znanium.com/read?id=145707>.

# ПРИЛОЖЕНИЕ А

## Листинг программы

WPF страницы

<Page x:Class="TimeManagement.AddEmployeeWindow"

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:TimeManagement"

mc:Ignorable="d"

Title="AddEmployeeWindow" Height="400" Width="550">

<Grid Margin="10">

<StackPanel>

<TextBlock Text="Фамилия:"/>

<TextBox Name="SurnameTextBox" Margin="5"/>

<TextBlock Text="Имя:" Margin="0,10,0,0"/>

<TextBox Name="NameTextBox" Margin="5"/>

<TextBlock Text="Отчество:" Margin="0,10,0,0"/>

<TextBox Name="PatronymicTextBox" Margin="5"/>

<TextBlock Text="Возраст:" Margin="0,10,0,0"/>

<TextBox Name="AgeTextBox" Margin="5"/>

<TextBlock Text="Пол:" Margin="0,10,0,0"/>

<ComboBox Name="GenderComboBox" Margin="5">

<ComboBoxItem>Man</ComboBoxItem>

<ComboBoxItem>Woman</ComboBoxItem>

</ComboBox>

<TextBlock Text="Должность:" Margin="0,10,0,0"/>

<ComboBox Name="RoleComboBox" Margin="5" DisplayMemberPath="RoleName" SelectedValuePath="ID\_Role"/>

<StackPanel Orientation="Horizontal" Height="45" HorizontalAlignment="Right" VerticalAlignment="Bottom" Margin="0 5 0 0">

<Button Content="Добавить" Width="100" Height="30" Margin="5" Click="AddButton\_Click"/>

<Button Content="Назад" Width="100" Height="30" Margin="5" Click="BackButton\_Click"/>

</StackPanel>

</StackPanel>

</Grid>

</Page>

C# страницы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Data.SqlClient;

using TimeManagement.User\_folder;

using System.Data;

using System.Collections.ObjectModel;

namespace TimeManagement

{

/// <summary>

/// Логика взаимодействия для AddEmployeeWindow.xaml

/// </summary>

public partial class AddEmployeeWindow : Page

{

public AddEmployeeWindow()

{

InitializeComponent();

Loaded += MainWindow\_Loaded;

Loaded += AddEmployeePage\_Loaded;

}

private void MainWindow\_Loaded(object sender, RoutedEventArgs e)

{

UserRoleManager manager = new UserRoleManager();

DataTable rolesTable = manager.GetRoles();

List<Role> roles = new List<Role>();

foreach (DataRow row in rolesTable.Rows)

{

roles.Add(new Role { ID\_Role = (int)row["ID\_Role"], Name = row["Name"].ToString() });

}

RoleComboBox.ItemsSource = roles;

RoleComboBox.DisplayMemberPath = "Name";

RoleComboBox.SelectedValuePath = "ID\_Role";

}

private void AddEmployeePage\_Loaded(object sender, RoutedEventArgs e)

{

Window mainWindow = Application.Current.MainWindow;

if (mainWindow != null)

{

mainWindow.Width = 560;

mainWindow.Height = 435;

double screenWidth = SystemParameters.PrimaryScreenWidth;

double screenHeight = SystemParameters.PrimaryScreenHeight;

mainWindow.Left = (screenWidth - mainWindow.Width) / 2;

mainWindow.Top = (screenHeight - mainWindow.Height) / 2;

}

}

public class Role

{

public int ID\_Role { get; set; }

public string Name { get; set; }

}

public class User

{

public string Gender { get; set; }

}

private readonly string connectionString = DBConnect.connectDB;

public void AddButton\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(SurnameTextBox.Text) || string.IsNullOrWhiteSpace(NameTextBox.Text) ||

string.IsNullOrWhiteSpace(PatronymicTextBox.Text) || string.IsNullOrWhiteSpace(AgeTextBox.Text) ||

RoleComboBox.SelectedValue == null || GenderComboBox.SelectedItem == null)

{

MessageBox.Show("Пожалуйста, заполните все обязательные поля (Фамилия, Имя, Отчество, Возраст, Роль, Пол).");

return;

}

int roleID = (int)RoleComboBox.SelectedValue;

string selectedGender = (string)((ComboBoxItem)GenderComboBox.SelectedItem).Content;

string login = GenerateLogin(SurnameTextBox.Text, NameTextBox.Text);

string password = GeneratePassword();

User user = new User();

if (selectedGender == "Man")

{

user.Gender = "Man";

}

else if (selectedGender == "Woman")

{

user.Gender = "Woman";

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlTransaction transaction = connection.BeginTransaction();

string userQuery = "INSERT INTO [User] (Surname, [Name], Patronymic, Age, Gender, ID\_Role, ID\_User\_data) VALUES (@Surname, @Name, @Patronymic, @Age, @Gender, @ID\_Role, @ID\_User\_data)";

string userDataQuery = "INSERT INTO User\_data (Login, Password) VALUES (@Login, @Password); SELECT SCOPE\_IDENTITY();";

int userDataId = 0;

using (SqlCommand userDataCommand = new SqlCommand(userDataQuery, connection, transaction))

{

userDataCommand.Parameters.AddWithValue("@Login", login);

userDataCommand.Parameters.AddWithValue("@Password", password);

userDataId = Convert.ToInt32(userDataCommand.ExecuteScalar());

using (SqlCommand userCommand = new SqlCommand(userQuery, connection, transaction))

{

userCommand.Parameters.AddWithValue("@Surname", SurnameTextBox.Text);

userCommand.Parameters.AddWithValue("@Name", NameTextBox.Text);

userCommand.Parameters.AddWithValue("@Patronymic", PatronymicTextBox.Text);

userCommand.Parameters.AddWithValue("@Age", AgeTextBox.Text);

userCommand.Parameters.AddWithValue("@Gender", selectedGender);

userCommand.Parameters.AddWithValue("@ID\_Role", roleID);

userCommand.Parameters.AddWithValue("@ID\_User\_data", userDataId);

userCommand.ExecuteNonQuery();

}

transaction.Commit();

}

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

}

}

private string GenerateLogin(string surname, string name)

{

string login = $"{name.ToLower().Substring(0, 2)}{surname.ToLower()}";

return login;

}

private string GeneratePassword()

{

string validChars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^&\*";

StringBuilder sb = new StringBuilder();

Random rnd = new Random();

while (sb.Length < 8)

{

sb.Append(validChars[rnd.Next(validChars.Length)]);

}

return sb.ToString();

}

private void BackButton\_Click(object sender, RoutedEventArgs e)

{

if (NavigationService.CanGoBack)

{

NavigationService.GoBack();

}

}

public DataTable GetRoles()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT ID\_Role, Name FROM Role";

using (SqlCommand command = new SqlCommand(query, connection))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable rolesTable = new DataTable();

adapter.Fill(rolesTable);

return rolesTable;

