КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА

САНКТ-ПЕТЕРБУРГСКОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
  
«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Специальность 09.02.03 «Программирование в компьютерных системах»

УТВЕРЖДАЮ

Зам. директора по УМР

\_\_\_\_\_\_\_\_\_\_\_\_ Венедиктова О.Н.

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

ДИПЛОМНЫЙ ПРОЕКТ

Пояснительная записка

Тема: Корпоративное приложение для контроля работы и управления проектами

Группа 463

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Н. Венедиктова

Консультант по экономической части\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.В. Львова

Руководитель дипломного проекта \_\_\_\_\_\_СОГЛАСОВАНО\_\_\_\_\_ И.П. Смирнова

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Тяпаева

Санкт-Петербург, 2020

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc42547778)

[ВВЕДЕНИЕ 4](#_Toc42547779)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 6](#_Toc42547780)

[1.1 Цель разработки 6](#_Toc42547781)

[1.2 Анализ рынка подобных решений 6](#_Toc42547782)

[1.3 Анализ среды разработки 9](#_Toc42547783)

[1.4 Анализ средств программирования 11](#_Toc42547784)

[1.5 Анализ используемых технологий 12](#_Toc42547785)

[2. ПРОЕКТНАЯ ЧАСТЬ 15](#_Toc42547786)

[2.1 Анализ средств программирования 15](#_Toc42547787)

[2.2 Постановка задачи 15](#_Toc42547788)

[2.3 Назначение программы 16](#_Toc42547789)

[2.4 Требования к программе 16](#_Toc42547790)

[2.5 Требования к функциональным характеристикам 16](#_Toc42547791)

[2.6 Требования к аппаратным и программным средствам 17](#_Toc42547792)

[2.7 Требования к надёжности 17](#_Toc42547793)

[2.8 Принцип работы приложения 18](#_Toc42547794)

[2.9 Структура базы данных 21](#_Toc42547795)

[2.10 Описание алгоритма 27](#_Toc42547796)

[2.11 Программная реализация 30](#_Toc42547797)

[2.12 Тестирование и отладка 38](#_Toc42547798)

[2.13 Оценка результатов выполненных задач 41](#_Toc42547799)

[3. ЭКОНОМИЧЕСКАЯ ЧАСТЬ 42](#_Toc42547800)

[3.1 Описание продукта 42](#_Toc42547801)

[3.2 Описание и расчёт затрат на выполнение проекта 42](#_Toc42547802)

[3.2.1 Расчет трудоемкости 43](#_Toc42547803)

[3.2.2 Расчет затрат на оплату труда 44](#_Toc42547804)

[3.2.3 Расходные материалы 45](#_Toc42547805)

[3.2.4 Расчёт амортизации 45](#_Toc42547806)

[3.2.5 Расчёт электроэнергии 45](#_Toc42547807)

[3.2.6 Расчёт сметы затрат 45](#_Toc42547808)

[3.3 Расчет экономического эффекта 46](#_Toc42547809)

[ЗАКЛЮЧЕНИЕ 47](#_Toc42547810)

[ЛИТЕРАТУРА 48](#_Toc42547811)

[СПИСОК ИСТОЧНИКОВ 48](#_Toc42547812)

[ПРИЛОЖЕНИЕ А – Серверная часть 49](#_Toc42547813)

[ПРИЛОЖЕНИЕ Б – Клиентская часть 62](#_Toc42547814)

[ПРИЛОЖЕНИЕ В – Структура базы данных 71](#_Toc42547815)

[ПРИЛОЖЕНИЕ Г – Блок схема работы приложения 72](#_Toc42547816)

ВВЕДЕНИЕ

Тема данного дипломного проекта: Корпоративное приложение для контроля работы и управления проектами. Эта тема затрагивает менеджмент времени. Для эффективной работы нужно организовывать своё время. Это касается как обычных людей, так и организации, и компании. Правильная расстановка задач помогает выполнить работу к сроку.

Приложение, разработанное для дипломного проекта, имеет в себе следующий функционал:

* возможность зарегистрироваться как обычный пользователь, либо зарегистрировать свою компанию;
* в задачнике пользователь может увидеть созданные им и назначенные ему задачи, добавить новые, следить за сроками, удалять лишние;
* можно объединить задачи в проект, если они направлены на достижение одной цели;
* компании могут создавать и удалять департаменты, сортировать по ним сотрудников, относящихся к ним. При выборе департамента из списка пользователь увидит принадлежащие ему проекты и информацию об успеваемости департамента;
* для того, чтобы посторонние пользователи не объявляли себя сотрудниками компании, к которой они не относятся, компания может сгенерировать специальный код и отправить нужному пользователю. При его активации сотрудник добавляется автоматически, вручную необходимо задать только его должность;
* компании так же могут составлять график. При входе в панель пользователя сотрудник видит свои рабочие дни на эту неделю и рабочее время на каждый конкретный день;
* при назначении исполнителя в задаче, сотрудник будет оповещён сообщением автоматически;
* пользователи смогут написать разработчикам о том, какие функции они хотели бы увидеть в приложении.

Таким образом, данный дипломный проект является менеджером задач.

Данный диплом подтверждает освоение следующих компетенции профессиональных модулей:

ПМ 01. Разработка программных модулей программного обеспечения для компьютерных систем:

* ПК 1.1 – Выполнять разработку спецификаций отдельных компонент;
* ПК 1.2 – Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля;
* ПК 1.3 – Выполнять отладку программных модулей с использованием специализированных программных средств;
* ПК 1.4 – Выполнять тестирование программных модулей;
* ПК 1.6 – Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.

ПМ 02. Разработка и администрирование баз данных:

* ПК 2.1 – Разрабатывать объекты базы данных;
* ПК 2.2 – Реализовывать базу данных в конкретной системе управления базами данных (СУБД);
* ПК 2.3 – решать вопросы администрирования базы данных;
* ПК 2.4 – Реализовывать методы и технологии защиты информации в базах данных.

ПМ 03. Участие в интеграции программных модулей:

* ПК 3.1 – Анализировать проектную и техническую документацию на уровне взаимодействия компонент программного обеспечения;
* ПК 3.3 – Выполнять отладку программного продукта с использованием специализированных программных средств.

# АНАЛИТИЧЕСКАЯ ЧАСТЬ

В этом разделе описаны цели дипломного проекта, обоснование выбора среды разработки, выбранные для реализации данного проекта, проведён анализ альтернатив.

## **Цель разработки**

Целью разработки дипломного проекта является desktop-приложение для упрощения работы с менеджментом проектов и задач, предназначенное как для отдельных организаций, так и для обычных пользователей, которые хотят упорядочить свои дела.

## **Анализ рынка подобных решений**

Данный проект является Task-менеджером, на данный момент существуют следующие аналоги:

* Asana — онлайн-сервис для управления проектами. Подходит как для командной работы, так и для личного использования. Плюсом данной платформы является мобильная версия сервиса. Имеет разделение на управляющих и исполнителей. Его минусом является отсутствие локализации. На рисунке 1 представлена главная страница сервиса.

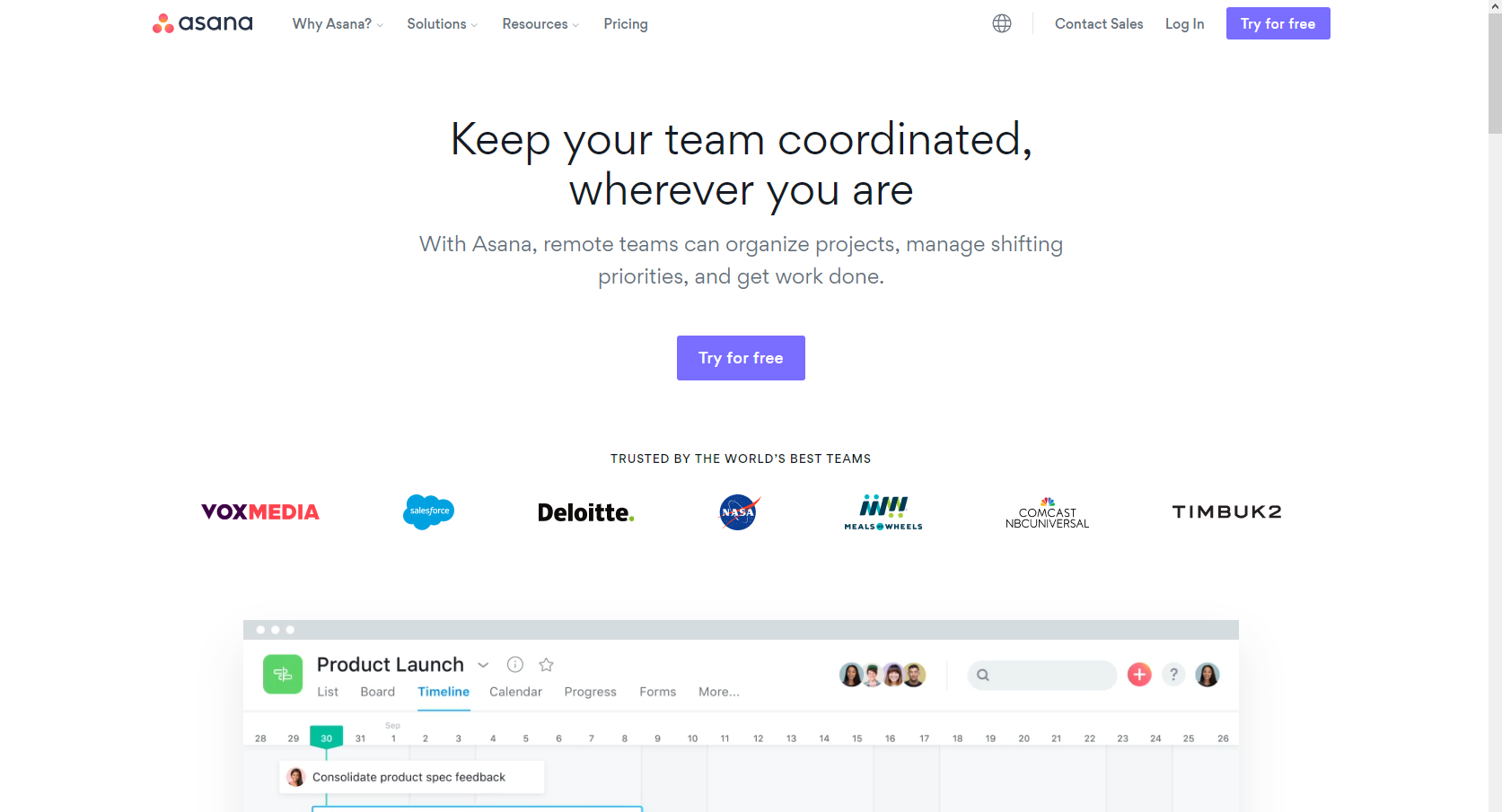


Рисунок 1 – Главная страница сервиса «Asana»

* Flow — приложение в виде канбан-доски для управления проектами и задачами по методологии Agile. Имеет версии для компьютеров, планшетов и смартфонов. Позволяет контролировать процесс распределения и выполнения задач. Плюсом приложения является отчёты, Agile доска, уведомления, возможность экспорта данных. Минусом является высокая цена при малом количестве функций. На рисунке 2 отображена главная страница сервиса.

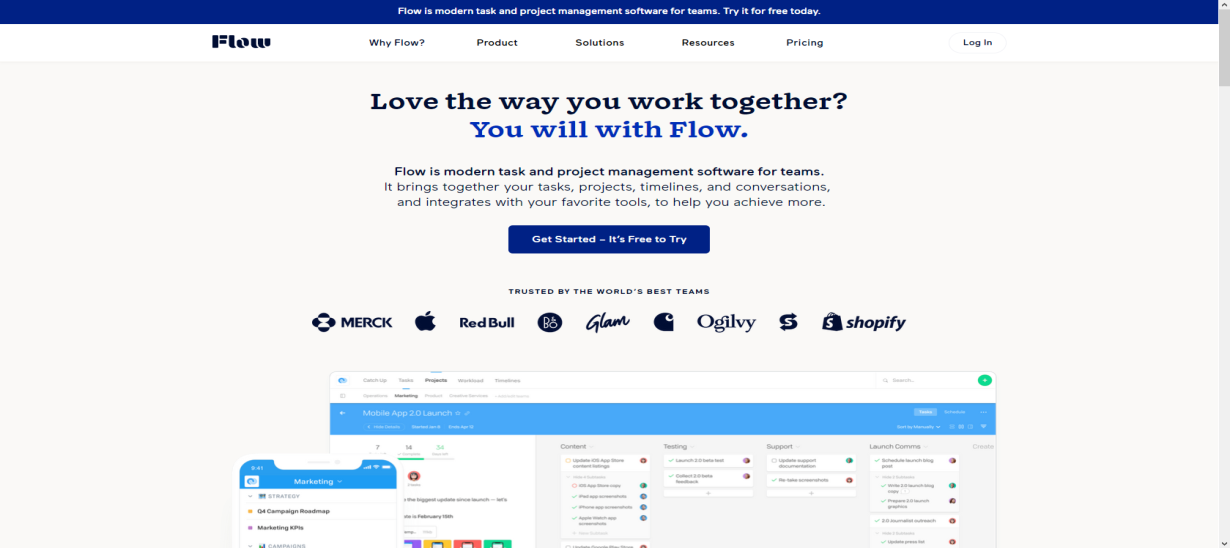


Рисунок 2 – Главная страница сервиса «Flow»

* Trello — приложение подходит для небольших команд, имеет минималистический стиль и большой функционал. Так же находится на web-платформе и имеет мобильную версию. Из плюсов имеет возможность интеграции с другими сервисами, автоматическое резервное копирование, фильтрация с помощью цветных меток. Минусом является отсутствие кастомизации. На рисунке 3 – главная страница сервиса.

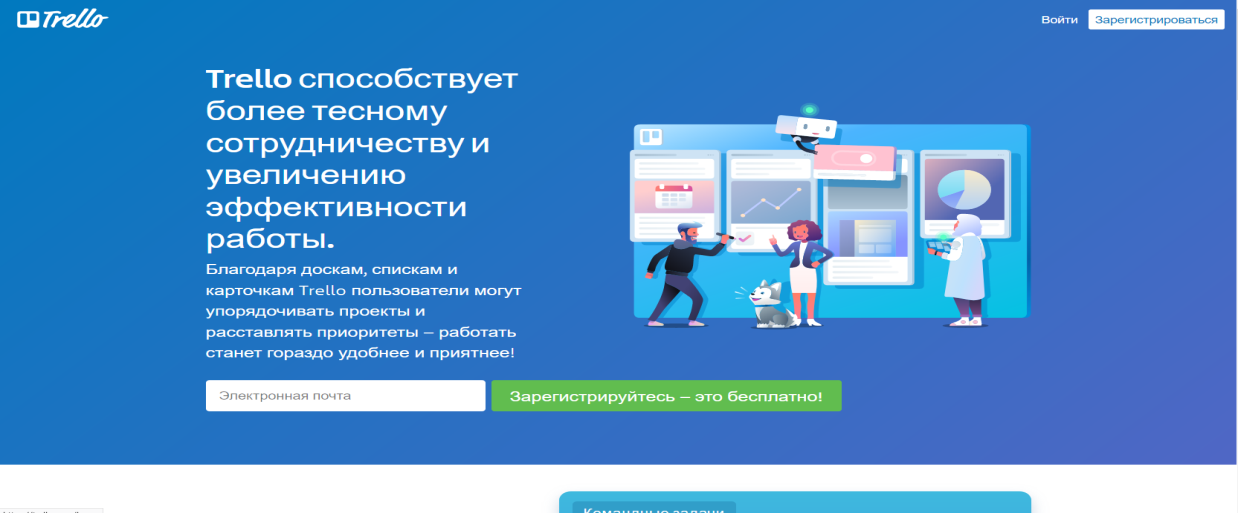


Рисунок 3 – Главная страница «Trello»

* Wrike — один из самых популярных Task-менеджеров. Облачный сервис, имеющий интеграцию с такими сервисами как Google, GitHub и Microsoft. Так же имеет мобильное приложение. Позволяет создавать отчёты для оценки эффективности работы сотрудников. Плюсами является двухэтапная верификация, обеспечивающая безопасность данных, шаблоны, учёт времени и бюджета. Из минусов – неудобная система комментирования. На рисунке 4 представлена главная страница сервиса.