}

}

}

public class UserRoleManager

{

private readonly string connectionString = DBConnect.connectDB;

public DataTable GetRoles()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT ID\_Role, Name FROM Role";

using (SqlCommand command = new SqlCommand(query, connection))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable rolesTable = new DataTable();

adapter.Fill(rolesTable);

return rolesTable;

}

}

}

}

}

}

WPF

<Page x:Class="TimeManagement.AddTaskForEmployeeWindow"

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:TimeManagement"

mc:Ignorable="d"

d:DesignHeight="500" d:DesignWidth="700"

Title="AddTaskForEmployeeWindow">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" MinHeight="59" />

<RowDefinition Height="Auto" MinHeight="200" />

</Grid.RowDefinitions>

<Label Grid.Row="0" Grid.Column="0" FontSize="12">Сотрудник:</Label>

<ComboBox x:Name="comboBoxEmployee" Grid.Row="0" Margin="5,20,0,0" Width="250" Height="30" HorizontalAlignment="Left" ItemsSource="{Binding Users}" SelectedItem="{Binding SelectedUser}" FontSize="15"/>

<StackPanel Grid.Row="1" Height="440">

<Label Grid.Row="0" Grid.Column="1" FontSize="12">Наименование задачи:</Label>

<TextBox x:Name="taskNameForEmployee" Text="{Binding TaskName}" Margin="5,5,5,5" Height="30" FontSize="20"/>

<Label Grid.Row="0" Grid.Column="1" FontSize="12">Описание:</Label>

<TextBox x:Name="taskDescriptionForEmployee" Text="{Binding Description}" Margin="5,5,5,5" Height="75" TextWrapping="Wrap"/>

<Label Grid.Row="0" Grid.Column="1" FontSize="12">Статус:</Label>

<ComboBox x:Name="taskStatusForEmployee" ItemsSource="{Binding Statuses}" SelectedItem="{Binding SelectedStatus}" DisplayMemberPath="Name" Margin="5,5,5,5" Height="30" Width="250" HorizontalAlignment="Left" FontSize="20"/>

<Label Grid.Row="0" Grid.Column="1" FontSize="12">Дата начала:</Label>

<DatePicker x:Name="taskStartDatePicker" SelectedDate="{Binding StartDate}" Margin="5,5,0,5" Height="30" Width="115" HorizontalAlignment="Left"/>

<Label Grid.Row="0" Grid.Column="1" FontSize="12">Дата окончания:</Label>

<DatePicker x:Name="taskEndDatePicker" SelectedDate="{Binding EndDate}" Margin="5,5,0,0" Height="30" Width="115" HorizontalAlignment="Left"/>

<Button Height="30" Width="100" HorizontalAlignment="Right" Margin="0 0 5 0" Click="ButtonAddTaskForEmployee\_Click">Добавить</Button>

<Button Height="30" Width="100" HorizontalAlignment="Right" Margin="0 5 5 0" Click="ButtonBack\_Click">Назад</Button>

</StackPanel>

</Grid>

</Page>

C#

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using TimeManagement.User;

using System.Globalization;

using System.Collections.ObjectModel;

using System.Data;

namespace TimeManagement

{

/// <summary>

/// Логика взаимодействия для AddTaskForEmployeeWindow.xaml

/// </summary>

public partial class AddTaskForEmployeeWindow : Page

{

private readonly string connectionString = DBConnect.connectDB;

public ObservableCollection<string> Users { get; set; }

public ObservableCollection<StatusViewModel> Statuses { get; set; }

public int UserID { get; set; }

public AddTaskForEmployeeWindow()

{

InitializeComponent();

Statuses = new ObservableCollection<StatusViewModel>();

taskStatusForEmployee.ItemsSource = Statuses;

FillComboBoxWithStatuses();

Users = new ObservableCollection<string>();

Loaded += ViewTaskWindow\_Loaded;

comboBoxEmployee.ItemsSource = Users;

FillComboBoxWithEmployees();

}

private void FillComboBoxWithEmployees()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT ID\_User, CONCAT(Surname, ' ', Name, ' ', Patronymic) AS FullName FROM [User] WHERE ID\_Role = 2";

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dt = new DataTable();

dataAdapter.Fill(dt);

foreach (DataRow row in dt.Rows)

{

int userID = (int)row["ID\_User"];

string fullName = row["FullName"].ToString();

Users.Add($"{userID} - {fullName}");

}

}

}

private void FillComboBoxWithStatuses()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT ID\_Statuses, Name FROM Statuses";

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dt = new DataTable();

dataAdapter.Fill(dt);

{

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

StatusViewModel status = new StatusViewModel

{

ID = Convert.ToInt32(reader["ID\_Statuses"]),

Name = reader["Name"].ToString()

};

Statuses.Add(status);

}

reader.Close();

}

}

}

private void ViewTaskWindow\_Loaded(object sender, RoutedEventArgs e)

{

Window mainWindow = Application.Current.MainWindow;

if (mainWindow != null)

{

mainWindow.Width = 700;

mainWindow.Height = 540;

double screenWidth = SystemParameters.PrimaryScreenWidth;

double screenHeight = SystemParameters.PrimaryScreenHeight;

mainWindow.Left = (screenWidth - mainWindow.Width) / 2;

mainWindow.Top = (screenHeight - mainWindow.Height) / 2;

}

}

public class StatusViewModel

{

public int ID { get; set; }

public string Name { get; set; }

}

private void ButtonAddTaskForEmployee\_Click(object sender, RoutedEventArgs e)

{

if (comboBoxEmployee.SelectedItem == null)

{

MessageBox.Show("Выберите сотрудника!");

return;

}

if (taskStatusForEmployee.SelectedItem == null)

{

MessageBox.Show("Выберите статус задачи!");

return;

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string selectedUser = comboBoxEmployee.SelectedItem.ToString();

int userID = int.Parse(selectedUser.Split('-')[0].Trim());

string query = "INSERT INTO Task (Name, Description, ID\_Statuses, Start\_date, End\_date, ID\_User) " +

"VALUES (@Name, @Description, @ID\_Statuses, @Start\_date, @End\_date, @ID\_User)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Name", taskNameForEmployee.Text);

command.Parameters.AddWithValue("@Description", taskDescriptionForEmployee.Text);

command.Parameters.AddWithValue("@ID\_Statuses", ((StatusViewModel)taskStatusForEmployee.SelectedItem).ID);

command.Parameters.AddWithValue("@Start\_date", taskStartDatePicker.SelectedDate);

command.Parameters.AddWithValue("@End\_date", taskEndDatePicker.SelectedDate);

command.Parameters.AddWithValue("@ID\_User", userID);

command.ExecuteNonQuery();

MessageBox.Show("Задача успешно добавлена!");

}

}

private void ButtonBack\_Click(object sender, RoutedEventArgs e)

{

if (NavigationService.CanGoBack)

{

NavigationService.GoBack();

}

}

}

}

WPF

<Page x:Class="TimeManagement.AddTaskWindow"