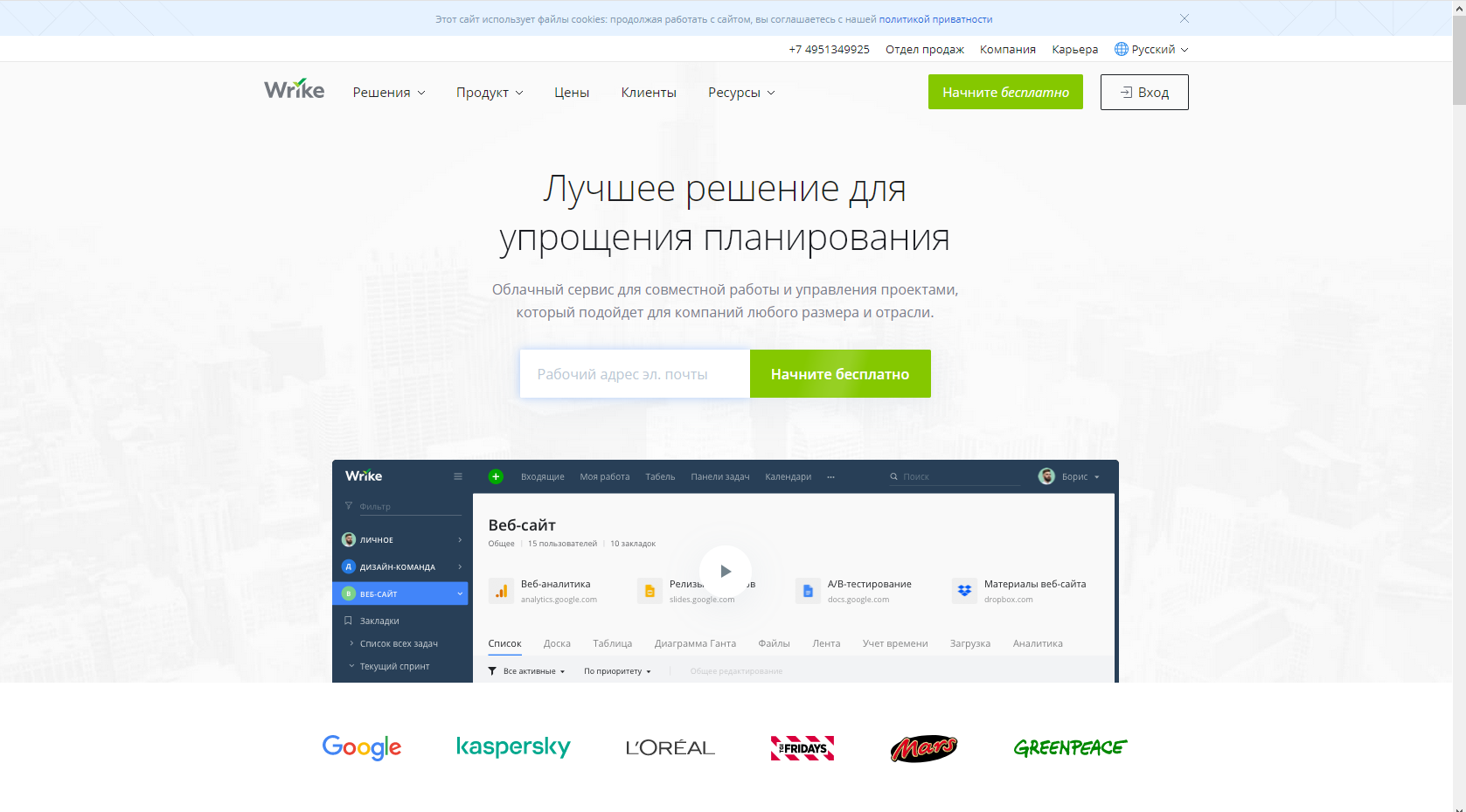


Рисунок 4 – Главная страница сервиса «Wrike»

Проанализировав аналоги, можно прийти к выводу, что данное приложение покрывает следующие минусы:

* более удобный и понятный интерфейс для простых пользователей;
* локализация, которая со временем будет пополняться;
* возможность получить желаемые функции, написав разработчикам и не платить лишние деньги;
* при обновлении приложения цена продукта останется такая, какая была изначально.

## **Анализ среды разработки**

Перед разработкой стояла задача проанализировать варианты средств разработки для реализации данного проекта, которые бы предоставляли весь необходимый функционал.

1. Sublime Text – является полнофункциональным текстовым редактором. Имеет различные для редактирования кода, которые помогают отследить изменения. Поддерживает такие функции как подсветка синтаксиса, авто отступ, распознание типов файлов, боковая панель с папкой и файлами, доступна загрузка различных плагинов и пакетов. Данный продукт совмести с операционными системами, такими как Windows, Linux и MacOS. Из плюсов есть такие возможности как настройка горячих клавиш, работа с 30-ю языками программирования и бесплатное использование. Из минусов – отсутствие встроенного компилятора, который мог бы упростить и ускорить работу. На рисунке 5 изображён интерфейс программы.

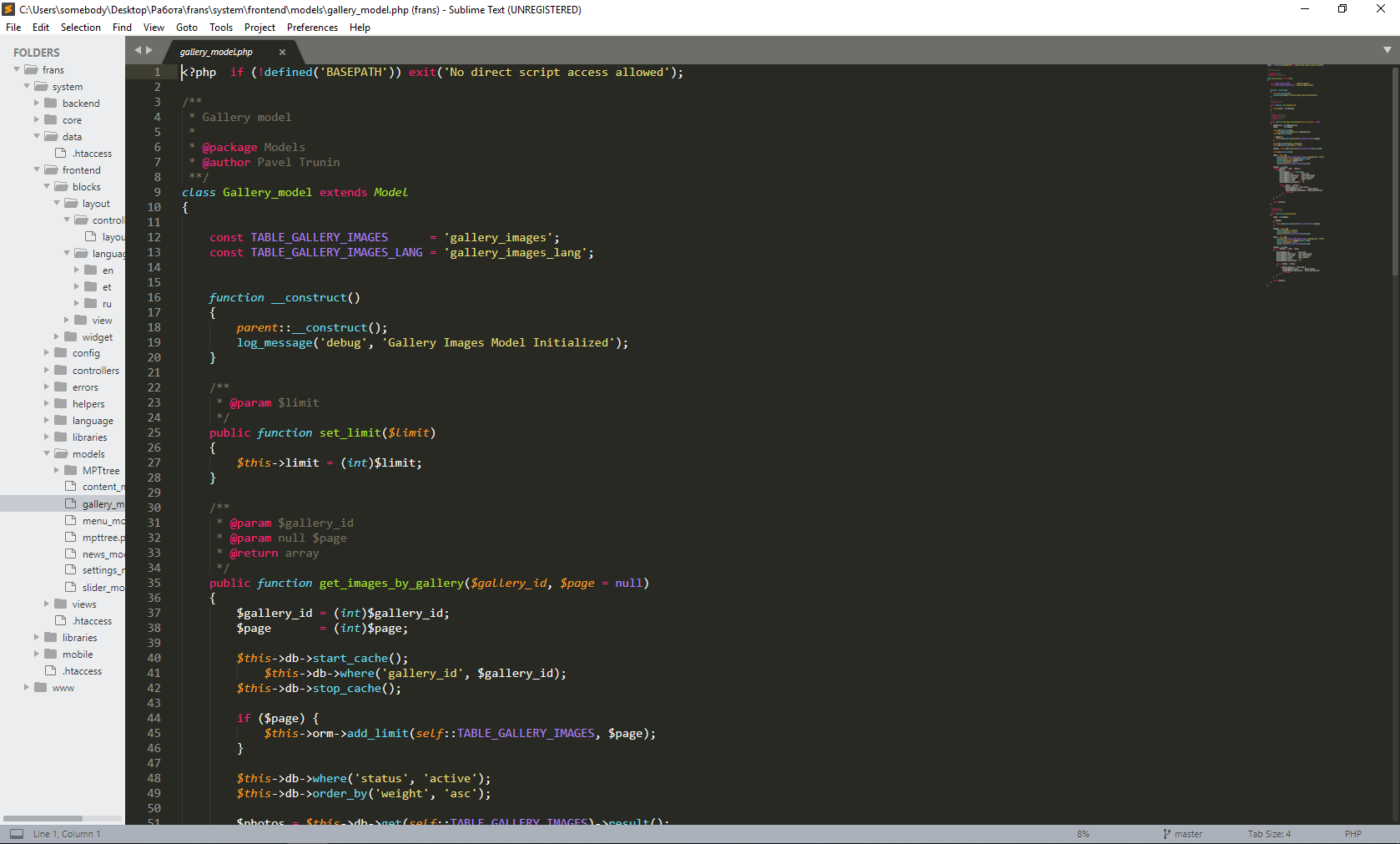


Рисунок 5 – Интерфейс программы «Sublime Text»

1. Visual Studio Code – это кроссплатформенный редактор кода, вмещающий в себя функции, которые поддерживают базовые интегрированной среды разработки. Позволяет создавать интерфейсы приложений, так же с Windows Forms, web-приложения. Плюсом является присутствие в редакторе встроенного отладчика, поддержка большого количества языков программирования. Минус этой среды является отсутствие полноценного компилятора, который необходим в работе над проектом. На рисунке 6 отображён интерфейс данной программы.

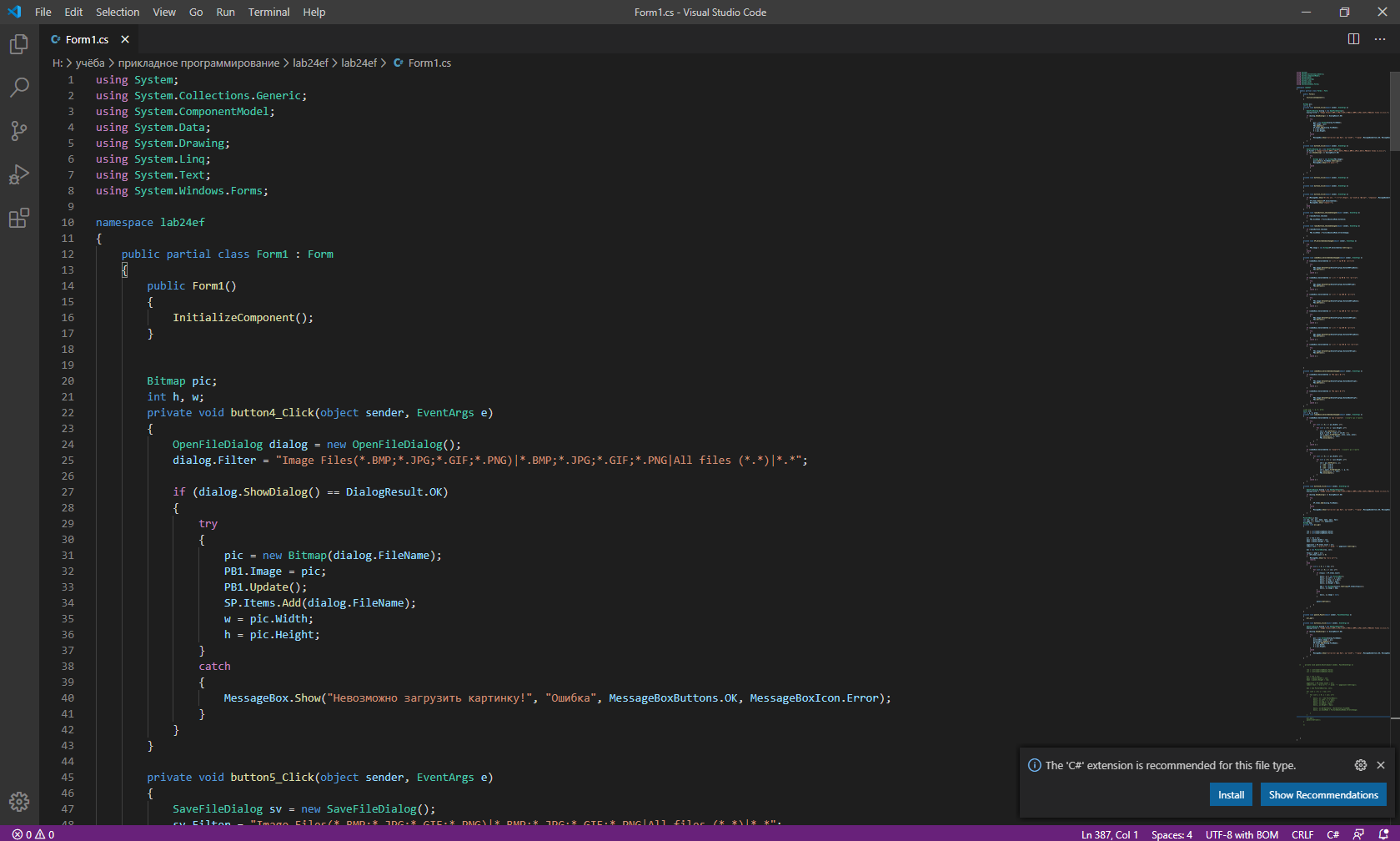


Рисунок 6 – Интерфейс программы «Visual Studio Code»

1. Visual Studio 2019 – полнофункциональная среда разработки приложений для платформ Windows (консольных и графических). Некоторые фреймворки позволяют создать кроссплатформенные приложение для таких ОС как Linux. Данная средство разработки включает в себя отладчик кода, редактор форм, для простого конструирования графических интерфейсов, веб-редактор, дизайнер классов, дизайнер схем баз данных. Данное приложение позволяет подключать дополнительные пакеты для расширения функционала, в котором нуждается разработчик, что является большим плюсом. Минусом данной системы являются системные требования. Для работы с такой средой слабые компьютеры и ноутбуки не подойдут. На рисунке 7 изображён интерфейс данной среды разработки.

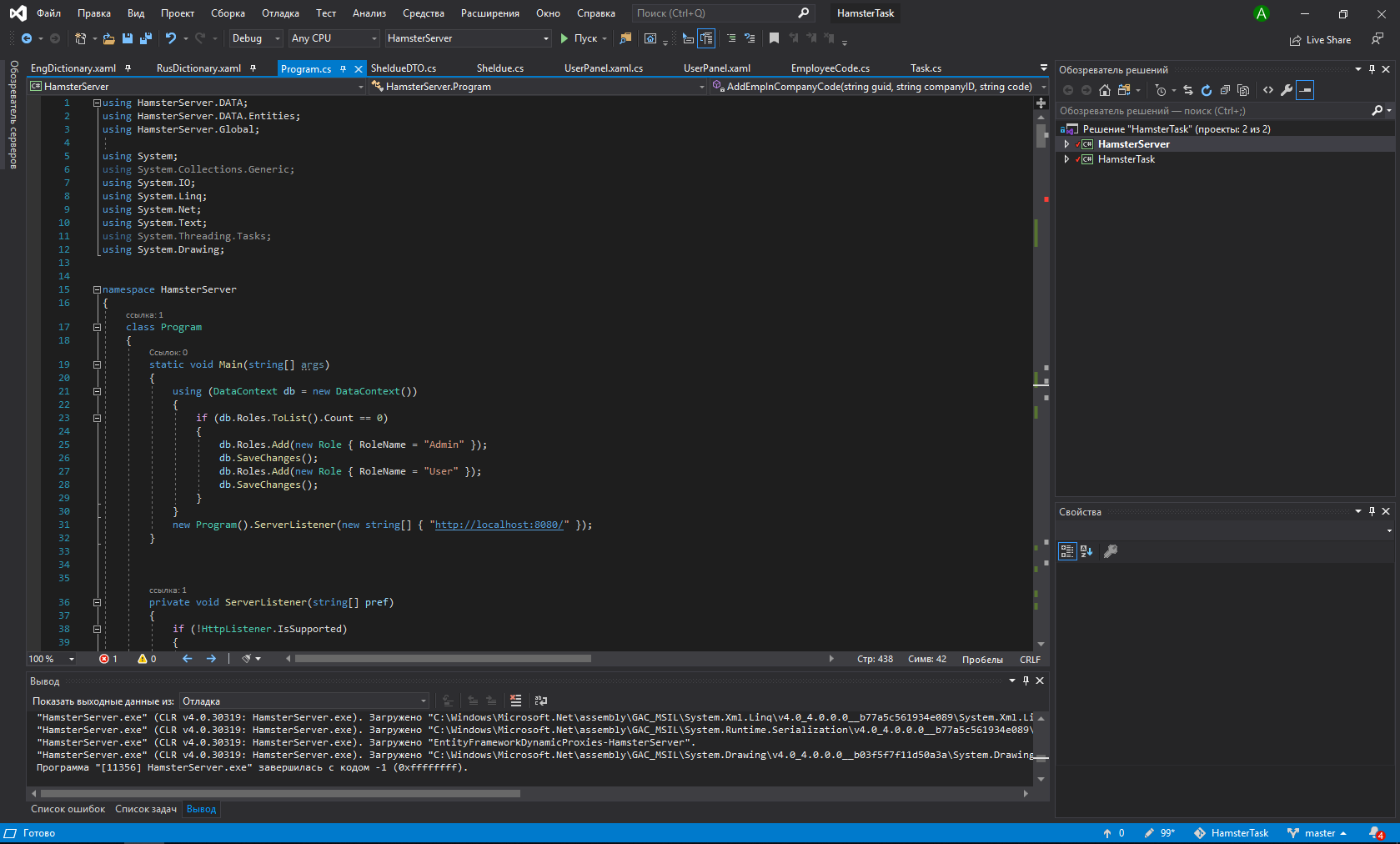


Рисунок 7 – Интерфейс программы «Visual Studio 2019»

* 1. Анализ средств программирования

Для разработки данного приложения были выбраны следующие средства разработки:

1. C# – объектно-ориентированный язык программирования. Используется в работе серверной части и взаимодействии с клиентской. На сегодняшний день C# является одним из самых мощных, быстроразвивающимся и достаточно востребованных в IT-сфере языков программирования. Он позволяет решить задачи по разработке крупных и гибких приложений, масштабированных и расширяемых проектов. Так же имеет функционал как динамические связывания и асинхронные методы. Код на этом языке компилируется в файлы с расширением .exe или .dll. Плюсом является то, что при разработке компилируется лишь та часть кода, к которой обращается пользователь, что повышает производительность и не несёт лишней нагрузки на систему, синтаксис данного языка богат, прост и удобен в изучении, так же язык является кроссплатформенным, расширяемость системы, скорость работы, защищённость и контроль версий подключаемых алгоритмов. Минусом является то, что данный язык легко дизассемблируется, приоритет ориентирован на Windows. Так же для крупных компаний язык не бесплатен.
2. XAML – расширенный язык разметки для приложений, который был основан на XML. Плюсами являются такие возможности, как сохранение прописанных пропорций элементов, за небольшой промежуток времени можно сделать аккуратный минималистический дизайн, можно так же использовать шаблоны, что ускоряет работу над проектом, можно использовать привязку одних данных для нескольких элементов. Минусами является то, что каждому элементу нужно прописывать свой стиль, то есть, чтобы кнопки выглядели одинаково, нужно в каждой прописывать один и тот же стиль.
3. LINQ – это простой и удобный язык запросов к источнику данных. Он позволяет существенно упростить взаимодействие с базой данных из программного кода. Из плюсов можно перечислить отсутствие в коде программы запросов на языке SQL, более компактный вид запросов, возможность согласованного запроса из разных источников. Минусами являются такие явления, как меньший функционал по сравнению с запросами на SQL, отсутствие некоторых операторов, а также в некоторых случаях код LINQ работает медленнее.
   1. Анализ используемых технологий

В процессе разработки данного проекта были использованы следующие технологии:

1. .Net Framework – мощная платформа для создания приложений:

* эта технология поддерживает несколько языков, которые компилируются в CIL – Common Intermediate Language («Общий Промежуточный Язык») виртуальной машины .NET., благодаря чему имеется возможность реализовывать отдельные модули на разных языках;
* так же технология кроссплатформенна, благодаря чему можно создать приложение, поддерживающееся на платформах Windows, Linux и MacOS;
* имеет единую для всех языков библиотеку классов, то есть любая разработка (веб-сайт, текстовый редактор, чат и т.п) будет использовать общую библиотеку классов .NET;
* к данной технологии можно подключить другие, для которых среда исполнения CLR и общая библиотека классов являются основой: например, ADO.NET или Entity Framework, которые предназначены для работы с базой данных. Для графических приложений подключаются такие технологии, как WPF и UWP;
* данная технология автоматически очищает память, так как имеет встроенный сборщик мусора.

1. Entity Framework – специализированная объектно-ориентированная технология на базе .Net, которая позволяет работать с базой данных, которая позволяет работать с данными независимо от типа хранилища.

Концепцией технологии являются «сущности», которые представляют из себя набор данных. То есть работа идёт не с таблицами, а с объектами, которые имеют свои ключи и связи.

Для работы с данными используется запросы LINQ, с помощью которым можно получать не только данные из базы данных, но и объекты, связанные с другими объектами.

Данная технология имеет три способа взаимодействия с базой:

* Code first – программируется класс модели данных, которые технология будет генерировать в базу данных;
* Database first – Entity Framework отражает модель определённой базы данных, которая уже существует;
* Model first – с помощью моделей, технология создаёт базу данных.

Плюсами технологии являются:

* повышение производительности;
* не нужно постоянно переключаться на СУБД, всё можно сделать через код;
* понятные и лёгкие запросы.

Минусы:

* обычно при первом запуске запроса, он сохраняется в кэш, что и повышает производительность, но бывают некоторые виды запросов, требующие перекомпиляции при обращении;
* чтобы получать все связанные с классом/моделью данные на каждую связь нужно прописывать include.

1. WPF – технология, представляющая собой подсистему для работы с графикой приложения. При отрисовки графики элементов технологии в отличии от Windows Forms, за который отвечают части Windows (напр. GDI+), у WPF за это отвечает DirectX. То есть отрисовка элементов ложиться на графический процессор, обеспечивающий аппаратное ускорение графики.

Преимущества данной технологии:

* позволяет использовать C# для построения логики проекта;
* позволяет создать разнообразный интерфейс с помощью языка разметки XAML;
* позволяет создать 3D-модели, а также анимацию;
* обеспечивает привязку данных к элементам, позволяет создать свои стили и локальные шаблоны;
* возможность создавать новые элементы;
* обеспечивает высокую производительность независимо от типа графики.

Минусы:

* потребление памяти в процессе работы приложения этой технологии несколько выше;
* чувствительность к драйверам видеокарты.

# ПРОЕКТНАЯ ЧАСТЬ

В проектной части дипломного проекта изложены сведения о выполненном программном продукте, предъявленные требования к приложению, техническим средствам, структуре и логике построения кода. Так же эта часть содержит в себе подробную инструкцию приложения для быстрого знакомства с продуктом и легкого использования в дальнейшем.

* 1. Анализ средств программирования

Перед началом работы над проектом, необходимо составить техническое задание, в котором должны быть чётко изложены требования к будущему продукту.

Техническое задание – это список требований заказчика к объекту закупки, в котором определены все условия и порядки проведения работ.

* 1. Постановка задачи

Основной задачей дипломного проекта, является разработка настольного приложения для менеджмента проектами, задачами и временем под название «HamsterTask».

Приложение должно иметь следующий функционал:

* регистрация пользователя;
* регистрация компании;
* авторизация;
* возможность редактировать данные профиля;
* создание задач и проектов;
* добавление задач в проекты;
* отправка сообщений пользователю;
* удаление и редактирование задач и проектов;
* создание департаментов организации, так же редактирование и удаление;
* добавление сотрудника в компанию;
* возможность распределять сотрудников по департаментам;
* назначение исполнителей задач;
* оповещение о появившейся задаче;
* удаление и изменение информации о сотруднике (изменение должности, перенаправление в другой департамент);
* просмотр эффективности работы по всей компании или отдельному департаменту;
* просмотр рейтинга самых эффективных работников по департаментам;
* загрузка логотипа для компании и аватара для пользователя;
* реализация возможность составления графика работы;
* редактирование информации о компании;
* возможность генерации кода для сотрудников и его активация;
* возможность разграничение прав в компании.
  1. Назначение программы

Данное приложение предназначено для пользователей, которым необходимо правильно организовать своё время и дела; для фрилансеров, которым нужна помощь в распределении работ над проектами; для компаний, которым нужно иметь простое приложение, обеспечивающее распределение задач и проектов между отделами и сотрудниками. Так же необходимо, чтобы данное приложение имело низкие системные требования.

* 1. Требования к программе

Проект должен обеспечить нормальное функционирование при, насколько это возможно, минимальных программно-аппаратных и системных требованиях. Таким образом, программой смогут воспользоваться люди, не имеющие мощных компьютеров или использующие параллельно с ней слишком требовательные программы. Программный продукт должен иметь повышенную производительность и минимальное влияние на сторонние программы. Следующие требования дадут возможность сэкономить на аппаратных средствах компании.

* 1. Требования к функциональным характеристикам

Программа должна успешно выполнять следующие функции:

* успешная регистрация и авторизация;
* возможность просмотра задач и проектов;
* возможность добавления и удаления задач и проектов;
* возможность отправить обратную связь администраторам, в случае пожеланий дополнения функционала программы;
* возможность создания департаментов и добавления сотрудников;
* просмотр информации и сводок о департаментах и компании;
* возможность редактирования профиля и информации о компании.
  1. Требования к аппаратным и программным средствам

Минимальные системные требования:

* операционная система: линейка Windows начиная с Windows 7;
* процессор: Intel Core i3-2100 и более поздние;
* объём свободного места на диске: 100 Мб;
* оперативная память: от 1 Гб.
  1. Требования к надёжности

При регистрации пароль пользователя кодируются с добавлением созданной специально для пользователя «соли». Так же каждый пользователь должен придумать уникальный логин.

При входе в системы программа выполняет следующие этапы:

1. Ищет пользователя по уникальному логину.
2. При удачном исполнении первого пункта из базы берётся соль пользователя и прогоняется вместе с введённым паролем через кодирующую функцию.
3. При удачном сравнении хранящихся строк пароля в базе и полученных из функции для пользователя генерируется уникальный GUID.
4. Сервер передаёт клиенту его GUID, с помощью которого можно будет запрашивать остальные данные, которые нужны пользователю.

Логин должен содержать от 8 до 12 символов. Пароль должен быть в размере от 8 до 16 символов. Так же пользователь не может ввести такие символы как «/» и «|», но должен помимо хотя бы одной заглавной и одной строчной латинской буквы указать цифры и один из дополнительных символов («!», «#», «$», «%»). При некорректном вводе данных программа попросит пользователя ввести данные снова.

* 1. Принцип работы приложения

В данном разделе описаны различные схемы, отображающие алгоритмы работы программы, которые показывают принцип взаимодействия пользователя с данной системой.

1. Общий принцип работы приложения.

Общий принцип работы приложения изображен на рисунке 8.

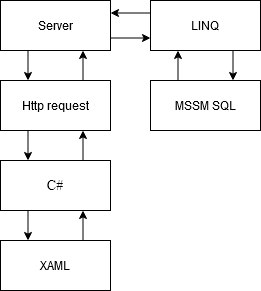


Рисунок 8 – Схема принципа работы приложения

2) Функциональная диаграмма IDEF0.

Для того, чтобы отразить в схеме функциональную структуру программы, необходимо обратиться к приёмам функционального моделирования. Для этого используется графическая нотация IDEF0.

Правила составления IDEF диаграмм:

* входящие стрелки – определённые задачи;
* исходящие – получаемый результат;
* сверху-вниз – механизмы управления;
* снизу-вверх – механизмы для воспроизведения нужной работы.

Для описания функционала программы используются контекстные диаграммы и диаграммы декомпозиции первого уровня.

Контекстная диаграмма для программы, разработанной для этого дипломного проекта, изображена на рисунке 9.

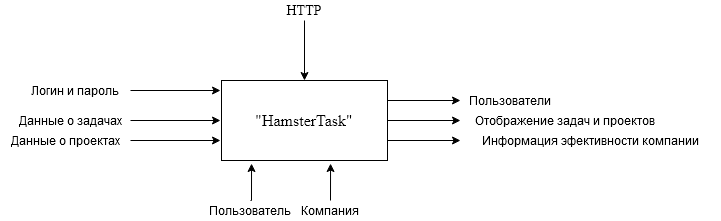


Рисунок 9 – Контекстная диаграмма приложения

На рисунке ниже представлена диаграмма декомпозиции первого уровня, на которой можно увидеть внутренние процессы программы (рисунок 10).

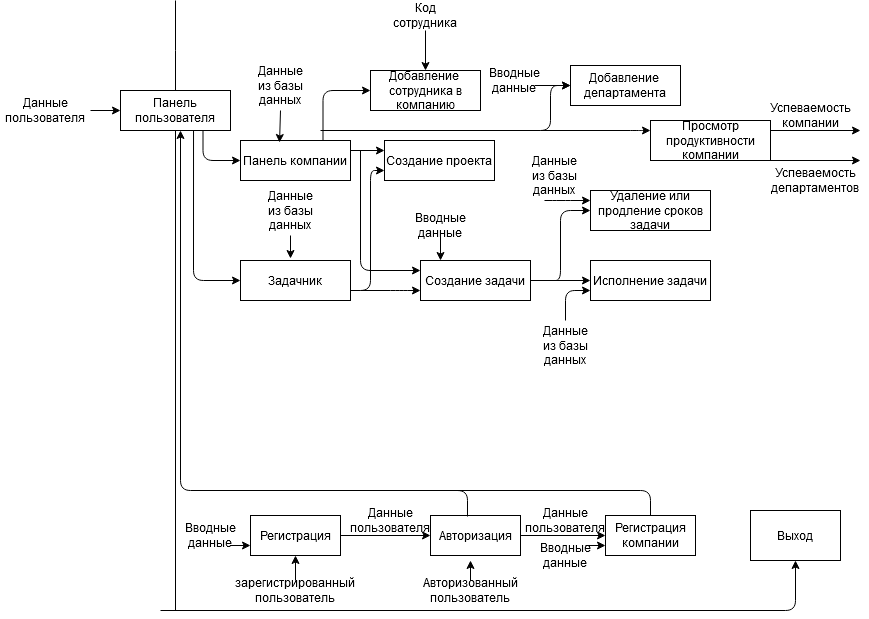


Рисунок 10 – Диаграмма декомпозиции первого уровня

3) Диаграмма вариантов использования

UML – Диаграмма использования является сценарием поведения в виде исходного концептуального исполнения в процессе проектирования и разработки.

Частью данной диаграммы являются такие элементы как:

* акторы – лицо, группа лиц или систем, которые взаимодействуют с приложением;
* варианты использования – сервисы, предоставляемые акторам;
* комментарии;
* отношения между элементами.

Описание вариантов использования являются пояснением, которые описывают процесс и активность. Основной же целью данной диаграммы является описание функциональности и поведения, позволяющее клиенту и программисту обсудить проектируемую систему.

На рисунке 11 изображена диаграмма вариантов использования.

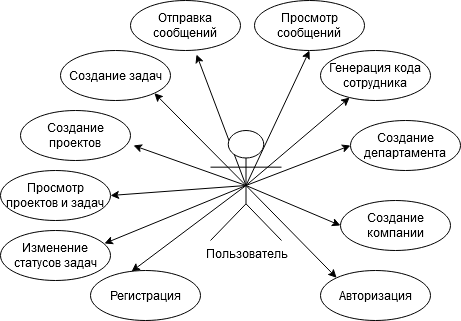


Рисунок 11 – UML диаграмма вариантов использования

* 1. Структура базы данных

Для хранения данных о пользователях, задачах, проектов и компаниях, с помощью Entity Framework Code First была создана база данных из 18 таблиц (рисунки 12). Структура базы представлена на рисунке 13 и в приложении В.

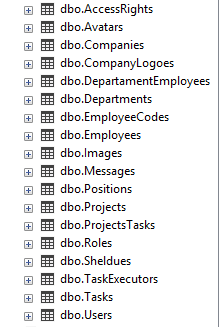


Рисунок 12 – Таблицы базы данных «HamsterBase»



Рисунок 13 – Структура базы данных «HamsterBase»

1. Таблица «Users»

Назначение: Хранение информации о пользователе (рисунок 14).

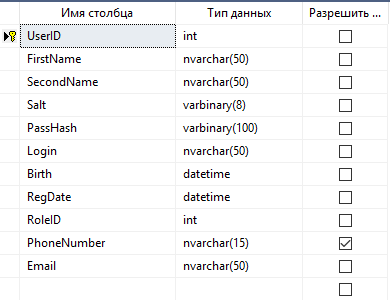


Рисунок 14 – Структура таблицы «User»

1. Таблица «Tasks»

Назначение: хранение информации о созданных задачах (рисунок 15).

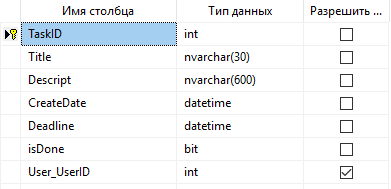


Рисунок 15 – Структура таблицы «Tasks»

1. Таблица «TaskExecutors»

Назначение: хранение информации об закрепленных как исполнителей определённых задач пользователей (рисунок 16).

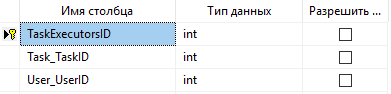


Рисунок 16 – Структура таблицы «TaskExecutors»

1. Таблица «Scheldues»

Назначение: хранение информации о рабочем дне сотрудника (рисунок 17).

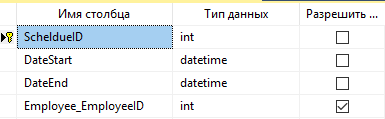


Рисунок 17 – Структура таблицы «Scheldue»

1. Таблица «Roles»

Назначение: хранение информации о ролях (рисунок 18).

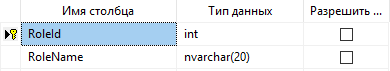


Рисунок 18 – Структура таблицы «Role»

1. Таблица «ProjectsTasks»

Назначение: хранение информации о задачах, относящихся к определённым проектам (рисунок 19).

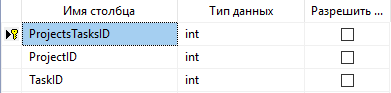


Рисунок 19 – Структура таблицы «ProjectsTasks»

1. Таблица «Projects»

Назначение: хранение информации о проектах (рисунок 20).

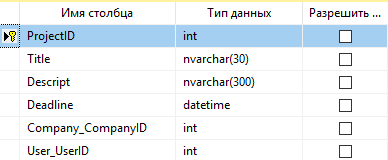


Рисунок 20 – Структура таблицы «Projects»

1. Таблица «Positions»

Назначение: хранение информации о должностях (рисунок 21).

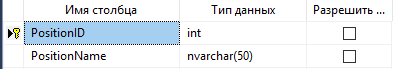


Рисунок 21 – Структура таблицы «Positions»

1. Таблица «Messages»

Назначение: хранение информации о сообщениях (рисунок 22).