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:TimeManagement"

mc:Ignorable="d" d:DesignWidth="500"

Title="AddTaskWindow" Height="335">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Text="Имя задачи:" VerticalAlignment="Center" Margin="5,2,5,0"/>

<TextBox Grid.Row="1" Name="txtTaskName" VerticalAlignment="Center" Margin="5"/>

<TextBlock Grid.Row="2" Text="Описание:" VerticalAlignment="Center" Margin="5,10,5,1"/>

<TextBox Grid.Row="3" Name="txtDescription" VerticalAlignment="Center" Margin="5" Height="75" TextWrapping="Wrap"/>

<StackPanel Grid.Row="4" VerticalAlignment="Bottom">

<Label Content="Дата начала" HorizontalAlignment="Left" Grid.Row="4" VerticalAlignment="Top"/>

<DatePicker HorizontalAlignment="Left" Grid.Row="4" VerticalAlignment="Top" Name="datefirst"/>

<Label Content="Дата окончания" HorizontalAlignment="Left" Grid.Row="4" VerticalAlignment="Top"/>

<DatePicker HorizontalAlignment="Left" Grid.Row="4" VerticalAlignment="Top" Name="datesecond"/>

<WrapPanel HorizontalAlignment="Right">

<Button Content="Добавить задачу" Width="100" Click="AddTaskButton\_Click" Margin="5" Height="30"/>

<Button Content="Назад" Width="100" Click="ReturnButton\_Click" Margin="5" Height="30"/>

</WrapPanel>

</StackPanel>

</Grid>

</Page>

C#

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using TimeManagement.User\_folder;

namespace TimeManagement

{

/// <summary>

/// Логика взаимодействия для AddTaskWindow.xaml

/// </summary>

public partial class AddTaskWindow : Page

{

private readonly string connectionString = DBConnect.connectDB;

public int UserID { get; set; }

public AddTaskWindow(int userID)

{

InitializeComponent();

UserID = userID;

Loaded += AddTaskWindow\_Loaded;

}

private void AddTaskWindow\_Loaded(object sender, RoutedEventArgs e)

{

Window mainWindow = Application.Current.MainWindow;

if (mainWindow != null)

{

mainWindow.Width = 500;

mainWindow.Height = 350;

double screenWidth = SystemParameters.PrimaryScreenWidth;

double screenHeight = SystemParameters.PrimaryScreenHeight;

mainWindow.Left = (screenWidth - mainWindow.Width) / 2;

mainWindow.Top = (screenHeight - mainWindow.Height) / 2;

}

}

private void AddTaskButton\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(txtTaskName.Text) || string.IsNullOrWhiteSpace(txtDescription.Text) || datefirst.SelectedDate == null || datesecond.SelectedDate == null)

{

MessageBox.Show("Пожалуйста, заполните все обязательные поля (Название задачи, Описание, Дата начала, Дата окончания).");

return;

}

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO [Task] (Name, Description, Start\_date, End\_date, ID\_User, ID\_Statuses) VALUES (@Name, @Description, @Start\_date, @End\_date, @ID\_User, @ID\_Statuses)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Name", txtTaskName.Text);

command.Parameters.AddWithValue("@Description", txtDescription.Text);

command.Parameters.AddWithValue("@Start\_date", datefirst.SelectedDate);

command.Parameters.AddWithValue("@End\_date", datesecond.SelectedDate);

command.Parameters.AddWithValue("@ID\_User", UserID);

command.Parameters.AddWithValue("@ID\_Statuses", 2);

command.ExecuteNonQuery();

}

MessageBox.Show("Задача успешно добавлена.");

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при добавлении задачи: {ex.Message}");

}

}

private void ReturnButton\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Uri("/EmployeePage.xaml", UriKind.Relative));

}

}

}

WPF

<Page x:Class="TimeManagement.AdminPage"

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"

mc:Ignorable="d"

d:DesignHeight="450"

Title="AdminPage" Width="1100">

<Grid>

<DataGrid x:Name="userDataGrid" ItemsSource="{Binding Users}" AutoGenerateColumns="False" IsReadOnly="False" Width="1070" Height="390" VerticalAlignment="Top" Margin="15,10,15,0" SelectedItem="{Binding SelectedUser, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" SelectionMode="Extended">

<DataGrid.Columns>

<DataGridTextColumn Header="Фамилия" Width="\*" Binding="{Binding Surname}"/>

<DataGridTextColumn Header="Имя" Width="\*" Binding="{Binding Name}"/>

<DataGridTextColumn Header="Отчество" Width="\*" Binding="{Binding Patronymic}"/>

<DataGridTextColumn Header="Возраст" Width="\*" Binding="{Binding Age}"/>

<DataGridComboBoxColumn Header="Пол" Width="\*" SelectedItemBinding="{Binding Gender, Mode=TwoWay}">

<DataGridComboBoxColumn.ElementStyle>

<Style TargetType="ComboBox">

<Setter Property="ItemsSource" Value="{Binding Genders, RelativeSource={RelativeSource AncestorType={x:Type Page}}}"/>

</Style>

</DataGridComboBoxColumn.ElementStyle>

<DataGridComboBoxColumn.EditingElementStyle>

<Style TargetType="ComboBox">

<Setter Property="ItemsSource" Value="{Binding Genders, RelativeSource={RelativeSource AncestorType={x:Type Page}}}"/>

</Style>

</DataGridComboBoxColumn.EditingElementStyle>

</DataGridComboBoxColumn>

<DataGridComboBoxColumn Header="Должность" Width="\*" SelectedItemBinding="{Binding Role}" DisplayMemberPath="Name">

<DataGridComboBoxColumn.ElementStyle>

<Style TargetType="ComboBox">

<Setter Property="ItemsSource" Value="{Binding Roles}" />

<Setter Property="DisplayMemberPath" Value="Name" />

<Setter Property="IsReadOnly" Value="True" />

</Style>

</DataGridComboBoxColumn.ElementStyle>

<DataGridComboBoxColumn.EditingElementStyle>

<Style TargetType="ComboBox">

<Setter Property="ItemsSource" Value="{Binding Roles}" />

<Setter Property="SelectedValuePath" Value="ID\_Role" />

<Setter Property="DisplayMemberPath" Value="Name" />

</Style>

</DataGridComboBoxColumn.EditingElementStyle>

</DataGridComboBoxColumn>

<DataGridTextColumn Header="Логин" Width="\*" Binding="{Binding User\_data.Login}"/>

<DataGridTextColumn Header="Пароль" Width="\*" Binding="{Binding User\_data.Password}"/>

</DataGrid.Columns>

</DataGrid>

<Button Content="Добавить сотрудника" Click="AddEmployee\_Click" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="0,0,160,7" Width="134" Height="30"/>

<Button Content="Просмотр задач" Click="ViewTask\_Click" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="0,0,15,7" Width="130" Height="30"/>

<Button Content="Сохранить" Click="EditUser\_Click" VerticalAlignment="Bottom" HorizontalAlignment="Left" Margin="15 0 0 7" Width="130" Height="30"/>

<Button Content="Удалить" Click="DeleteUser\_Click" VerticalAlignment="Bottom" HorizontalAlignment="Left" Margin="160 0 0 7" Width="130" Height="30"/>

</Grid>

</Page>

C#

using System;

using System.Collections;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace TimeManagement

{

/// <summary>

/// Логика взаимодействия для AdminPage.xaml

/// </summary>

public partial class AdminPage : Page

{

public UserViewModel SelectedUser { get; set; }

public List<string> Genders { get; } = new List<string> { "Man", "Woman" };

public ObservableCollection<RoleViewModel> Roles { get; set; }

public ObservableCollection<UserViewModel> Users { get; set; }

private readonly string connectionString = DBConnect.connectDB;

public AdminPage()

{

InitializeComponent();

Roles = new ObservableCollection<RoleViewModel>();

Users = new ObservableCollection<UserViewModel>();

Loaded += AdminPage\_Loaded;

LoadUsersFromDatabase();

}

public class UserDataViewModel

{

public int ID\_User\_data { get; set; }

public string Login { get; set; }

public string Password { get; set; }

}

public class RoleViewModel

{

public int ID\_Role { get; set; }

public string Name { get; set; }

public RoleViewModel() { }

}

public class UserViewModel

{

public List<string> Genders { get; } = new List<string> { "Man", "Woman" };

public int ID\_User { get; set; }

public string Surname { get; set; }

public string Name { get; set; }

public string Patronymic { get; set; }

public string Gender { get; set; }

public int Age { get; set; }

public RoleViewModel Role { get; set; }

public UserDataViewModel User\_data { get; set; }

public ObservableCollection<RoleViewModel> Roles { get; set; }

public UserViewModel(List<RoleViewModel> roles)

{

Roles = new ObservableCollection<RoleViewModel>(roles);

}

}

public void DeleteUserAndTasksFromDatabase(UserViewModel user)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlTransaction transaction = connection.BeginTransaction())

{

using (SqlCommand command = connection.CreateCommand())

{

command.Transaction = transaction;

try

{

string deleteTasksQuery = "DELETE FROM Task WHERE ID\_User = @UserID";

command.Parameters.Clear();

command.Parameters.AddWithValue("@UserID", user.ID\_User);

command.CommandText = deleteTasksQuery;

command.ExecuteNonQuery();

string deleteUserQuery = "DELETE FROM [User] WHERE ID\_User = @ID\_User";

command.Parameters.Clear();

command.Parameters.AddWithValue("@ID\_User", user.ID\_User);

command.CommandText = deleteUserQuery;

command.ExecuteNonQuery();

string deleteUserDataQuery = "DELETE FROM User\_data WHERE ID\_User\_data = @ID\_User\_data";

command.Parameters.Clear();

command.Parameters.AddWithValue("@ID\_User\_data", user.User\_data.ID\_User\_data);

command.CommandText = deleteUserDataQuery;

command.ExecuteNonQuery();

transaction.Commit();

MessageBox.Show("Пользователь и связанные задачи успешно удалены.", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

LoadUsersFromDatabase();

}

catch (Exception ex)

{

transaction.Rollback();

throw new Exception("Ошибка при удалении пользователя и задач: " + ex.Message);

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении пользователя и задач: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

public void LoadUsersFromDatabase()

{

List<RoleViewModel> roles = LoadRolesFromDatabase();

try

{

Users.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT u.ID\_User, u.Surname, u.Name, u.Patronymic, u.Gender, u.Age, ud.ID\_User\_data, ud.Login, ud.Password, r.ID\_Role, r.Name AS RoleName " +

"FROM [User] u " +

"INNER JOIN User\_data ud ON u.ID\_User\_data = ud.ID\_User\_data " +

"INNER JOIN Role r ON u.ID\_Role = r.ID\_Role";

SqlCommand command = new SqlCommand(query, connection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

foreach (DataRow row in dataTable.Rows)

{

int userRoleId = Convert.ToInt32(row["ID\_Role"]);

UserViewModel user = new UserViewModel(roles)

{

ID\_User = Convert.ToInt32(row["ID\_User"]),

Surname = row["Surname"].ToString(),

Name = row["Name"].ToString(),

Patronymic = row["Patronymic"].ToString(),

Gender = row["Gender"].ToString(),

Age = Convert.ToInt32(row["Age"]),

Role = roles.FirstOrDefault(r => r.ID\_Role == userRoleId),

User\_data = new UserDataViewModel

{

ID\_User\_data = Convert.ToInt32(row["ID\_User\_data"]),

Login = row["Login"].ToString(),

Password = row["Password"].ToString()

}

};

Users.Add(user);

}

userDataGrid.ItemsSource = Users;

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

public void UpdateUserInDatabase(UserViewModel user)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "UPDATE [User] " +

"SET Surname = @Surname, Name = @Name, Patronymic = @Patronymic, Gender = @Gender, Age = @Age, ID\_Role = @ID\_Role " +

"WHERE ID\_User = @ID\_User";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Surname", user.Surname);

command.Parameters.AddWithValue("@Name", user.Name);

command.Parameters.AddWithValue("@Patronymic", user.Patronymic);

command.Parameters.AddWithValue("@Gender", user.Gender);

command.Parameters.AddWithValue("@Age", user.Age);

command.Parameters.AddWithValue("@ID\_Role", user.Role.ID\_Role);

command.Parameters.AddWithValue("@ID\_User", user.ID\_User);

command.ExecuteNonQuery();

string updateUserDataQuery = "UPDATE User\_data SET Login = @Login, Password = @Password WHERE ID\_User\_data = @ID\_User\_data";

SqlCommand commandUserData = new SqlCommand(updateUserDataQuery, connection);

commandUserData.Parameters.AddWithValue("@Login", user.User\_data.Login);

commandUserData.Parameters.AddWithValue("@Password", user.User\_data.Password);

commandUserData.Parameters.AddWithValue("@ID\_User\_data", user.User\_data.ID\_User\_data);

commandUserData.ExecuteNonQuery();

MessageBox.Show("Данные пользователя успешно обновлены.", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при обновлении пользовательских данных: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

public List<RoleViewModel> LoadRolesFromDatabase()

{

List<RoleViewModel> roles = new List<RoleViewModel>();

try

{

string query = "SELECT ID\_Role, Name FROM Role";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

RoleViewModel role = new RoleViewModel

{

ID\_Role = Convert.ToInt32(reader["ID\_Role"]),

Name = reader["Name"].ToString()

};

roles.Add(role);

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при загрузке ролей.: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

return roles;

}

private void AdminPage\_Loaded(object sender, RoutedEventArgs e)

{

Window mainWindow = Application.Current.MainWindow;

if (mainWindow != null)