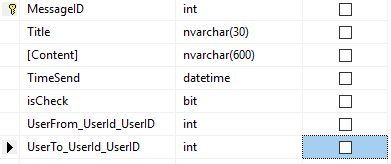


Рисунок 22 – Структура таблицы «Messages»

1. Таблица «Images»

Назначение: хранение информации о изображениях (рисунок 23).

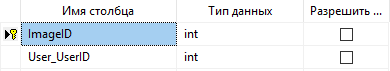


Рисунок 23 – Структура таблицы «Images»

1. Таблица «Employees»

Назначение: хранение информации о сотрудниках (рисунок 24).

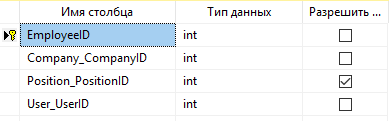


Рисунок 24 – Структура таблицы «Employees»

1. Таблица «EmployeeCodes»

Назначение: хранение информации о кодах сотрудников (рисунок 25).

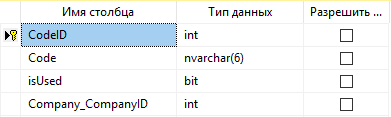


Рисунок 25 – Структура таблицы «EmployeeCodes»

1. Таблица «Departaments»

Назначение: хранение информации об отделах/департаментов компаний (рисунок 26).

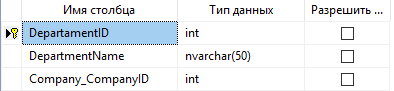


Рисунок 26 – Структура таблицы «Departaments»

1. Таблица «DepartamentEmployees»

Назначение: хранение информации о сотрудниках, относящихся к департаментам (рисунок 27).

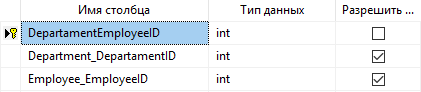


Рисунок 27 – Структура таблицы «DepartamentEmployees»

1. Таблица «CompanyLogos»

Назначение: хранение информации о логотипах компаний (рисунок 28).

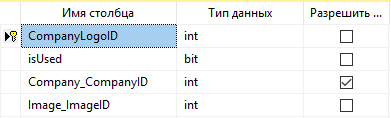


Рисунок 28 – Структура таблицы «CompanyLogos»

1. Таблица «Companies»

Назначение: хранение информации о компаниях (рисунок 29).

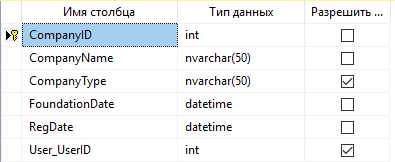


Рисунок 29 – Структура таблицы «Companies»

1. Таблица «Avatars»

Назначение: хранение информации об аватарах пользователей (рисунок 30).

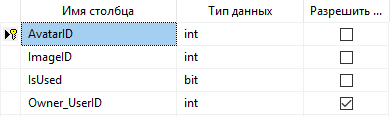


Рисунок 30 – Структура таблицы «Avatars»

1. Таблица «AccessRights»

Назначение: хранение информации о правах сотрудников (рисунок 31).

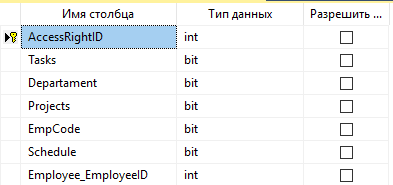


Рисунок 31 – Структура таблицы «AccessRights»

* 1. Описание алгоритма

В этой части дипломного проекта подробно разобрано описание программного продукта:

* функции и их значения;
* структура приложения;
* входные и выходные данные.

1. Общий алгоритм работы.

Когда пользователь запускает приложение – он попадает в окно авторизации. Он может ввести свои данные и войти в панель пользователя, если уже зарегистрирован. Пользователь может так же зарегистрироваться. После регистрации, вернувшись на главное окно (окно авторизации) он может зарегистрировать свою компанию. При входе в систему пользователь, в зависимости от своей роли, попадает в панель пользователя или панель администратора.

Панель пользователя предоставляет доступ к таким окнам как:

* настройки профиля;
* сообщения;
* задачник;
* код компании (если он не является сотрудником какой-либо компании);
* компания (там можно просмотреть успеваемость по своему отделу или собственную).

Панель администратора предоставляет следующие возможности:

* добавить администратора;
* удалить компанию;
* удалить пользователя;
* сообщения.

Как говорилось ранее, пользователь может настроить свой профиль, а именно: изменить учётные данные (логин, пароль, почту), загрузить изображение, например, аватар профиля.

В задачнике пользователь может просмотреть свои уже имеющиеся задачи, отметить задачу как выделенную, удалить или изменить, либо создать новую задачу, а перейдя на вкладку «проекты» может создать целый проект, добавить в него последовательность задач. Проекты можно так же изменять, удалять.

Нажав на кнопку «сообщения», пользователь может просмотреть полученные сообщения и отправить сообщение другому пользователю.

Кнопка «код компании» даёт пользователю ввести код, который должна прислать компания для подтверждения профиля, как сотрудника этой компании. После удачного подтверждения, в зависимости от поставленных прав доступа, пользователь может взаимодействовать с проектами и задачами компании, перейдя на панель компании из панели пользователя. Также в профиле будет отражаться его статус (работа/выходной).

У администратора есть доступ к удалению компаний или пользователей, если те хотели сделать это самостоятельно, но не смогли по каким-либо причинам.

Администратор получает письма от пользователей и других администраторов. Так приложение обеспечивает обратную связь с пользователями приложения, и разработчик может узнать, какой функционал необходим пользователям.

Для того, чтобы пользователи знали, кому отправлять сообщения, почта, которая у каждого пользователя индивидуальная, будет указана в правом верхнем углу панели пользователя, ее можно будет скопировать при помощи кнопки. Так же работает почта остальных пользователей.

На рисунке 32 и в приложении Г можно увидеть блок-схему приложения.

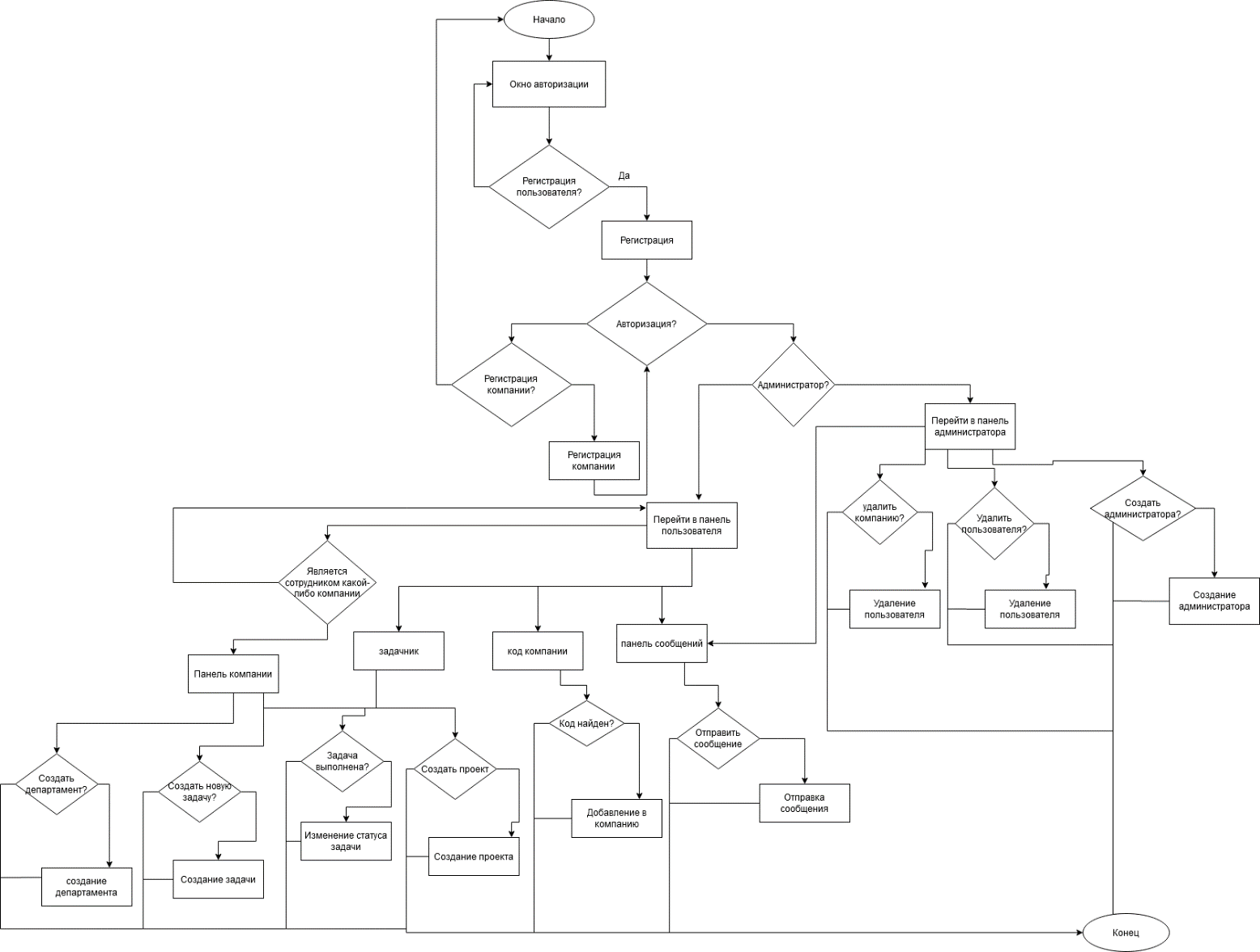


Рисунок 32 – Блок-схема работы приложения

* 1. Программная реализация

В данном разделе детально рассматривается интерфейс, реализованный в программе.

На рисунке 33 изображена схема навигации приложения.

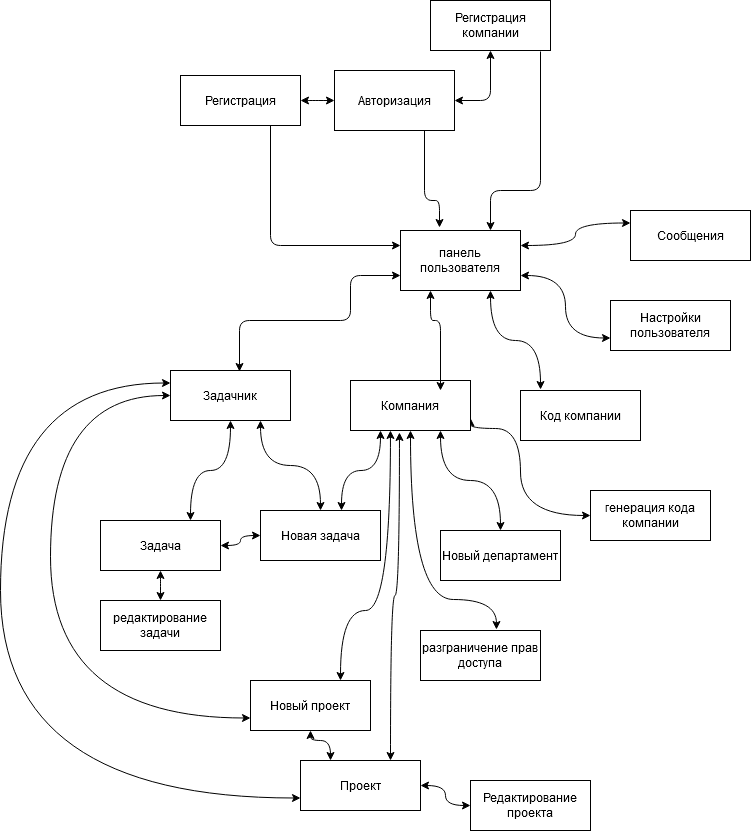


Рисунок 33 – Схема навигации приложения

1. Окно авторизации.

На рисунках 34 – 35 показано главное окно приложения – окно авторизации.

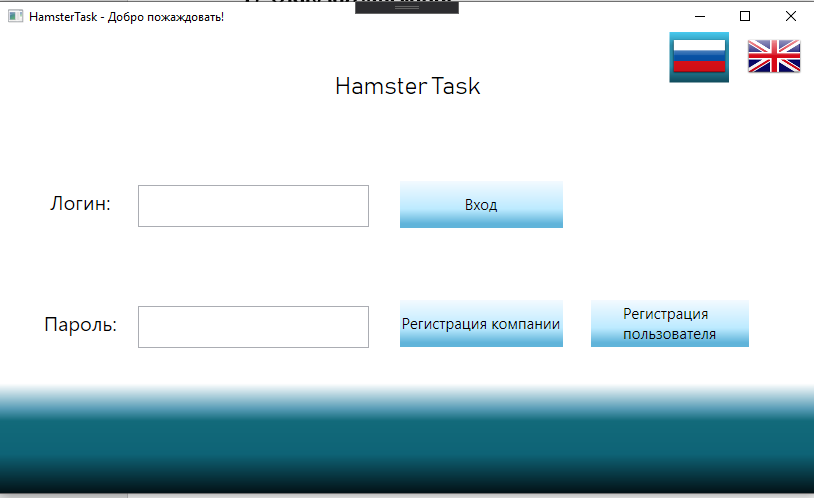


Рисунок 34 – Окно авторизации на русском языке

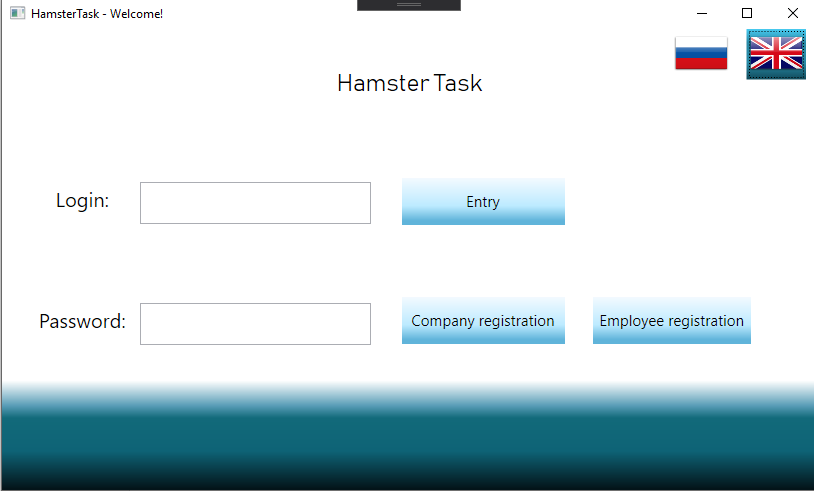


Рисунок 35 – Окно авторизации на английском языке

1. Окно регистрации.

На рисунке 36 отображено окно регистрации.

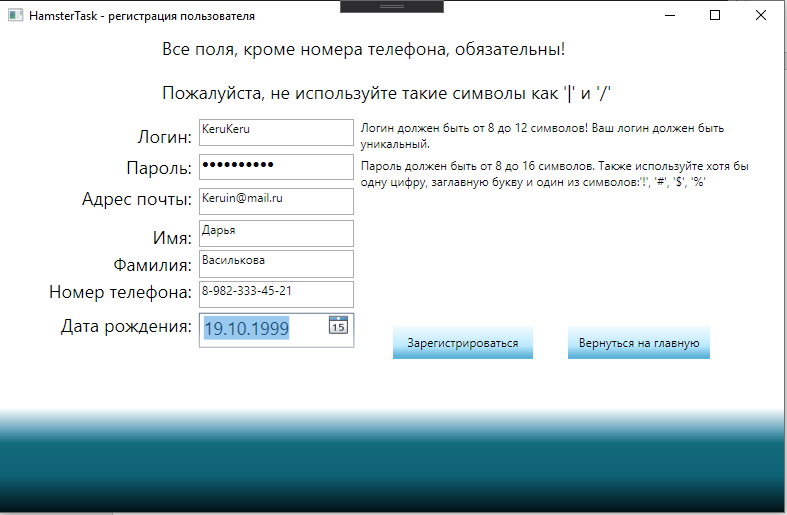


Рисунок 36 – Окно регистрации пользователя

1. Окно регистрации компании.

Для того, чтобы зарегистрировать свою компанию, пользователь должен быть уже зарегистрирован в системе. Для регистрации нужно ввести название, тип компании и дату её создания. На рисунке 37 показано окно регистрации компании.

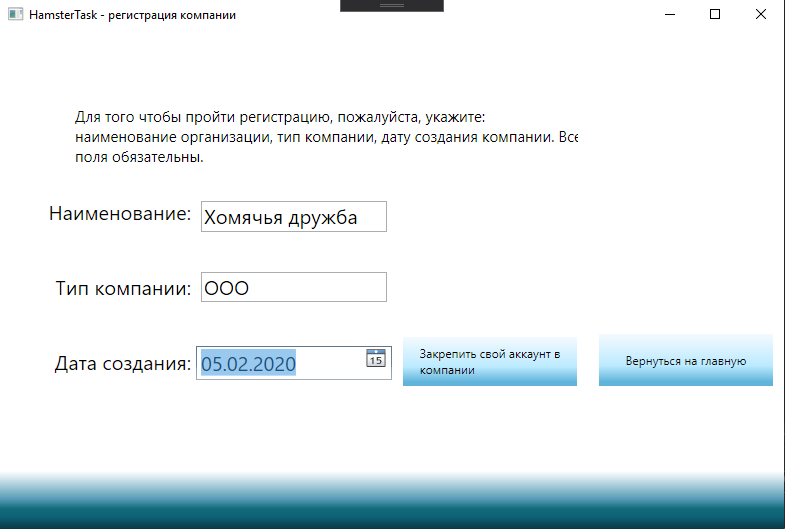


Рисунок 37 – Окно регистрации компании

После введения данных идёт закрепление учётной записи пользователя за этой компанией. На рисунке 38 изображено окно закрепления учётной записи.