{

mainWindow.Width = 1150;

mainWindow.Height = 500;

double screenWidth = SystemParameters.PrimaryScreenWidth;

double screenHeight = SystemParameters.PrimaryScreenHeight;

mainWindow.Left = (screenWidth - mainWindow.Width) / 2;

mainWindow.Top = (screenHeight - mainWindow.Height) / 2;

this.DataContext = this;

LoadUsersFromDatabase();

}

}

private void AddEmployee\_Click(object sender, RoutedEventArgs e)

{

AddEmployeeWindow addEmployeeWindow = new AddEmployeeWindow();

this.NavigationService.Navigate(addEmployeeWindow);

}

private void ViewTask\_Click(object sender, RoutedEventArgs e)

{

ViewTaskWindow addViewTaskWindow = new ViewTaskWindow();

this.NavigationService.Navigate(addViewTaskWindow);

}

private void EditUser\_Click(object sender, RoutedEventArgs e)

{

if (SelectedUser != null)

{

UpdateUserInDatabase(SelectedUser);

}

else

{

MessageBox.Show("Пожалуйста, выберите пользователя для редактирования.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

private void DeleteUser\_Click(object sender, RoutedEventArgs e)

{

if (SelectedUser != null)

{

MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите удалить этого пользователя?", "Подтвердить", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

DeleteUserAndTasksFromDatabase(SelectedUser);

}

}

else

{

MessageBox.Show("Пожалуйста, выберите пользователя для удаления.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

WPF

<Page x:Class="TimeManagement.EmployeePage"

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:TimeManagement"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="EmployeePage">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0" Orientation="Horizontal" Margin="10">

<TextBlock Text="Выберите дату начала: " VerticalAlignment="Center"/>

<DatePicker Name="datePicker" Width="110" Height="30" Margin="5"/>

<Button Name="ViewButton" Content="Обзор" Click="ViewButton\_Click" Height="30" Width="110" Margin="5"/>

<Button Content="Добавить задачу" Height="30" Width="110" Margin="5" Click="AddTaskButton\_Click"/>

<Button Content="Сохранить" Click="SaveTaskButton\_Click" Height="30" Width="110" Margin="5"/>

<Button Content="Удалить" Click="DeleteTaskButton\_Click" Height="30" Width="110" Margin="5"/>

</StackPanel>

<DataGrid Grid.Row="1" x:Name="taskDataGrid" Margin="10" AutoGenerateColumns="False" ItemsSource="{Binding}" IsReadOnly="False">

<DataGrid.Columns>

<DataGridTextColumn Header="Наименование задачи" Width="\*" Binding="{Binding Name}"/>

<DataGridTextColumn Header="Описание" Width="\*" Binding="{Binding Description}"/>

<DataGridComboBoxColumn Header="Статус" Width="\*" TextBinding="{Binding StatusName}" DisplayMemberPath="Name">

<DataGridComboBoxColumn.ElementStyle>

<Style TargetType="ComboBox">

<Setter Property="ItemsSource" Value="{Binding Path=Statuses, RelativeSource={RelativeSource AncestorType={x:Type Page}}}"/>

<Setter Property="IsReadOnly" Value="True"/>

</Style>

</DataGridComboBoxColumn.ElementStyle>

<DataGridComboBoxColumn.EditingElementStyle>

<Style TargetType="ComboBox">

<Setter Property="ItemsSource" Value="{Binding Path=Statuses, RelativeSource={RelativeSource AncestorType={x:Type Page}}}"/>

</Style>

</DataGridComboBoxColumn.EditingElementStyle>

</DataGridComboBoxColumn>

<DataGridTextColumn Header="Дата начала" Width="\*" Binding="{Binding Start\_date}"/>

<DataGridTextColumn Header="Дата окончания" Width="\*" Binding="{Binding End\_date}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</Page>

C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using TimeManagement.User\_folder;

using System.Data.SqlClient;

using System.Data;

using TimeManagement.User;

using System.Security.Principal;

using System.Collections.ObjectModel;

namespace TimeManagement

{

/// <summary>

/// Логика взаимодействия для EmployeePage.xaml

/// </summary>

public partial class EmployeePage : Page

{

private readonly string connectionString = DBConnect.connectDB;

public ObservableCollection<StatusViewModel> Statuses { get; set; }

public ObservableCollection<UserViewModel> Users { get; set; }

public int UserID { get; set; }

public EmployeePage()

{

InitializeComponent();

Statuses = new ObservableCollection<StatusViewModel>();

Users = new ObservableCollection<UserViewModel>();

LoadUserData();

Loaded += EmployeePage\_Loaded;

FillComboBoxWithStatuses();

}

public class TaskViewModel

{

public int ID\_Task { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public DateTime Start\_date { get; set; }

public DateTime End\_date { get; set; }

public UserViewModel ID\_User { get; set; }

public StatusViewModel ID\_Statuses { get; set; }

}

public class UserViewModel

{

public List<string> Genders { get; } = new List<string> { "Man", "Woman" };

public int ID\_User { get; set; }

public string Surname { get; set; }

public string Name { get; set; }

public string Patronymic { get; set; }

public string Gender { get; set; }

public int Age { get; set; }

}

public class StatusViewModel

{

public int ID\_Statuses { get; set; }

public string Name { get; set; }

}

private void AddTaskButton\_Click(object sender, RoutedEventArgs e)

{

AddTaskWindow addTaskWindow = new AddTaskWindow(UserID);

this.NavigationService.Navigate(addTaskWindow);

}

private void EmployeePage\_Loaded(object sender, RoutedEventArgs e)

{

Window mainWindow = Application.Current.MainWindow;

if (mainWindow != null)

{

mainWindow.Width = 800;

mainWindow.Height = 450;

double screenWidth = SystemParameters.PrimaryScreenWidth;

double screenHeight = SystemParameters.PrimaryScreenHeight;

mainWindow.Left = (screenWidth - mainWindow.Width) / 2;

mainWindow.Top = (screenHeight - mainWindow.Height) / 2;

}

}

private void FillComboBoxWithStatuses()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT ID\_Statuses, Name FROM Statuses " +

"UNION " +

"SELECT DISTINCT Task.ID\_Statuses AS ID\_Statuses, Statuses.Name FROM Task " +

"JOIN Statuses ON Task.ID\_Statuses = Statuses.ID\_Statuses";

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dt = new DataTable();

dataAdapter.Fill(dt);

if (dt.Rows.Count > 0)

{

foreach (DataRow row in dt.Rows)

{

StatusViewModel status = new StatusViewModel

{

ID\_Statuses = (int)row["ID\_Statuses"],

Name = row["Name"].ToString()

};

Statuses.Add(status);

}

}

}

}

private string GetStatusName(int statusID)

{

string statusName = string.Empty;

string connect = DBConnect.connectDB;

using (SqlConnection connection = new SqlConnection(connect))

{

string query = "SELECT Name FROM Statuses WHERE ID\_Statuses = @StatusID";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@StatusID", statusID);

try

{

connection.Open();

object result = command.ExecuteScalar();

if (result != null)

{

statusName = result.ToString();

}

else

{

statusName = "Статус не найден";

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении запроса: " + ex.Message);

}

finally

{

connection.Close();

}

}

return statusName;

}

private void LoadUserData()

{

string connect = DBConnect.connectDB;

int userId = CurrentUser.ID\_User;

DataTable dataTable = new DataTable();

using (SqlConnection connection = new SqlConnection(connect))

{

string query = "SELECT Task.ID\_Task, Task.Name, Task.Description, Task.Start\_date, Task.End\_date, Statuses.Name AS StatusName " +

"FROM Task " +

"JOIN [User] ON [User].ID\_User = Task.ID\_User " +

"JOIN Statuses ON Task.ID\_Statuses = Statuses.ID\_Statuses " +

"WHERE [User].ID\_User = @ID\_User";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@ID\_User", userId);

try

{

connection.Open();

SqlDataAdapter dataAdapter = new SqlDataAdapter(command);

dataAdapter.Fill(dataTable);

foreach (DataRow row in dataTable.Rows)

{

string statusName = row["StatusName"].ToString();

int statusID = GetStatusID(statusName);

row["StatusName"] = GetStatusName(statusID);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении запроса: " + ex.Message);

}

}

taskDataGrid.ItemsSource = dataTable.DefaultView;

}

private int GetStatusID(string statusName)

{

int statusID = 0;

string connect = DBConnect.connectDB;

using (SqlConnection connection = new SqlConnection(connect))

{

string query = "SELECT ID\_Statuses FROM Statuses WHERE Name = @StatusName";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@StatusName", statusName);

try

{

connection.Open();

object result = command.ExecuteScalar();

if (result != null)

{

statusID = Convert.ToInt32(result);

}

else

{

Console.WriteLine("Условие не выполнено. Дополнительная обработка.");

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении запроса: " + ex.Message);

}

finally

{

connection.Close();

}

}

return statusID;

}

private void GetAndUpdateTask(DataRow row)

{

int taskID = (int)row["ID\_Task"];

string name = row["Name"].ToString();

string description = row["Description"].ToString();

DateTime startDate = (DateTime)row["Start\_date"];

DateTime endDate = (DateTime)row["End\_date"];

int statusID = GetStatusID(row["StatusName"].ToString());

string query = "UPDATE Task SET Name = @Name, Description = @Description, Start\_date = @StartDate, End\_date = @EndDate, ID\_Statuses = @StatusID WHERE ID\_Task = @TaskID";

using (SqlConnection connection = new SqlConnection(connectionString))

{

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@TaskID", taskID);

command.Parameters.AddWithValue("@Name", name);

command.Parameters.AddWithValue("@Description", description);

command.Parameters.AddWithValue("@StartDate", startDate);

command.Parameters.AddWithValue("@EndDate", endDate);

command.Parameters.AddWithValue("@StatusID", statusID);

try

{

connection.Open();

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

Console.WriteLine("Задача обновлена успешно.");

}

else

{

Console.WriteLine("Произошла ошибка при обновлении задачи.");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при обновлении задачи: {ex.Message}");

}

}

}

}

private void ViewButton\_Click(object sender, RoutedEventArgs e)

{

if (datePicker.SelectedDate != null)

{

DateTime startDate = datePicker.SelectedDate.Value;

int userId = CurrentUser.ID\_User;

string connect = DBConnect.connectDB;

using (SqlConnection connection = new SqlConnection(connect))

{

string query = "SELECT Task.Name, Task.Description, Task.Start\_date, Task.End\_date, Statuses.Name AS StatusName, Statuses.ID\_Statuses " +

"FROM Task " +

"JOIN [User] ON [User].ID\_User = Task.ID\_User " +

"JOIN Statuses ON Task.ID\_Statuses = Statuses.ID\_Statuses " +

"WHERE [User].ID\_User = @ID\_User AND Task.Start\_date = @StartDate";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@ID\_User", userId);

command.Parameters.AddWithValue("@StartDate", startDate);

DataTable dataTable = new DataTable();

try

{

connection.Open();

using (SqlDataReader reader = command.ExecuteReader())

{

dataTable.Load(reader);

}

foreach (DataRow row in dataTable.Rows)

{

int statusID = Convert.ToInt32(row["ID\_Statuses"]);

row["StatusName"] = GetStatusName(statusID);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении запроса3: " + ex.Message);

}

finally

{

connection.Close();

}

if (dataTable.Rows.Count > 0)

{

taskDataGrid.ItemsSource = dataTable.DefaultView;

}

else

{

MessageBox.Show("Нет задач для отображения.");

datePicker.SelectedDate = null;

LoadUserData();

}

}

}

else

{

MessageBox.Show("Выберите дату начала задачи.");

}

}

private void SaveTaskButton\_Click(object sender, RoutedEventArgs e)

{

DataTable dataTable = ((DataView)taskDataGrid.ItemsSource).ToTable();

foreach (DataRow row in dataTable.Rows)

{

GetAndUpdateTask(row);

}

MessageBox.Show("Изменения сохранены.");

}

private void DeleteTaskButton\_Click(object sender, RoutedEventArgs e)

{

if (taskDataGrid.SelectedItems.Count > 0)

{

DataRowView row = (DataRowView)taskDataGrid.SelectedItems[0];

int taskID = (int)row["ID\_Task"];

string query = "DELETE FROM Task WHERE ID\_Task = @TaskID";

using (SqlConnection connection = new SqlConnection(connectionString))

{

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@TaskID", taskID);

try

{

connection.Open();

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Задача успешно удалена.");

LoadUserData();

}

else

{

MessageBox.Show("Произошла ошибка при удалении задачи.");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении задачи: {ex.Message}");

}

}

}

}

else

{

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

}

}

}

}

WPF

<Page x:Class="TimeManagement.LoginPage"

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:TimeManagement"

mc:Ignorable="d"

d:DesignHeight="200" d:DesignWidth="350"

Title="LoginPage">

<Grid>

<StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">

<Label Content="Логин" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="15"/>

<TextBox Name="txtLogin" HorizontalAlignment="Center" VerticalAlignment="Center" Width="150" Height="25" FontSize="17"/>

<Label Content="Пароль" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="15"/>

<PasswordBox Name="txtPassword" HorizontalAlignment="Center" VerticalAlignment="Center" Width="150" Height="25" FontSize="17"/>

<Button Content="Вход" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,10,0,0" Width="80" Height="25" Name="loginButton" Click="LoginButtonClick"/>

</StackPanel>

</Grid>

</Page>

C#

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using TimeManagement.User;

namespace TimeManagement

{

/// <summary>

/// Логика взаимодействия для LoginPage.xaml

/// </summary>

public partial class LoginPage : Page

{

private readonly string connectionString = DBConnect.connectDB;

public LoginPage()

{

InitializeComponent();

Loaded += Window\_Loaded;

}

private void LoginButtonClick(object sender, RoutedEventArgs e)