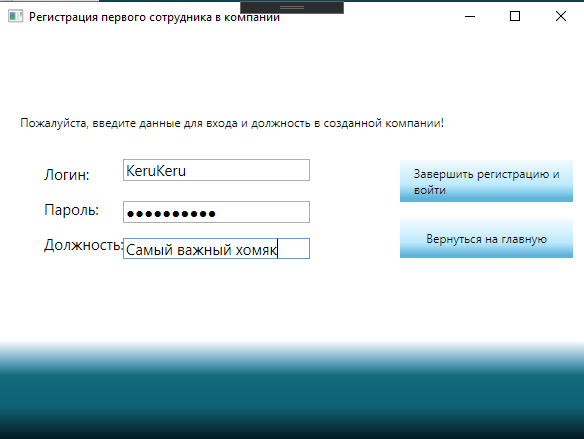


Рисунок 38 – Окно закрепления пользователя в компании

1. Панель пользователя.

После удачной авторизации пользователь попадает в панель пользователя, которая изображена на рисунке 39.

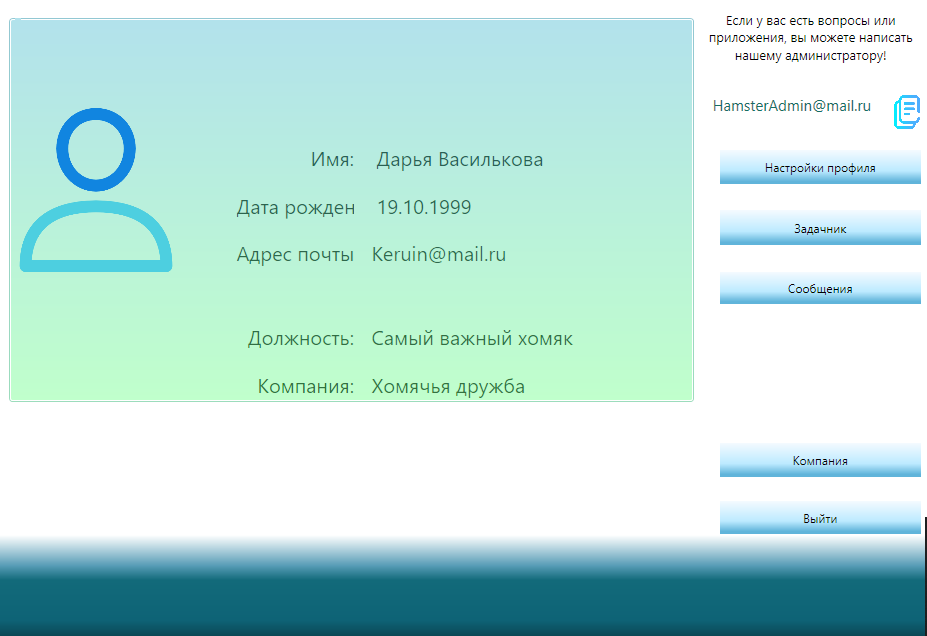


Рисунок 39 – Панель пользователя

1. Задачник.

Из панели пользователя можно попасть в задачник. Тут пользователь имеет возможность просмотреть или добавить задачи. Или же, перейдя на вкладку «проекты», просмотреть существующие проекты или создать новый проект. Окно задачника представлено на рисунке 40.

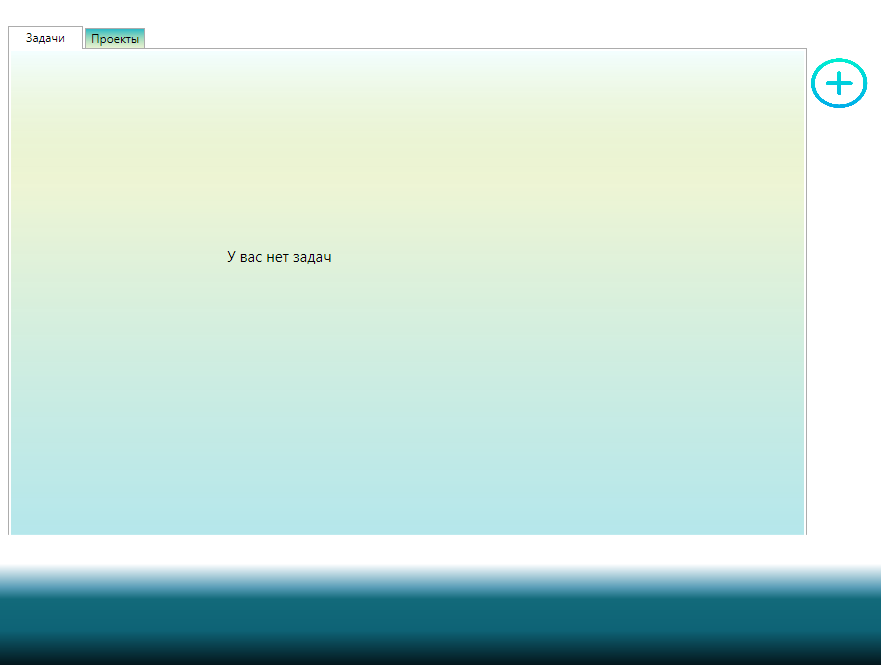


Рисунок 40 – Окно приложения «Задачник»

1. Создание задачи.

Если пользователь создаёт задачу из окна «задачник», то он не может назначить исполнителем никого кроме себя, так как тут создаются в основном личные задачи. Но просматривать задачи, которые назначены из компании он может.

На рисунке 41 можно увидеть окно добавления задачи.

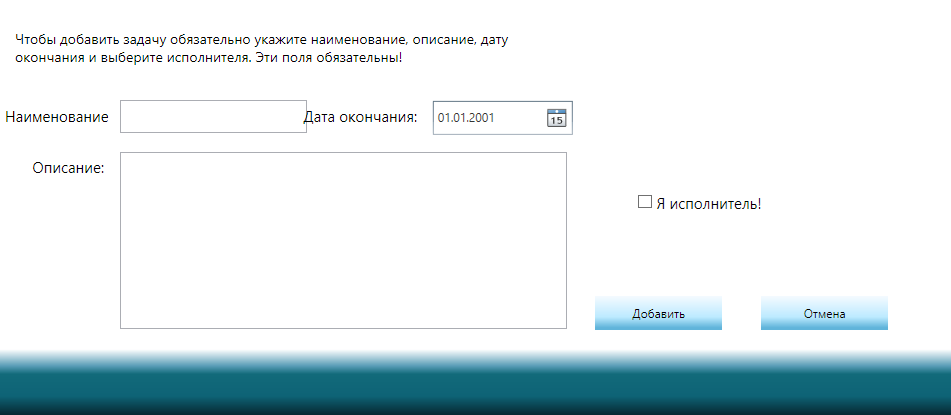


Рисунок 41 – Окно добавления задачи

1. Добавление проекта.

Для того, чтобы создать новый проект, необходимо перейти во вкладку проектов и нажать на ту же кнопку справа, которая будет добавлять задачу или проект в зависимости от выбранной вкладки. На рисунке 42 представлено окно добавления проекта.

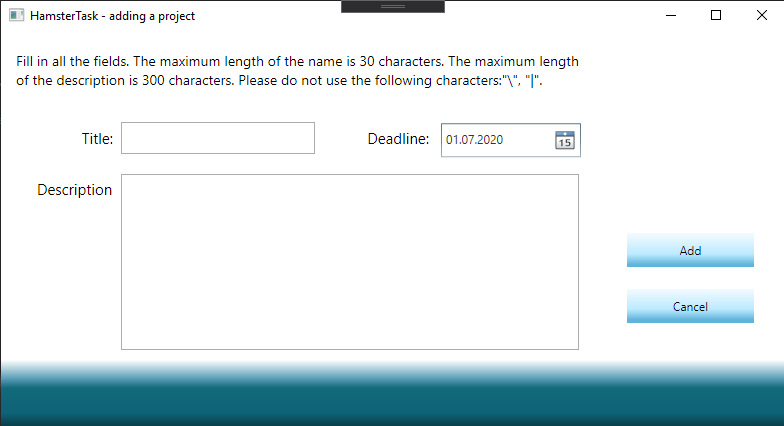


Рисунок 42 – Окно добавления проекта

1. Просмотр задачи.

Задачу можно рассмотреть подробнее, для этого существует окно задачи. С помощью неё можно изменить, отметить задачу выполненной или удалить. На рисунке 43 изображено окно задачи.

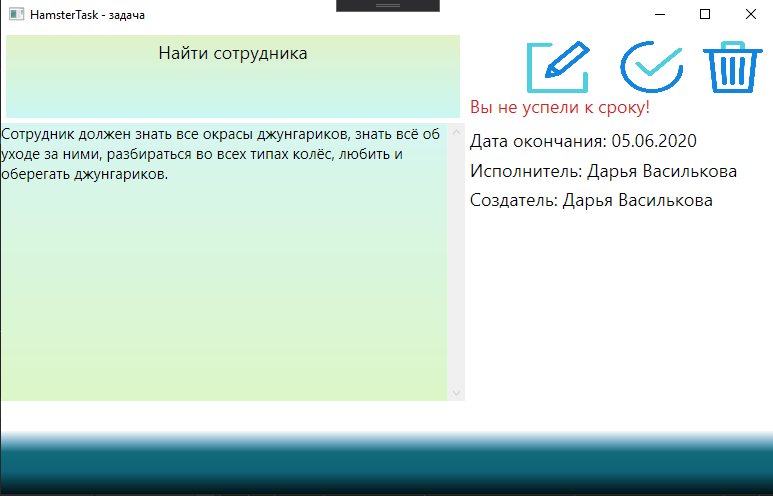


Рисунок 43 – Окно задачи

1. Окно сообщений.

Окно сообщений позволяет просмотреть и отправлять сообщения. Так как адресата указывают по почте, указанной в профиле, есть возможность копировать её при помощи специальной кнопки, которую видно на рисунке 39. То есть, можно скопировать почту выбранного пользователя и вставить, нажав на кнопку рядом с полем получателя, после чего поле автоматически заполнится.

На рисунке 44 изображено окно сообщений.

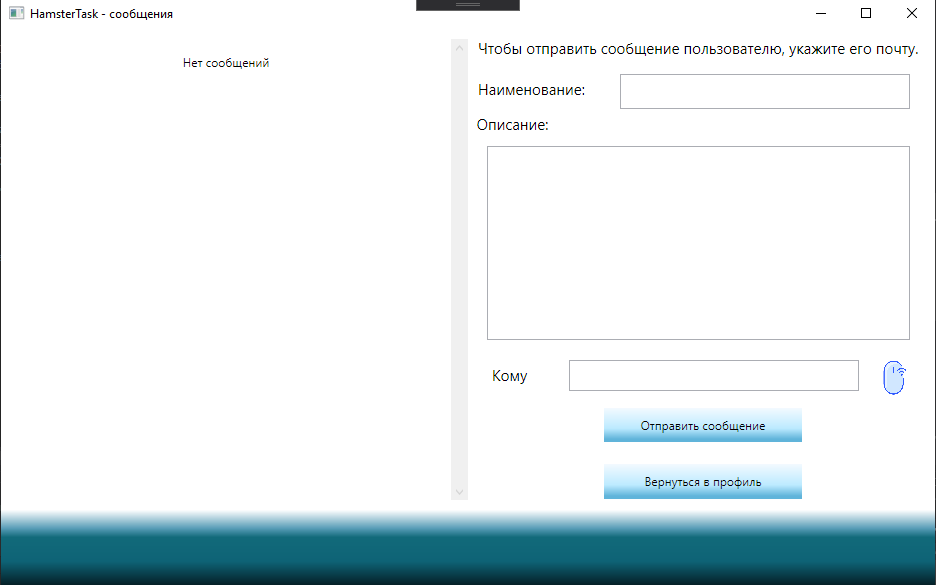


Рисунок 44 – Окно сообщений

1. Редактирование профиля пользователя.

На рисунке 45 представлено окно редактирования профиля пользователя.

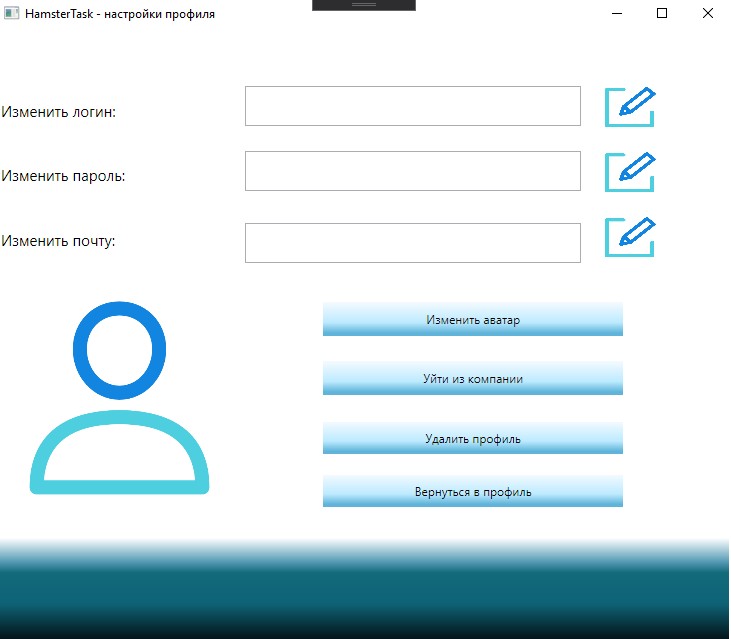


Рисунок 45 – Окно настроек профиля

1. Окно настроек прав сотрудника.

Каждому сотруднику компания может назначить права. В зависимости от отмеченных пунктов программа будет разрешать или запрещать пользователю с настроенными правами тот или иной функционал. Например, если сотрудник не имеет галочки в графе «генерация кода сотрудника» то, соответственно, программа не разрешит ему создавать код для добавления сотрудника в компанию.

Так же присутствует функция вставки почты пользователей.

Окно настроек прав доступа изображено на рисунке 46.

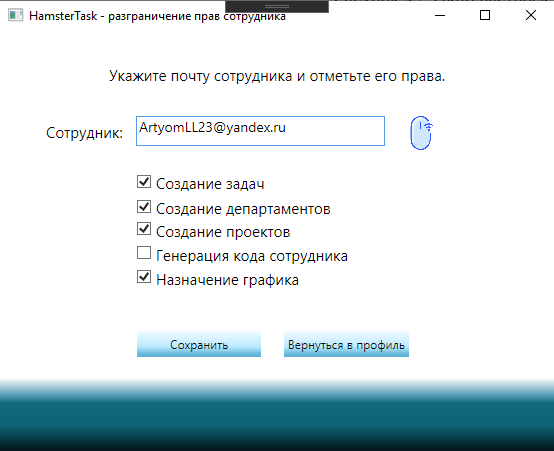


Рисунок 46 - Окно настроек доступа

1. Панель администратора.

У приложения есть панель администратора, которая предназначена для разработчиков. Здесь они могут получать письма от пользователей и по возможности отвечать. Эта панель позволяет добавлять новых администраторов или удалять компании или пользователей.

На рисунке 47 изображена панель администратора.

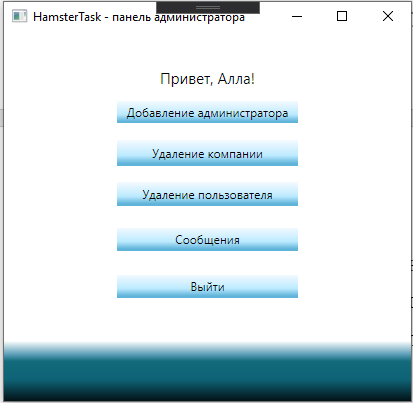


Рисунок 47 – Панель администратора

* 1. Тестирование и отладка

Тестирование – это процесс проверки работы приложения, при котором вводятся данные, результат которых уже известен.

Отладка – это этап разработки, на котором разработчик может увидеть ошибки и исправить их. Целью этого раздела является процесс тестирования приложения «HamsterTask». В таблицах 1-3 вы можете просмотреть этапы тестирования.