{

if (ValidateUser())

{

NavigateToCorrectPage();

}

else

{

MessageBox.Show("Неправильный логин или пароль. Пожалуйста, попробуйте снова.", "Ошибка аутентификации", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

txtLogin.PreviewKeyDown += new KeyEventHandler(HandleEnterKey);

txtPassword.PreviewKeyDown += new KeyEventHandler(HandleEnterKey);

}

private void HandleEnterKey(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

LoginButtonClick(sender, e);

}

}

private bool ValidateUser()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT [User].ID\_User, [User].ID\_Role FROM [User] JOIN User\_data ON [User].ID\_User\_data = User\_data.ID\_User\_data JOIN Role ON [User].ID\_Role = Role.ID\_Role WHERE Login = @UserId AND Password = @Password";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@UserId", txtLogin.Text);

command.Parameters.AddWithValue("@Password", txtPassword.Password);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

CurrentUser.ID\_User = Convert.ToInt32(reader["ID\_User"]);

CurrentUser.ID\_Role = Convert.ToInt32(reader["ID\_Role"]);

return true;

}

else

{

return false;

}

}

}

}

}

private void NavigateToCorrectPage()

{

switch (CurrentUser.ID\_Role)

{

case 1:

NavigationService?.Navigate(new AdminPage());

break;

case 2:

EmployeePage employeePage = new EmployeePage

{

UserID = CurrentUser.ID\_User

};

NavigationService?.Navigate(employeePage);

break;

default:

break;

}

}

}

}

WPF

<Window x:Class="TimeManagement.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:TimeManagement"

mc:Ignorable="d"

Title="Time Management" Height="500" Width="1100" ResizeMode="NoResize">

<Grid>

<Frame x:Name="Main" NavigationUIVisibility="Hidden"></Frame>

</Grid>

</Window>

C#

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Data.Entity;

using TimeManagement.User;

using Microsoft.EntityFrameworkCore;

using System.Data.SqlClient;

namespace TimeManagement

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

Main.NavigationService.Navigate(new LoginPage());

Width = 550;

Height = 500;

WindowStartupLocation = WindowStartupLocation.CenterScreen;

}

}

}

WPF

<Page x:Class="TimeManagement.ViewTaskWindow"

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:TimeManagement"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="785"

Title="ViewTaskWindow">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="41\*"/>

<ColumnDefinition Width="59\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Margin="11,1,229,2">Сотрудник:</TextBlock>

<ComboBox x:Name="userComboBox" Height="23" DisplayMemberPath="FullName" SelectedItem="{Binding SelectedEmployee, UpdateSourceTrigger=PropertyChanged}" Grid.Row="2" Grid.Column="0"

HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10 0 0 0" Width="200"/>

<DataGrid Grid.Row="2" x:Name="taskDataGrid" Margin="10,34,10,45" Grid.ColumnSpan="2"

ItemsSource="{Binding Tasks}" AutoGenerateColumns="False" IsReadOnly="False"

SelectedItem="{Binding SelectedTask, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" SelectionMode="Extended">

<DataGrid.Columns>

<DataGridTextColumn Header="Наименование задачи" Width="\*" Binding="{Binding Name}"/>

<DataGridTextColumn Header="Описание" Width="\*" Binding="{Binding Description}"/>

<DataGridComboBoxColumn Header="Статус" Width="\*" SelectedItemBinding="{Binding Status}" DisplayMemberPath="Name">

<DataGridComboBoxColumn.ElementStyle>

<Style TargetType="ComboBox">

<Setter Property="ItemsSource" Value="{Binding Path=Statuses, RelativeSource={RelativeSource AncestorType={x:Type Page}}}"/>

<Setter Property="IsReadOnly" Value="True"/>

</Style>

</DataGridComboBoxColumn.ElementStyle>

<DataGridComboBoxColumn.EditingElementStyle>

<Style TargetType="ComboBox">

<Setter Property="ItemsSource" Value="{Binding Path=Statuses, RelativeSource={RelativeSource AncestorType={x:Type Page}}}"/>

</Style>

</DataGridComboBoxColumn.EditingElementStyle>

</DataGridComboBoxColumn>

<DataGridTextColumn Header="Дата начала" Width="\*" Binding="{Binding Start\_date}">

<DataGridTextColumn.ElementStyle>

<Style TargetType="TextBlock">

<Setter Property="Text" Value="{Binding Start\_date}"/>

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Header="Дата окончания" Width="\*" Binding="{Binding End\_date}">

<DataGridTextColumn.ElementStyle>

<Style TargetType="TextBlock">

<Setter Property="Text" Value="{Binding End\_date}"/>

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel Grid.Row="2" Grid.Column="1" Orientation="Horizontal" HorizontalAlignment="Right" VerticalAlignment="Bottom" Margin="0,0,3,0" Width="402">

<Button Grid.Row="2" Content="Просмотр" Height="30" Width="90" Click="ButtonView\_Click" Grid.Column="1" Margin="5"/>

<Button Grid.Row="2" Content="Сохранить" Height="30" Width="90" Click="ButtonSave\_Click" Grid.Column="1" Margin="5"/>

<Button Grid.Row="2" Height="30" Width="90" Click="ButtonAdd\_Click" Grid.Column="1" Margin="5" Content="Добавить"/>

<Button Grid.Row="2" Content="Назад" Height="30" Width="90" Click="ButtonReturn\_Click" Grid.Column="1" Margin="5"/>

</StackPanel>

</Grid>

</Page>

C#

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Globalization;

namespace TimeManagement

{

/// <summary>

/// Логика взаимодействия для ViewTaskWindow.xaml

/// </summary>

public partial class ViewTaskWindow : Page

{

private readonly string connectionString = DBConnect.connectDB;

public ObservableCollection<StatusViewModel> Statuses { get; set; }

public ObservableCollection<TaskViewModel> Tasks { get; set; }

public ViewTaskWindow()

{

InitializeComponent();

Statuses = new ObservableCollection<StatusViewModel>();

Tasks = new ObservableCollection<TaskViewModel>();

FillComboBoxWithStatuses();

FillComboBoxWithEmployees();

Loaded += ViewTaskWindow\_Loaded;

this.KeyUp += new KeyEventHandler(ViewTaskWindow\_KeyUp);

}

private void ViewTaskWindow\_Loaded(object sender, RoutedEventArgs e)

{

Window mainWindow = Application.Current.MainWindow;

if (mainWindow != null)

{

mainWindow.Width = 821;

mainWindow.Height = 450;

double screenWidth = SystemParameters.PrimaryScreenWidth;

double screenHeight = SystemParameters.PrimaryScreenHeight;

mainWindow.Left = (screenWidth - mainWindow.Width) / 2;

mainWindow.Top = (screenHeight - mainWindow.Height) / 2;