Таблица 1 – Тестирование регистрации тест-кейс №1

|  |  |  |  |
| --- | --- | --- | --- |
| Название | | Описание | |
| Имя проекта | | Desktop-приложение «HamsterTask» | |
| Версия | | 1.0 | |
| Имя тестера | | Иван | |
| Дата тестирования | | 30.05.2020 14:34 | |
| Test Case # | | 1 | |
| Приоритет | | Высокий | |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Название теста | «Регистрация» |
| Резюме | Проверка корректности работы регистрации |
| Шаги | 1. Запуск компьютера. 2. Запуск программы «HamsterTask». 3. Нажать на кнопку «Регистрация пользователя». 4. Ввести данные: логин, пароль, почта, имя, фамилия, номер телефона, дата рождения. 5. Нажать кнопку «Зарегистрироваться». |
| Данные теста | Логин: Artyom23  Пароль: Pass234#  Почта: ArtyomLL23@yandex.ru  Имя: Артём  Фамилия: Иванов  Номер телефона: 8-911-235-43-56  Дата рождения: 16.08.1992 |
| Ожидаемый результат | Пользователь зарегистрирован |
| Фактический результат | Пользователь зарегистрирован |
| Постусловия | Нет |
| Статус | Пройдено |

Таблица 2 – Тестирование авторизации тест-кейс №2

|  |  |
| --- | --- |
| Название | Описание |
| Имя проекта | Desktop-приложение «HamsterTask» |
| Версия | 1.0 |
| Имя тестера | Артём |
| Дата тестирования | 30.05.2020 15:03 |
| Test Case # | 2 |
| Приоритет | Высокий |
| Название теста | «Авторизация» |
| Резюме | Проверка корректности работы авторизации |
| Шаги | 1. Запуск компьютера. 2. Запуск программы «HamsterTask». 3. Ввести данные: логин, пароль. 4. Нажать кнопку «Войти». |

Продолжение таблицы 2

|  |  |
| --- | --- |
| Данные теста | Логин: Artyom23  Пароль: Pass234# |
| Ожидаемый результат | Вход завершён успешно |
| Фактический результат | Вход завершён успешно |
| Постусловия | Нет |
| Статус | Пройдено |

Таблица 3 – Тестирование авторизации тест-кейс №3

|  |  |
| --- | --- |
| Название | Описание |
| Имя проекта | Desktop-приложение «HamsterTask» |
| Версия | 1.0 |
| Имя тестера | Елена |
| Дата тестирования | 01.06.2020 12:00 |
| Test Case # | 3 |
| Приоритет | Высокий |
| Название теста | «Авторизация» |
| Резюме | Проверка корректности работы добавления задачи |
| Шаги | 1. Запуск компьютера. 2. Запуск программы «HamsterTask». 3. Ввести данные: логин, пароль. 4. Нажать кнопку «Войти». 5. Нажать кнопку «Задачник». 6. Нажать кнопку «Добавить». 7. Ввести данные: название, описание, срок окончания, отметить исполнителя. |
| Данные теста | Логин: LenaKim1  Пароль: KimChi54  Название: Test1  Описание: TestTestTest  Срок окончания: 03.07.2020 |
| Ожидаемый результат | Задача добавлена |
| Фактический результат | Задача добавлена |
| Постусловия | Нет |
| Статус | Пройдено |

## **Оценка результатов выполненных задач**

Задачей данного дипломного проекта является создание приложения, которым могут пользоваться компании для менеджмента задач и проектов. Данная программа должна предоставить возможность пользователям создавать ряд задач, согласно собственному плану, отметив для каждой свой срок сдачи, а также предоставить возможность создания проектов и разделения по ним соответствующих задач.

Приложение должно быть доступно так же и для обычных пользователей, которые ищут приложение для удобного введения списка дел.

Дополнительно ко всему реализована возможность обратной связи с администраторами, которым можно написать обращение в самом приложении, без перехода в браузер, и получить ответ там же. Такой вариант обратной связи сэкономит большое количество времени сотрудников, которое они могут потратить на работу.

В данном приложении понятный и лёгкий интерфейс, который добавит пользователю удобств, а всплывающие подсказки не дадут запутаться.

Программный продукт не нагружает систему и занимает совсем немного места на жёстком диске, что позволяет использовать его даже на самых слабых компьютерах.

Итоговый программный продукт имеет весь необходимый функционал, в том числе возможность регистрации пользователей и компаний, создание, редактирование и удаление задач и проектов, редактирование отделов компании и их сотрудников, назначение сотрудников на отдельные задачи и проекты, связь между пользователями и разработчиком, общение между пользователями посредством личных сообщений, анализ эффективность сотрудников и другое.

Учитывая всё вышеперечисленное, можно сказать, что программный продукт реализован в рамках технического задания, а поставленные задачи выполнены полностью.

# ЭКОНОМИЧЕСКАЯ ЧАСТЬ

Расчёт предполагаемой прибыли от реализации созданного программного продукта.

## **3.1 Описание продукта**

Дипломный проект представляет собой корпоративное приложение для удобного управления проектами и менеджмента времени.

Приложение имеет следующие свойства:

* позволяет обычному пользователю создавать задачи и целые проекты, в которых так же могут быть задачи;
* зарегистрированные компании могут создавать департаменты – отдельные части компании, как это обычно бывает, занимающиеся определёнными проектами, просматривать информацию о каждом департаменте, создавать новые проекты и задачи, генерировать код для сотрудников для доступа к работе с задачами компании;
* так же в приложении есть функция распределения прав для пользователей, благодаря которым создавать проекты и задачи, указывать исполнителей и добавлять сотрудников смогут лишь назначенные сотрудники.

Данный раздел помогает оценить стоимость данного продукта и возможную прибыль от его реализации.

## **3.2** **Описание и расчёт затрат на выполнение проекта**

Работа над данным проектом требовала определённых затрат: материальных, временных и трудовых. Со стороны экономики прибыль от реализации проекта должна окупить и покрыть наши затраты. Необходимо рассчитать стоимость всех используемых продуктов: техника, материалы, рабочая сила и потраченное время. Плюс ко всему, хорошая прибыль послужит дальнейшему продвижению и росту проекта. Ведь пользователям понадобятся новые функции, которые можно будет добавить при обновлении приложения. И более широкий функционал привлечет внимание новых пользователей.

## **3.2.1 Расчет трудоемкости**

Трудоёмкость – это перечень основных этапов и работ, затронутых при разработке данного проекта.

Этапы работы над данным проектом:

* разработка технического задания (ТЗ): получение ТЗ.

1. Подготовительный этап:

* сбор необходимой информации;
* выбор программных средств, фреймворков;
* выбор среды разработки;
* разработка общей методики создания продукта.

1. Основной этап:

* проектирование базы данных с помощью Entity Framework 6.0;
* проектирование и разработка серверной части приложения .net Framework;
* проектирование и разработка интерфейса .net Framework + WPF;
* отладка.

1. Завершающий этап:

* подготовка технической документации;
* презентация продукта;
* сдача продукта.

Трудоемкость выполнения работы оценивается в человеко-часах. Трудоемкость по видам работ представлена в таблице 4:

Таблица 4 – Расчёт трудоёмкости

|  |  |  |
| --- | --- | --- |
| № п.п. | Виды работ | Трудоёмкость |
| 1 | Получение ТЗ | 1 |
| 2 | Сбор необходимой информации | 3 |
| 3 | Выбор программных средств, фреймворков | 1 |
| 4 | Выбор среды разработки | 5 |
| 5 | Разработка общей методики создания продукта | 10 |
| 6 | Проектирование базы данных с помощью Entity Framework 6.0 | 3 |
| 7 | Проектирование и разработка серверной части приложения .net Framework | 5 |
| 8 | Проектирование и разработка интерфейса .net Framework + WPF | 5 |
| 9 | Отладка | 10 |
| 10 | Подготовка технической документации | 10 |
| 11 | Презентация продукта | 3 |
| 12 | Сдача продукта | 1 |
|  | Итого: | 57 |

В итоге суммарная трудоёмкость составила 57 чел./час.

## **3.2.2 Расчет затрат на оплату труда**

Теперь, получив трудоёмкость, можно рассчитать расходы на заработную плату. Опираясь на данные с сайта hh.ru, можно определить, что средняя заработная плата программиста .NET без опыта работы составляет 25000 рублей. То есть, за один рабочий месяц (21 дней по 8 часов) стоимость оплаты составляет – 168 рублей в час.

Таким образом, расходы на заработную плату по проекту составляют:

ЗП = 168 \* 57= 8482 руб.

Отчисления на ЗП (Пенсионный Фонд, Фонд социального страхования, Фонд обязательно медицинского страхования, территориальные фонды медицинского страхования) составляют 30%:

8482 \* 0,3 = 2544 руб.

## **3.2.3 Расходные материалы**

В приведённой ниже таблице 5 показаны материалы, затраченные на реализацию проекта:

Таблица 5 – Расходные материалы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Материальные ресурсы | Единица измерения | Потребление | Цена за единицу, руб. | Сумма, руб. |
| 1 | Хостинг | шт. | 1 | 300 | 300 |
| 2 | Офисная бумага А4 | упаковка | 1 | 219 | 219 |
| 3 | Файлы | упаковка | 1 | 121 | 121 |
| 4 | Папка | шт. | 1 | 163 | 163 |
| 5 | USB-флешка | шт. | 1 | 300 | 300 |
|  | Итог |  |  |  | 1103 |

## **3.2.4 Расчёт амортизации**

Стоимость компьютера составляет 35000 рублей. Используем его в течение 3х лет (36 месяцев). Годовая амортизация составляет 1250 руб. В день – 7,44 руб. Умножив на трудоёмкость, можно получить амортизацию:

АО = 7,44 \* 57 = 424 руб.

## **3.2.5 Расчёт электроэнергии**

Персональный компьютер в среднем потребляет 0,4 КВт/час. Следовательно, за время разработки проекта будет потреблено:

0,4 \* 57 = 22,8 КВт

Стоимость одного киловатта электроэнергии составляет 2,62 руб. Затраты на электроэнергию рассчитываются таким образом:

22,8 \* 2,62 = 59,74 руб.

## **3.2.6 Расчёт сметы затрат**

В таблице 6 приведён расчёт сметы затрат.

Таблица 6 – Расчёт сметы затрат

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование затрат | Сумма, руб. |
| 1 | Зарплата | 8482 |
| 2 | Отчисления на ЗП | 2544 |
| 3 | Расходные материалы | 1103 |
| 4 | Амортизационные отчисления | 424 |
| 5 | Расходы на электроэнергию | 59,74 |
|  | Итого: | 12612,74 |

Исходя из таблицы, суммарная стоимость затрат составляет 12612,74 рубля.

## **3.3 Расчет экономического эффекта**

Экономический эффект предполагает прибыль от реализации данного продукта.

Приложение обладает следующими свойствами:

* производительность;
* малые системные требования;
* удобный интерфейс;
* обратная связь с разработчиками;
* безопасность хранения данных;
* удобное распределение задач для обычного пользователя;
* удобный интерфейс для распределения задач и проектов по департаментам.

Средняя плата за использования приложения составляет 200 рублей в месяц, следовательно, в год 2400 рублей. Если продукт будут использовать 7 клиента, то получается 16900 рублей.

В таком случае прибыль составит 4187,26 рублей.

Модернизация и продвижение проекта, благодаря обратной связи от клиентов, даст нам возможность в разы улучшить проект и притянуть ещё больше клиентов, которые будут знать, что в случае необходимости они могут получить недостающий функционал. Ведь только общение с клиентами может показать, чего не хватает для автоматизации и удобства работы.

ЗАКЛЮЧЕНИЕ

В рамках дипломного проекта было разработано настольное приложение «HamsterTask», обладающее свойствами Task-менеджера, имеющее простой и понятный интерфейс, которое позволит распределить задачи по компании, не затрачивая на это много ресурсов и времени. Пользователи могут отправлять предложения недостающего для них функционала прямо в приложении и получать ответ в нём же.

Исполнители, при появлении новых задач, будут уведомлены оповещением, и рядом с задачником всегда будет виден счётчик ещё не выполненных задач.

Компания так же может просматривать, как справляется работник с задачами и проектами в целом или по отделам/департаментам.

Приложение является мультиязычным, что может расширить количество пользователей, а также разработчики имеют возможность узнать глобальнее о потребностях пользователей. Плюс ко всему, английский язык – это международный язык, который даст общение с различными людьми в разных странах и которые, в случае необходимости, могут испытывать потребность дополнить приложение новым языком.

Приложение имеет скромные аппаратные требования, что позволяет экономить на глобальной модернизации аппаратных средств.

Дипломный проект – это показание своих знаний на практике, приобретённых в процессе обучения в учебном заведении, закреплены, а также полученных новых знаний и технологий во время его разработки:

* WPF;
* Entity Framework 6.0;
* C#;
* .NET Framework.

В процессе работы над дипломным проектом были получены знания в области объектно-ориентированного программирования. Полученный опыт принесёт пользу в дальнейших реализациях более сложных задач.

ЛИТЕРАТУРА

1. Дж. Рихтер. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. Питер, 2018. 896с.
2. Мэтью Мак-Дональд. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C#. Вильямс, 2016. 1024с.
3. Эндрю Троелсен, Филипп Джепикс. Язык программирования C# 7 и платформы .NET и .NET Core. 8-е издание. Apress, 2017. 1410с.

СПИСОК ИСТОЧНИКОВ

1. ГОСТ Р 7.0.97-2016. Национальный стандарт Российской Федерации. Система стандартов по информации, библиотечному и издательскому делу. Организационно-распорядительная документация. Требования к оформлению документов" (ред. от 14.05.2018).
2. ГОСТ 19.202-78 Спецификация. Требования к содержанию и оформлению;
3. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению.
4. https://entityframework.net/ - сайт с документацией по Entity Framework.
5. https://habr.com/ - сайт с потоками различных статей, включая IT-сферу.
6. https://metanit.com/ - сайт о программировании с различными языками и технологиями.

# ПРИЛОЖЕНИЕ А – Серверная часть приложения

public class Cipher

{

public static byte[] NewSalt()

{

RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();

byte[] salt = new byte[8];

rng.GetBytes(salt);

return salt;

}

public static byte[] PassHash(string pass, byte[] salt)

{

HashAlgorithm algorithm = new SHA256Managed();

var temp = Encoding.UTF8.GetBytes(pass);

byte[] res = new byte[temp.Length + salt.Length];

int i = 0;

foreach(var b in temp)

{

res[i++] = b;

}

foreach (var b in salt)

{

res[i++] = b;

}

return algorithm.ComputeHash(res);

}

}

namespace HamsterServer.DATA.Entities

{

public class AccessRight

{

[Key]

public int AccessRightID { get; set; }

[Required]

public Employee Employee { get; set; }

public bool Tasks { get; set; }

public bool Departament { get; set; }

public bool Projects { get; set; }

public bool EmpCode { get; set; }

public bool Schedule { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class Avatar

{

[Key]

public int AvatarID { get; set; }

[Required]

public int ImageID { get; set; }

[Required]

public bool IsUsed { get; set; }

public User Owner { get; set; }

public Image Image { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class Company

{

public Company()

{

Employees = new HashSet<Employee>();

EmployeeCodes = new HashSet<EmployeeCode>();

Departments = new HashSet<Department>();

}

[Key]

public int CompanyID { get; set; }

[Required]

[StringLength(50)]

public string CompanyName { get; set; }

[StringLength(50)]

public string CompanyType { get; set; }

[Required]

public DateTime FoundationDate { get; set; }

[Required]

public DateTime RegDate { get; set; }

public User User { get; set; }

public virtual ICollection<Employee> Employees { get; set; }

public virtual ICollection<Department> Departments { get; set; }

public virtual ICollection<EmployeeCode> EmployeeCodes { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class CompanyLogo

{

[Key]

public int CompanyLogoID { get; set; }

[Required]

public bool isUsed { get; set; }

[Required]

public Image Image { get; set; }

public Company Company { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class DepartamentEmployees

{

[Key]

public int DepartamentEmployeeID { get; set; }

public Department Department { get; set; }

public Employee Employee { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class Department

{

[Key]

public int DepartamentID { get; set; }

[Required]

public string DepartmentName { get; set; }

[Required]

public Company Company { get; set; }

public ICollection<DepartamentEmployees> DepartamentEmployees { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class Employee

{

[Key]

public int EmployeeID { get; set; }

[Required]

public User User { get; set; }

[Required]

public Company Company { get; set; }

public Position Position { get; set; }

public ICollection<DepartamentEmployees> DepartamentEmployees { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class EmployeeCode

{

[Key]

public int CodeID { get; set; }

[Required]

[StringLength(6)]

public string Code { get; set; }

[Required]

public bool isUsed { get; set; }

[Required]

public Company Company { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class Image

{

[Key]

public int ImageID { get; set; }

[Required]

public User User { get; set; }

public virtual ICollection<CompanyLogo> CompanyLogos { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class Message

{

[Key]

public int MessageID { get; set; }

[Required]

[MaxLength(30)]

public string Title { get; set; }

[Required]

[MaxLength(600)]

public string Content { get; set; }

public User UserTo\_UserId { get; set; }

public User UserFrom\_UserId { get; set; }

[Required]

public DateTime TimeSend { get; set; }

public bool isCheck { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class Position

{

[Key]

public int PositionID { get; set; }

[StringLength(50)]

[Required]

public string PositionName { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class Projects

{

[Key]

public int ProjectID { get; set; }

[Required]

[StringLength(30)]

public string Title { get; set; }

[Required]

[StringLength(300)]

public string Descript { get; set; }

[Required]

public DateTime Deadline { get; set; }

public Company Company { get; set; }

[Required]

public User User { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class ProjectsTasks

{

[Key]

public int ProjectsTasksID { get; set; }

[Required]

public int ProjectID { get; set; }

[Required]

public int TaskID { get; set; }

public Projects Projects { get; set; }

public Task Task { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class Role

{

[Key]

public int RoleId { get; set; }

[StringLength(20)]

[Required]

public string RoleName { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class Sсheldue

{

[Key]

public int SсheldueID { get; set; }

public DateTime DateStart { get; set; }

public DateTime DateEnd { get; set; }

public Employee Employee {get;set;}

}

}

namespace HamsterServer.DATA.Entities

{

public class Task

{

[Key]

public int TaskID { get; set; }

[Required]

[StringLength(30)]

public string Title { get; set; }

[Required]

[StringLength(600)]

public string Descript { get; set; }

[Required]

public DateTime CreateDate { get; set; }

public DateTime Deadline { get; set; }

public bool isDone { get; set; }

public User User { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class TaskExecutors

{

[Key]

public int TaskExecutorsID { get; set; }

[Required]

public Task Task { get; set; }

[Required]

public User User { get; set; }

}

}

namespace HamsterServer.DATA.Entities

{

public class User

{

public User()

{

Images = new List<Image>();

}

[Key]

public int UserID { get; set; }

[Required]

[StringLength(50)]

public string FirstName { get; set; }

[Required]

[StringLength(50)]

public string SecondName { get; set; }

[Required]

[MaxLength(8)]

public byte[] Salt { get; set; }

[Required]

[MaxLength(100)]

public byte[] PassHash { get; set; }

[Required]

[StringLength(50)]

public string Login { get; set; }

public DateTime Birth { get; set; }

[Required]

public DateTime RegDate { get; set; }

[Required]

public int RoleID { get; set; }

[StringLength(15)]

public string PhoneNumber { get; set; }

[StringLength(50)]

[Required]

public string Email { get; set; }

public Role Role { get; set; }

public virtual ICollection<Image> Images { get; set; }

}

}

namespace HamsterServer.Global

{

public class GlobalList

{

public static List<Auth> Authorized = new List<Auth>();

public static User GetUser(string login, string pass)

{

using(DataContext db = new DataContext())

{

User user = db.Users.Where(p => p.Login == login).FirstOrDefault();

byte[] hash = DATA.BLL.Cipher.PassHash(pass, user.Salt);

if(user.PassHash == hash)

{

return user;

}

}

return null;

}

public static Auth IsAuthed(Guid guid)

{

foreach (Auth auth in Authorized)

{

if (auth.guid == guid) return auth;

}

return null;

}

}

}

static void Main(string[] args)

{

using (DataContext db = new DataContext())

{

if (db.Roles.ToList().Count == 0)

{

db.Roles.Add(new Role { RoleName = "Admin" });

db.SaveChanges();

db.Roles.Add(new Role { RoleName = "User" });

db.SaveChanges();

}

}

new Program().ServerListener(new string[] { "http://localhost:8080/" });

}

private void ServerListener(string[] pref)

{

if (!HttpListener.IsSupported)

{

Console.WriteLine("Windows XP SP2 or Server 2003 is required to use the HttpListener class.");

return;

}

if (pref == null || pref.Length == 0)

throw new ArgumentException("prefixes");

HttpListener Listener = new HttpListener();

foreach (string s in pref)

{

Listener.Prefixes.Add(s);

}

Listener.Start();

Console.WriteLine("Listening...");

while (true)

{

HttpListenerContext context = Listener.GetContext();

HttpListenerRequest request = context.Request;

Console.WriteLine("Got request from: {0}", request.Url);

HttpListenerResponse response = context.Response;

string responseString = GetResult(WebUtility.UrlDecode(request.Url.LocalPath));

byte[] buffer = Encoding.UTF8.GetBytes(responseString);

response.ContentLength64 = buffer.Length;

System.IO.Stream output = response.OutputStream;

output.Write(buffer, 0, buffer.Length);

output.Close();

}

}

private string TryRegistration(string ABC, string [] parts)

{

using (DataContext db = new DataContext())

{

User user = new User();

byte[] salt = DATA.BLL.Cipher.NewSalt();

byte[] PassHash = DATA.BLL.Cipher.PassHash(parts[3], salt);

var guid = Guid.NewGuid();

Auth auth;

int role = 2;

string info = string.Empty;

switch (ABC)

{

case "A":

user.Login = parts[2];

user.Salt = salt;

user.PassHash = PassHash;

user.FirstName = parts[4];

user.SecondName = parts[5];

user.Email = parts[6];

user.Birth = Convert.ToDateTime(parts[7]);

user.RegDate = DateTime.Today;

user.RoleID = role;

db.Users.Add(user);

db.SaveChanges();

user = db.Users.Where(p => p.Login == user.Login).FirstOrDefault();

auth = new Auth(guid, user);

Global.GlobalList.Authorized.Add(auth);

Directory.CreateDirectory(@"Documents\Users\" + user.UserID.ToString());

Directory.CreateDirectory(@"Documents\Users\" + user.UserID.ToString() + @"\Images");

info = user.UserID + "|" + user.RoleID + "|" + user.FirstName + "|" + user.SecondName + "|" + user.Birth + "|" + user.Email;

return guid.ToString() + "|" + info;

case "Full":

user.Login = parts[2];

user.Salt = salt;

user.PassHash = PassHash;

user.FirstName = parts[4];

user.SecondName = parts[5];

user.Email = parts[6];

user.PhoneNumber = parts[7];

user.Birth = Convert.ToDateTime(parts[8]);

user.RegDate = DateTime.Now;

user.RoleID = role;

db.Users.Add(user);

db.SaveChanges();

user = db.Users.Where(p => p.Login == user.Login).FirstOrDefault();

auth = new Auth(guid, user);

Global.GlobalList.Authorized.Add(auth);

Directory.CreateDirectory(@"Documents\Users\" + user.UserID.ToString());

Directory.CreateDirectory(@"Documents\Users\" + user.UserID.ToString() + @"\Images");

info = user.UserID + "|" + user.RoleID + "|" + user.FirstName + "|" + user.SecondName + "|" + user.Birth + "|" + user.Email + "|" + user.PhoneNumber;

return guid.ToString() + "|" + info;

default: return "meow";

}

}

private string TryRegistrationOrg(string AB, string [] parts)

{

using (DataContext db = new DataContext())

{

Company company = new Company();

Employee employee = new Employee();

Position position = new Position();

switch (AB) {

case "A":

company.CompanyName = parts[2];

company.CompanyType = parts[3];

company.FoundationDate = Convert.ToDateTime(parts[4]);

company.RegDate = DateTime.Now;

string[] infoUser = TryAuth(parts[5], parts[6]).Split('|');

if (infoUser != null)

{

int id = Convert.ToInt32(infoUser[1]);

employee.User = db.Users.Where(p => p.UserID == id).FirstOrDefault();

company.User = db.Users.Where(p => p.UserID == id).FirstOrDefault();

}

else return "Логин или пароль введены неверно или такого пользователя не сущетствует";

db.Companies.Add(company);

db.SaveChanges();

employee.Company = company;

string name = parts[7];

position = db.Positions.Where(p => p.PositionName == name).FirstOrDefault();

if (position != null)

{

employee.Position = position;

} else {

position = new Position();

position.PositionName = parts[7];

db.Positions.Add(position);

db.SaveChanges();

employee.Position = position;

}

db.Employees.Add(employee);

db.SaveChanges();

Directory.CreateDirectory(@"\Documents\Companies\" + company.CompanyName);

Directory.CreateDirectory(@"\Documents\Companies\" + company.CompanyName + @"\Files\Images");

return TryAuth(parts[5], parts[6]);

default: return "WrongKeys";

}

}

private string TryAuth(string login, string password)

{

using (DataContext db = new DataContext())

{

User user = db.Users.Where(User => User.Login == login).FirstOrDefault();

if(Encoding.UTF8.GetString(user.PassHash) == Encoding.UTF8.GetString(DATA.BLL.Cipher.PassHash(password, user.Salt)))

{

var guid = Guid.NewGuid();

Auth auth = new Auth(guid, user);

Global.GlobalList.Authorized.Add(auth);

string info = string.Empty;

Employee infoEmp = db.Employees.Include("User").Include("Position").Include("Company").FirstOrDefault(emp => emp.User.UserID == user.UserID);

Position positionName = null;

Company companyName = null;

if (infoEmp != null)

{

int pos = infoEmp.Position.PositionID;

int comp = infoEmp.Company.CompanyID;

positionName = db.Positions.Where(p => p.PositionID == pos).FirstOrDefault();

companyName = db.Companies.Where(p => p.CompanyID == comp).FirstOrDefault();

}

if (user.PhoneNumber != null)

{

if (infoEmp != null)

{

info = user.UserID + "|" + user.RoleID + "|" + user.FirstName + "|" + user.SecondName + "|" + user.Birth + "|" + user.Email + "|" + user.PhoneNumber + "|" + positionName.PositionName + "|" + companyName.CompanyName + "|" + "WP";

}

else info = user.UserID + "|" + user.RoleID + "|" + user.FirstName + "|" + user.SecondName + "|" + user.Birth + "|" + user.Email + "|" + user.PhoneNumber;

}

else {

if (infoEmp != null)

{

info = user.UserID + "|" + user.RoleID + "|" + user.FirstName + "|" + user.SecondName + "|" + user.Birth + "|" + user.Email + positionName.PositionName + "|" + companyName.CompanyName + "|" + "W";

} else info = user.UserID + "|" + user.RoleID + "|" + user.FirstName + "|" + user.SecondName + "|" + user.Birth + "|" + user.Email; }

return guid.ToString() + "|" + info;

}

return "bad";

}

}

private string GetTask(string guid, string Num)

{

using (DataContext db = new DataContext())

{

User user = GlobalList.IsAuthed(new Guid(guid)).user;

int i = Convert.ToInt32(Num);

var data = db.TaskExecutors.Include("User").Include("Task").Where(p => p.User.UserID == user.UserID).ToList();

var d = data[i];

var task = db.Tasks.Include("User").Where(t => t.TaskID == d.Task.TaskID).FirstOrDefault();

string title = task.Title;

string Descript = task.Descript;

string deadline = task.Deadline.ToString();

string creater = task.User.FirstName + " " + task.User.SecondName;

string exec = data[i].User.FirstName + " " + data[i].User.SecondName;

string done = task.isDone.ToString();

string TaskID = Convert.ToString(task.TaskID);

return TaskID + "|" + title + "|" + Descript + "|" + deadline + "|" + creater + "|" + exec + "|" + done;

}

}

private string GetOneTask(string guid, string Id)

{

using (DataContext db = new DataContext())

{

User user = GlobalList.IsAuthed(new Guid(guid)).user;

int i = Convert.ToInt32(Id);

var task = db.Tasks.Include("User").Where(t => t.TaskID == i).FirstOrDefault();

var execU = db.TaskExecutors.Include("Task").Include("User").FirstOrDefault(e => e.Task.TaskID == task.TaskID);

string title = task.Title;

string Descript = task.Descript;

string deadline = task.Deadline.ToString();

string creater = task.User.FirstName + " " + task.User.SecondName;

string exec = execU.User.FirstName + " " + execU.User.SecondName;

string done = task.isDone.ToString();

string TaskID = Convert.ToString(task.TaskID);

return TaskID + "|" + title + "|" + Descript + "|" + deadline + "|" + creater + "|" + exec + "|" + done;

}

}

private string GetProject(string guid, string Num)

{

using (DataContext db = new DataContext())

{

User user = GlobalList.IsAuthed(new Guid(guid)).user;

int i = Convert.ToInt32(Num);

var data = db.Projects.Include("User").Where(p => p.User.UserID == user.UserID).ToList();

string title = data[i].Title;

string content = data[i].Descript;

string creater = data[i].User.FirstName + " " + data[i].User.SecondName;

return title + "|" + content + "|" + creater;

}

}

private string GetTasks(string guid)

{

using (DataContext db = new DataContext())

{

User user = GlobalList.IsAuthed(new Guid(guid)).user;

var data = db.TaskExecutors.Include("User").Where(p => p.User.UserID == user.UserID).ToList();

return data.Count.ToString();

}

private string GetProjects(string guid)

{

using (DataContext db = new DataContext())

{

User user = GlobalList.IsAuthed(new Guid(guid)).user;

var data = db.Projects.Include("User").Where(p => p.User.UserID == user.UserID).ToList();

return data.Count.ToString();

}

}

private string CreateTask(string guid, string[] part)

{

using (DataContext db = new DataContext())

{

User user = GlobalList.IsAuthed(new Guid(guid)).user;

User user1 = db.Users.FirstOrDefault(u => u.UserID == user.UserID);

TaskExecutors executors = new TaskExecutors();

DateTime dead = Convert.ToDateTime(part[6]);

switch (part[3])

{

case "U":

DATA.Entities.Task temp = db.Tasks.Add(new DATA.Entities.Task { User = user1, Deadline = dead, CreateDate = DateTime.Now, Descript = part[5], isDone = false, Title = part[4] });

db.TaskExecutors.Add(new TaskExecutors { User = user1, Task = temp });

db.SaveChanges();

return "Task is added";

case "E":

DATA.Entities.Task temp1 = db.Tasks.Add(new DATA.Entities.Task { User = user1, Deadline = dead, CreateDate = DateTime.Now, Descript = part[5], isDone = false });

int id = Convert.ToInt32(part[7]);

User exec = db.Users.FirstOrDefault(u => u.UserID == id);

db.TaskExecutors.Add(new TaskExecutors { Task = temp1, User = exec });

db.SaveChanges();

return "Task is added";

default:

return "no";

}

}

private string SetExecutor(string guid, string TaskID, string UserID)

{

using (DataContext db = new DataContext())

{

User user = GlobalList.IsAuthed(new Guid(guid)).user;

int ExecID = Convert.ToInt32(UserID);

int temp = Convert.ToInt32(TaskID);

User Exec = db.Users.FirstOrDefault(u => u.UserID == ExecID);

DATA.Entities.Task T = db.Tasks.FirstOrDefault(t => t.TaskID == temp);

db.TaskExecutors.Add(new TaskExecutors { Task = T, User = Exec });

db.SaveChanges();

return "Ok";

}

}

private string SetRights(string guid, string UserID, string Report, string Task, string Departament, string Projects, string EmpCode, string Schedule, string Marks)

{

using (DataContext db = new DataContext())

{

User user = GlobalList.IsAuthed(new Guid(guid)).user;

int ID = Convert.ToInt32(UserID);

Employee employee = db.Employees.Include("User").FirstOrDefault(e => e.User.UserID == ID);

bool T = Convert.ToBoolean(Task);

bool D = Convert.ToBoolean(Departament);

bool P = Convert.ToBoolean(Projects);

bool E = Convert.ToBoolean(EmpCode);

bool S = Convert.ToBoolean(Schedule);

db.AccessRights.Add(new AccessRight { Tasks = T, Departament = D, Projects = P, EmpCode = E, Schedule = S, Employee = employee });

db.SaveChanges();

return "Access rights saved";

}

}

private string GetTaskCount(string guid)

{

using (DataContext db = new DataContext())

{

User user = GlobalList.IsAuthed(new Guid(guid)).user;

int count = db.TaskExecutors.Include("User").Where(u => u.User.UserID == user.UserID && u.Task.Deadline < DateTime.Now).Count();

if (count > 0)

{

return count.ToString();

}

else return "No tasks";

}

}

private string GetMesssagesCount(string guid)

{

using (DataContext db = new DataContext())

{

User user = GlobalList.IsAuthed(new Guid(guid)).user;

int count = db.Messages.Include("UserTo\_UserId").Where(m => m.UserTo\_UserId.UserID == user.UserID && m.isCheck == false).Count();

if (count > 0)

{

return count.ToString();

}

else return "No messages";

}

}

# ПРИЛОЖЕНИЕ Б – Клиентская часть приложения

namespace HamsterTask.Helper

{

public class Http

{

public static string GetRequest(string url)

{

string html = string.Empty;

HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);

request.AutomaticDecompression = DecompressionMethods.GZip;

using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())

using (Stream stream = response.GetResponseStream())

using (StreamReader reader = new StreamReader(stream))

{

html = reader.ReadToEnd();

}

return html;

}

}

}

public static string GlobLang;

public static void LanguageSwitch(Window win)

{

ResourceDictionary res = new ResourceDictionary();

switch (Global.GlobLang)

{

case "ENG":

res.Source = new Uri("/Language/ENG/EngDictionary.xaml", UriKind.Relative);

break;

case "RUS":

res.Source = new Uri("/Language/RUS/RusDictionary.xaml", UriKind.Relative);

break;

}

win.Resources.MergedDictionaries.Add(res);

}

public static void LanguageSwitchControll(TaskControl task)

{

ResourceDictionary res = new ResourceDictionary();

switch (Global.GlobLang)

{

case "ENG":

res.Source = new Uri("/Language/ENG/EngDictionary.xaml", UriKind.Relative);

break;

case "RUS":

res.Source = new Uri("/Language/RUS/RusDictionary.xaml", UriKind.Relative);

break;

}

task.Resources.MergedDictionaries.Add(res);

}

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

mc:Ignorable="d"

Title="{DynamicResource HamsterTask.MainWindow}" Height="500" Width="830" WindowStartupLocation="CenterScreen">