}

}

public class StatusViewModel

{

public int ID { get; set; }

public string Name { get; set; }

}

public class TaskViewModel

{

public int ID { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public DateTime Start\_date { get; set; }

public DateTime End\_date { get; set; }

public StatusViewModel Status { get; set; }

}

private void FillComboBoxWithEmployees()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT CONCAT(Surname, ' ', Name, ' ', Patronymic) AS FullName FROM [User] WHERE ID\_Role = 2";

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dt = new DataTable();

dataAdapter.Fill(dt);

if (dt.Rows.Count > 0)

{

userComboBox.ItemsSource = dt.DefaultView;

}

else

{

MessageBox.Show("Нет сотрудников с заданным ID\_Role");

}

}

}

private void FillComboBoxWithStatuses()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT ID\_Statuses, Name FROM Statuses";

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dt = new DataTable();

dataAdapter.Fill(dt);

if (dt.Rows.Count > 0)

{

foreach (DataRow row in dt.Rows)

{

StatusViewModel status = new StatusViewModel

{

ID = (int)row["ID\_Statuses"],

Name = row["Name"].ToString()

};

Statuses.Add(status);

}

}

}

}

private void ButtonView\_Click(object sender, RoutedEventArgs e)

{

if (userComboBox.SelectedItem != null)

{

Tasks.Clear();

DataRowView dataRow = (DataRowView)userComboBox.SelectedItem;

string selectedEmployeeFullName = dataRow["FullName"].ToString();

string queryTasks = "SELECT Task.ID\_Task, Task.Name, Task.Description, Task.Start\_date, Task.End\_date, Task.ID\_Statuses FROM Task INNER JOIN [User] ON Task.ID\_User = [User].ID\_User WHERE CONCAT([User].Surname, ' ', [User].Name, ' ', [User].Patronymic) = @EmployeeFullName";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(queryTasks, connection);

command.Parameters.AddWithValue("@EmployeeFullName", selectedEmployeeFullName);

SqlDataAdapter dataAdapter = new SqlDataAdapter(command);

DataTable dtTasks = new DataTable();

dataAdapter.Fill(dtTasks);

if (dtTasks.Rows.Count > 0)

{

foreach (DataRow row in dtTasks.Rows)

{

TaskViewModel task = new TaskViewModel

{

ID = Convert.ToInt32(row["ID\_Task"]),

Name = row["Name"].ToString(),

Description = row["Description"].ToString(),

Start\_date = Convert.ToDateTime(row["Start\_date"]),

End\_date = Convert.ToDateTime(row["End\_date"]),

Status = Statuses.FirstOrDefault(s => s.ID == Convert.ToInt32(row["ID\_Statuses"]))

};

Tasks.Add(task);

}

taskDataGrid.ItemsSource = Tasks;

}

else

{

MessageBox.Show("У выбранного сотрудника нет задач.");

}

}

}

else

{

MessageBox.Show("Пожалуйста, выберите сотрудника.");

}

}

private void ButtonSave\_Click(object sender, RoutedEventArgs e)

{

if (userComboBox.SelectedItem == null)

{

MessageBox.Show("Выберите пользователя перед сохранением изменений.");

return;

}

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlTransaction transaction = connection.BeginTransaction())

{

foreach (TaskViewModel task in Tasks)

{

string queryUpdate = "UPDATE Task SET Name = @Name, Description = @Description, Start\_date = @Start\_date, End\_date = @End\_date, ID\_Statuses = @ID\_Statuses WHERE ID\_Task = @ID\_Task";

using (SqlCommand command = new SqlCommand(queryUpdate, connection, transaction))

{

command.Parameters.AddWithValue("@Name", task.Name);

command.Parameters.AddWithValue("@Description", task.Description);

command.Parameters.AddWithValue("@Start\_date", task.Start\_date);

command.Parameters.AddWithValue("@End\_date", task.End\_date);

command.Parameters.AddWithValue("@ID\_Statuses", task.Status.ID);

command.Parameters.AddWithValue("@ID\_Task", task.ID);

command.ExecuteNonQuery();

}

}

transaction.Commit();

FillDataGrid();

MessageBox.Show("Изменения успешно сохранены в базе данных.");

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка при сохранении изменений: {ex.Message}");

}

}

private void FillDataGrid()

{

try

{

Tasks.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Task.ID\_Task, Task.Name, Task.Description, Task.Start\_date, Task.End\_date, Task.ID\_Statuses FROM Task INNER JOIN [User] ON Task.ID\_User = [User].ID\_User WHERE [User].ID\_Role = 2";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

TaskViewModel task = new TaskViewModel

{

ID = Convert.ToInt32(reader["ID\_Task"]),

Name = reader["Name"].ToString(),

Description = reader["Description"].ToString(),

Start\_date = Convert.ToDateTime(reader["Start\_date"]),

End\_date = Convert.ToDateTime(reader["End\_date"]),

Status = Statuses.FirstOrDefault(s => s.ID == Convert.ToInt32(reader["ID\_Statuses"]))

};

Tasks.Add(task);

}

taskDataGrid.ItemsSource = Tasks;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при заполнении DataGrid: {ex.Message}");

}

}

private void DeleteSelectedTask()

{

if (taskDataGrid.SelectedItem is TaskViewModel)

{

int selectedIndex = taskDataGrid.SelectedIndex;

if (taskDataGrid.Items.Count > 0 && selectedIndex >= 0 && selectedIndex < taskDataGrid.Items.Count)

{

TaskViewModel task = taskDataGrid.Items.GetItemAt(selectedIndex) as TaskViewModel;

int taskId = task.ID;

MessageBoxResult result = MessageBox.Show("Вы хотите удалить запись?", "Подтверждение удаления", MessageBoxButton.YesNo);

if (result == MessageBoxResult.Yes)

{

string queryDelete = "DELETE FROM Task WHERE ID\_Task = @TaskID";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(queryDelete, connection))

{

command.Parameters.AddWithValue("@TaskID", taskId);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Выбранная задача успешно удалена из базы данных.");

FillDataGrid();

}

else

{

MessageBox.Show("Не удалось удалить выбранную задачу из базы данных.");

}

}

}

}

else if (result == MessageBoxResult.No)

{

MessageBox.Show("Запись не была удалена.");

}

}

else

{

MessageBox.Show("Пожалуйста, выберите задачу для удаления.");

}

}

else

{

MessageBox.Show("Пожалуйста, выберите задачу для удаления.");

}

}

private void ViewTaskWindow\_KeyUp(object sender, KeyEventArgs e)

{

if (e.Key == Key.Delete)

{

DeleteSelectedTask();

}

}

private void ButtonReturn\_Click(object sender, RoutedEventArgs e)

{

if (NavigationService.CanGoBack)

{

NavigationService.GoBack();

}

}

private void ButtonAdd\_Click(object sender, RoutedEventArgs e)

{

AddTaskForEmployeeWindow addTaskForEmployeeWindow = new AddTaskForEmployeeWindow();

\_ = NavigationService.Navigate(addTaskForEmployeeWindow);

}

}

}