<Window.Resources>

<Style x:Key="FocusVisual">

<Setter Property="Control.Template">

<Setter.Value>

<ControlTemplate>

<Rectangle Margin="2" SnapsToDevicePixels="true" Stroke="{DynamicResource {x:Static SystemColors.ControlTextBrushKey}}" StrokeThickness="1" StrokeDashArray="1 2"/>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<SolidColorBrush x:Key="Button.Static.Background" Color="#FFDDDDDD"/>

<SolidColorBrush x:Key="Button.Static.Border" Color="#FF707070"/>

<SolidColorBrush x:Key="Button.MouseOver.Background" Color="#FFBEE6FD"/>

<SolidColorBrush x:Key="Button.MouseOver.Border" Color="#FF3C7FB1"/>

<SolidColorBrush x:Key="Button.Pressed.Background" Color="#FFC4E5F6"/>

<SolidColorBrush x:Key="Button.Pressed.Border" Color="#FF2C628B"/>

<SolidColorBrush x:Key="Button.Disabled.Background" Color="#FFF4F4F4"/>

<SolidColorBrush x:Key="Button.Disabled.Border" Color="#FFADB2B5"/>

<SolidColorBrush x:Key="Button.Disabled.Foreground" Color="#FF838383"/>

<Style x:Key="ButtonStyle1" TargetType="{x:Type Button}">

<Setter Property="FocusVisualStyle" Value="{StaticResource FocusVisual}"/>

<Setter Property="Background" Value="{StaticResource Button.Static.Background}"/>

<Setter Property="BorderBrush" Value="{StaticResource Button.Static.Border}"/>

<Setter Property="Foreground" Value="{DynamicResource {x:Static SystemColors.ControlTextBrushKey}}"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="HorizontalContentAlignment" Value="Center"/>

<Setter Property="VerticalContentAlignment" Value="Center"/>

<Setter Property="Padding" Value="1"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="border" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" Background="{TemplateBinding Background}" SnapsToDevicePixels="true">

<ContentPresenter x:Name="contentPresenter" Focusable="False" HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}" Margin="{TemplateBinding Padding}" RecognizesAccessKey="True" SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}" VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsDefaulted" Value="true">

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

</Trigger>

<Trigger Property="IsMouseOver" Value="true">

</Trigger>

<Trigger Property="IsPressed" Value="true">

</Trigger>

<Trigger Property="IsEnabled" Value="false">

<Setter Property="Background" TargetName="border" Value="{StaticResource Button.Disabled.Background}"/>

<Setter Property="BorderBrush" TargetName="border" Value="{StaticResource Button.Disabled.Border}"/>

<Setter Property="TextElement.Foreground" TargetName="contentPresenter" Value="{StaticResource Button.Disabled.Foreground}"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid Loaded="Grid\_Loaded">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="160.5"/>

<ColumnDefinition Width="165"/>

<ColumnDefinition Width="164"/>

<ColumnDefinition Width="175"/>

<ColumnDefinition Width="145\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="117\*"/>

<RowDefinition Height="121"/>

<RowDefinition Height="121"/>

<RowDefinition Height="117\*"/>

</Grid.RowDefinitions>

<Label Grid.Column="2" Content="Hamster Task" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" FontSize="24" FontFamily="Bahnschrift Light"/>

<Rectangle Name="Border" Grid.Column="4" Margin="5,0,85,59">

<Rectangle.Fill>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF48CAF0" Offset="0"/>

<GradientStop Color="#FF104A5B" Offset="1"/>

<GradientStop Color="#FF298AA6" Offset="0.669"/>

<GradientStop Color="#FF298AA6" Offset="0.268"/>

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<Button Name="RusSw" Grid.Column="4" Margin="5,0,85,59" Opacity="100" Click="RusSw\_Click" BorderBrush="{x:Null}" Cursor="Hand" ClickMode="Press" Style="{DynamicResource ButtonStyle1}">

<Button.Background>

<ImageBrush ImageSource="/Image/ru.png"/>

</Button.Background>

</Button>

<Button Name="EngSw" Grid.Column="4" Margin="80,0,10,59" Opacity="100" Click="EngSw\_Click" BorderBrush="{x:Null}" Cursor="Hand" ClickMode="Press" Style="{DynamicResource ButtonStyle1}">

<Button.Background>

<ImageBrush ImageSource="/Image/uk.png"/>

</Button.Background>

</Button>

<Label x:Name="LabelLogin" Content="{DynamicResource LabelLogin}" Grid.Row="1" Grid.Column="0" FontSize="20" VerticalContentAlignment="Center" HorizontalContentAlignment="Center" />

<Label x:Name="LabelPass" Content="{DynamicResource LabelPass}" Grid.Row="2" Grid.Column="0" FontSize="20" VerticalContentAlignment="Center" HorizontalContentAlignment="Center" />

<TextBox Name="Login" Grid.Row="1" FontSize="20" Grid.ColumnSpan="3" Margin="138,43,121,36" VerticalContentAlignment="Center"/>

<PasswordBox Name="Password" Grid.Row="2" FontSize="20" Grid.ColumnSpan="3" Margin="138,43,121,36" VerticalContentAlignment="Center"/>

<Button x:Name="LoginBtn" Content="{DynamicResource LoginBtn}" FontSize="15" Grid.Column="2" Grid.Row="1" Margin="73,38,101,34" Grid.ColumnSpan="2" Click="LoginBtn\_Click" BorderBrush="{x:Null}">

<Button.Background>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="#FFF4FAFF"/>

<GradientStop Color="#FFBCEAFF" Offset="0.608"/>

<GradientStop Color="#FF60B4DA" Offset="0.919"/>

</LinearGradientBrush>

</Button.Background>

</Button>

<Button x:Name="RegOrgBtn" FontSize="15" Grid.Column="2" Grid.Row="2" Margin="73,36,101,36" Click="RegOrgBtn\_Click" Grid.ColumnSpan="2" BorderBrush="{x:Null}">

<Button.Background>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="#FFF4FAFF"/>

<GradientStop Color="#FFBCEAFF" Offset="0.608"/>

<GradientStop Color="#FF60B4DA" Offset="0.919"/>

</LinearGradientBrush>

</Button.Background>

<TextBlock TextWrapping="Wrap" Text="{DynamicResource RegOrgBtn}"></TextBlock>

</Button>

<Button x:Name="RegEmpBtn" FontSize="15" Grid.Column="3" Grid.Row="2" Margin="100,36,64,36" Grid.ColumnSpan="2" Click="RegEmpBtn\_Click" BorderBrush="{x:Null}">

<Button.Background>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="#FFF4FAFF"/>

<GradientStop Color="#FFBCEAFF" Offset="0.608"/>

<GradientStop Color="#FF60B4DA" Offset="0.919"/>

</LinearGradientBrush>

</Button.Background>

<TextBlock TextWrapping="Wrap" Text="{DynamicResource RegEmpBtn}"></TextBlock>

</Button>

<Rectangle Grid.Column="0" Grid.ColumnSpan="5" Grid.Row="3">

<Rectangle.Fill>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="White"/>

<GradientStop Color="#FF031419" Offset="0.989"/>

<GradientStop Color="#FF126A7A" Offset="0.342"/>

<GradientStop Color="#FF579CB6" Offset="0.232"/>

<GradientStop Color="#FF0E6376" Offset="0.644"/>

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

</Grid>

</Window>

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

{

this.Hide();

new RegistrationOrg().Show();

this.Close();

}

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

{

Global.GlobLang = "RUS";

Global.LanguageSwitch(this);

}

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

{

Global.GlobLang = "ENG";

Border.Margin = EngSw.Margin;

Global.LanguageSwitch(this);

}

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

{

Global.GlobLang = "RUS";

Border.Margin = RusSw.Margin;

Global.LanguageSwitch(this);

}

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

{

new RegistrationUser().Show();

this.Close();

}

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

{

Global.Guid = Helper.Http.GetRequest("http://localhost:8080/auth/" + Login.Text + "/" + Password.Password);

if (Global.Guid != "bad" && Global.Guid != "404")

{

new UserPanel().Show();

this.Close();

}

else MessageBox.Show(TryFindResource("Bad").ToString());

<Window x:Class="HamsterTask.TaskForm"

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:HamsterTask"

mc:Ignorable="d"

Title="{DynamicResource TaskForm}" Height="503" Width="789" WindowStartupLocation="CenterScreen">

<Grid Loaded="Grid\_Loaded">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Label x:Name="TaskName" FontSize="18" Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="3" HorizontalContentAlignment="Center" Margin="5">

<Label.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#72BBE284" Offset="0"/>

<GradientStop Color="#728BECE3" Offset="1"/>

</LinearGradientBrush>

</Label.Background>

</Label>

<Label x:Name="Deadline" Content="{DynamicResource TaskDeadline}" FontSize="18" Grid.Column="3" Grid.Row="1" Grid.ColumnSpan="2" HorizontalContentAlignment="Left"/>

<Label x:Name="Exec" Content="{DynamicResource TaskExecutor}" FontSize="18" Grid.Column="3" Grid.Row="1" Grid.ColumnSpan="2" VerticalContentAlignment="Center" HorizontalContentAlignment="Left"/>

<Label x:Name="Own" Content="{DynamicResource TaskOwner}" FontSize="18" Grid.Column="3" Grid.Row="1" Grid.ColumnSpan="2" VerticalContentAlignment="Bottom" HorizontalContentAlignment="Left"/>

<Label x:Name="TaskFailed" Visibility="Hidden" Content="{DynamicResource TaskFailed}" FontSize="18" Grid.Column="3" Grid.Row="0" Grid.ColumnSpan="2" VerticalContentAlignment="Bottom" HorizontalContentAlignment="Left" Foreground="#FFD61E1E"/>

<ScrollViewer Name="scrollDesc" CanContentScroll="True" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="3" Grid.RowSpan="3">

<TextBlock x:Name="TaskDesk" FontSize="15" TextWrapping="Wrap" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="3" Grid.RowSpan="3">

<TextBlock.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#7FACECE3" Offset="0"/>

<GradientStop Color="#7FBAEC90" Offset="1"/>

</LinearGradientBrush>

</TextBlock.Background>

</TextBlock>

</ScrollViewer>

<Rectangle Grid.Column="0" Grid.ColumnSpan="5" Grid.Row="4" Margin="0,29,0,0">

<Rectangle.Fill>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="White"/>

<GradientStop Color="#FF031419" Offset="0.989"/>

<GradientStop Color="#FF126A7A" Offset="0.342"/>

<GradientStop Color="#FF579CB6" Offset="0.232"/>

<GradientStop Color="#FF0E6376" Offset="0.644"/>

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<Button x:Name="RemTask" Grid.Column="4" HorizontalAlignment="Left" Margin="81,10,0,0" VerticalAlignment="Top" Width="65" Height="54" BorderBrush="{x:Null}" Foreground="{x:Null}" Click="RemTask\_Click">

<Button.Background>

<ImageBrush ImageSource="Image/delete.png"/>

</Button.Background>

</Button>

<Button x:Name="IsDone" Grid.Column="4" HorizontalAlignment="Left" Margin="0,10,0,0" VerticalAlignment="Top" Width="65" Height="54" BorderBrush="{x:Null}" Foreground="{x:Null}" Click="IsDone\_Click">

<Button.Background>

<ImageBrush ImageSource="Image/check.png"/>

</Button.Background>

</Button>

<Image x:Name="Done" Visibility="Hidden" Grid.Column="3" HorizontalAlignment="Right" Height="67" Margin="0,0,80,0" VerticalAlignment="Top" Width="89" Source="/Image/tick.png" Grid.ColumnSpan="2" Stretch="Fill"/>

<Button x:Name="Edit" Grid.Column="3" HorizontalAlignment="Left" Margin="61,10,0,0" VerticalAlignment="Top" Width="65" Height="54" Foreground="{x:Null}" BorderBrush="{x:Null}" Click="Edit\_Click">

<Button.Background>

<ImageBrush ImageSource="Image/Edit.png"/>

</Button.Background>

</Button>

</Grid>

</Window>

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

{

try

{

string[] parts = Global.Guid.Split('|');

if (MessageBox.Show(TryFindResource("MessRemTask").ToString(), TryFindResource("MessRemTaskHead").ToString(), MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

var res = Helper.Http.GetRequest("http://localhost:8080/DeleteTask/" + parts[0] + "/" + Global.GlobTaskID);

if(res == "Deleted!")

{

MessageBox.Show(TryFindResource("MessRemTaskRes").ToString());

} else

{

MessageBox.Show(TryFindResource("MessRemTaskResFail").ToString());

}

}

}

catch { MessageBox.Show(TryFindResource("MessRemTaskResFail").ToString()); }

}

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

{

try

{

string[] parts = Global.Guid.Split('|');

var res = Helper.Http.GetRequest("http://localhost:8080/DeleteTask/" + parts[0] + "/" + Global.GlobTaskID).Split('|');

TaskName.Content = res[1];

TaskDesk.Text = res[2];

Exec.Content += " " + res[5];

Own.Content += " " + res[4];

Deadline.Content += " " + res[3];

if(Convert.ToDateTime(res[3]) >= DateTime.Now)

{

TaskFailed.Visibility = Visibility.Visible;

IsDone.Visibility = Visibility.Hidden;

}

if(res[6] == "true")

{

IsDone.Visibility = Visibility.Hidden;

Done.Visibility = Visibility.Visible;

}

}

catch

{

}

}

<Window x:Class="HamsterTask.RegistrationOrg"

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:HamsterTask"

mc:Ignorable="d"

Title="{DynamicResource HamsterTask.RegistrationOrg}" Height="550" Width="800" WindowStartupLocation="CenterScreen">

<Window.Resources>

<Style TargetType="{x:Type DatePickerTextBox}">

<Setter Property="Text" Value="01.01.2001" />

</Style>

</Window.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="75\*"/>

<RowDefinition Height="75"/>

<RowDefinition Height="71"/>

<RowDefinition Height="79"/>

<RowDefinition Height="71"/>

<RowDefinition Height="74"/>

<RowDefinition Height="74\*"/>

</Grid.RowDefinitions>

<Label x:Name="LabelCompanyName" Content="{DynamicResource LabelCompanyName}" Grid.Row="2" FontSize="20" VerticalContentAlignment="Center" HorizontalContentAlignment="Right"/>

<TextBox Name="CompanyName" Grid.Row="2" Grid.Column="1" Margin="5, 25,5,15" FontSize="20"/>

<Label x:Name="LabelCompanyType" Content="{DynamicResource LabelCompanyType}" Grid.Row="3" FontSize="20" VerticalContentAlignment="Center" HorizontalContentAlignment="Right"/>

<TextBox Name="CompanyType" Grid.Row="3" Grid.Column="1" Margin="5,25,5,24" FontSize="20"/>

<Label x:Name="LabelCompanyBirth" Content="{DynamicResource LabelCompanyBirth}" Grid.Row="4" FontSize="20" VerticalContentAlignment="Center" HorizontalContentAlignment="Right"/>

<DatePicker x:Name="CompanyBirth" Grid.Row="4" Grid.Column="1" FontSize="20" VerticalAlignment="Center" Height="34" Margin="0,20,0,17">

</DatePicker>

<Label x:Name="LabelCompanyRegDescript" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3" VerticalContentAlignment="Bottom" Margin="70,0,5,0" Height="70" VerticalAlignment="Top">

<TextBlock TextWrapping="Wrap" FontSize="15" Text="{DynamicResource LabelCompanyRegDescript}" Height="65" Width="509"></TextBlock>

</Label>

<Button x:Name="BtnContinueNoReg" Grid.Column="2" Grid.Row="4" Margin="10" Click="BtnContinueNoReg\_Click" BorderBrush="{x:Null}">

<Button.Background>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="#FFF4FAFF"/>

<GradientStop Color="#FFBCEAFF" Offset="0.608"/>

<GradientStop Color="#FF60B4DA" Offset="0.919"/>

</LinearGradientBrush>

</Button.Background>

<TextBlock TextWrapping="Wrap" Text="{DynamicResource BtnContinueNoReg}"></TextBlock>

</Button>

<Button x:Name="BTNBackToMain" Content="{DynamicResource BTNBackToMain}" Grid.Column="3" Grid.Row="4" Margin="10,7,10,10" BorderBrush="{x:Null}">

<Button.Background>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="#FFF4FAFF"/>

<GradientStop Color="#FFBCEAFF" Offset="0.608"/>

<GradientStop Color="#FF60B4DA" Offset="0.919"/>

</LinearGradientBrush>

</Button.Background>

</Button>

<Rectangle Grid.Column="0" Grid.ColumnSpan="4" Grid.Row="6">

<Rectangle.Fill>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="White"/>

<GradientStop Color="#FF031419" Offset="0.989"/>

<GradientStop Color="#FF126A7A" Offset="0.539"/>

<GradientStop Color="#FF579CB6" Offset="0.401"/>

<GradientStop Color="#FF0E6376" Offset="0.644"/>

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

</Grid>

</Window>

public RegistrationOrg()

{

InitializeComponent();

Global.LanguageSwitch(this);

}

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

{

try

{

Global.RegOrgInfo = CompanyName.Text + "/" + CompanyType.Text + "/" + CompanyBirth.SelectedDate.ToString() + "/";

new UserRegOrg().Show();

this.Close();

}

catch

{

}

}

# ПРИЛОЖЕНИЕ В – Структура базы данных

# ПРИЛОЖЕНИЕ Г – Блок-схема работы приложения