**Компьютерная академия «ШАГ»**

**Харьковский филиал**

**Кафедра Разработки программного обеспечения**

**КУРСОВОЙ ПРОЕКТ**

**“Инфраструктура к системе управления проектами”**

**Студентки группы 28ПР11  
Денисенко О.А.**

**Научный руководитель:  
Колтунов А.К.**

**Харьков 2018**

Аннотация

Курсовая работа посвящена решению теоретических, организационных и практических вопросов организации работы по управлению IT проектами, в частности посвящена решению задач связанных с реализацией интерфейса пользователя для десктопной версии и мобильной версии проекта-сайта по управлению проектами (IPlanProject).

Проект реализован на языке С# c использованием технологий Unity 3D межплатформенная среда разработки компьютерных игр, WPF - графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML, Entity Framework — объектно-ориентированная технология доступа к данным. Веб-версия реализована на языке программирования PHP, фреймверк Laravel.

Итогами курсовой работы стали десктопное и мобильные приложения для администрирования сайта IPlanProject, который позволяет организовать взаимодействие между клиентом и исполнителями. Для этого в проекте реализована система обмена данными между тремя версиями проекта: десктопной, мобильной и веб-версией(сайтом).

Данные необходимые для организации информационных потоков сохраняются в базе данных MySQL.

Ключевые слова: Entity Framework , WPF, UI(пользовательский интерфейс), Unity 3D, PHP, Laravel, система управления проектами, база данных MySQL, распределение ролей и прав, Bcrypt.

Содержание

Введение……………………………………………………………………4

Техническое задание………………………………………………………6

1. Выбор технологии реализации проекта………………………………7
2. Разработка структуры системы……………………………………….9
3. Разработка алгоритмов функционирования системы………………12
4. Разработка базы данных для системы……………………………….15
5. Архитектура проекта………………………………………………….19
6. Разработка интерфейса системы…………………………………….22

Выводы……………………………………………………………………29

Список использованной литературы…………………………………...31

**Введение**

Проект реализован в качестве поддержки и расширения возможностей сайта IPlanProject,с помощью которого возможно организовать работы по управлению IT проектами.

Проект разделен на две составляющие. Первая часть проекта – это десктопное приложение для администрирования сайта, реализованное на языке С#, с использованием технологии WPF для построения графического интерфейса и Entity Framework для подключения к базе данных MySQL и создания CRUD-интерфейса. Вторая часть проекта – это мобильное приложение для отображения данных для администрирования сайта, реализованное с помощью Unity 3D и скриптов на языке С#. Основная веб-версия сайта IPlanProject реализована на языке программирования PHP и фреймверке Laravel.

Объектом исследования данной работы является ORM (англ. Object-Relational Mapping, рус. объектно-реляционное отображение, или преобразование) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Предметом исследования является технология Entity Framework и ее взаимодействие с системой управления базами данных MySQL.

Целью данной работы является создание десктопного и мобильного приложений к сайту по управлению проектами(IPlanProject). С помощью этих приложений возможно выполнение задач по администрированию сайта с помощью смартфона (на операционной системе Android) или персонального компьютера (на операционной системе Windows v. 7-10) . А также, целью данного проекта является организация передачи и обмена данными между десктопной, мобильной и веб-версией проекта IPlanProject (сайта, по управлению IT проектами).

Для достижения этой цели необходимо выполнить следующие задания:

Разработать интерфейсы для десктопного и мобильного приложений , подключить необходимые модули для работы, организовать передачу и обмен данными от удаленной базы данных MySQL, реализовать передачу данных в мобильное приложение с помощью JSON - текстовый формат обмена данными, основанный на JavaScript .

**Техническое задание**

Разработать систему администрирования сайта IPlanProject ,с помощью которого возможно организовать работы по управлению IT проектами. Система включает в себя десктопное и мобильное приложение.

Описание работы десктопного приложения:

Администратор должен иметь возможность изменять почту, пароль, имя, менять аватар, добавлять контакты, создавать роли пользователей и добавлять роли пользователям, создать проект и добавлять к нему задачи. При создании задач задаётся время для их исполнения – сроки, описание задачи. Администратор имеет доступ ко всем ресурсам базы данных и обладает неограниченными правами: может все просматривать, создавать, удалять, редактировать. Обмен данными с базой данных организовать с помощью Entity Framework.

Описание работы мобильного приложения:

Администратор имеет право просматривать пользователей и видеть проекты, созданные пользователями. Обмен данными с базой данных организовать с помощью JSON-файлов.

Рабочая среда: Одиночный ПК.

База данных: Удаленная база данных MySQL.

Данные для хранения: пользователи, роли, права, проекты, задачи, контакты, комментарии, технологии, типы контактов, файлы, миграции.

Пользователи - основные роли : Client, Admin, ProjectMan, Programmer, TeamLeader – для веб версии, Admin – для мобильной и десктопной версии.

Информация, поступающая в базу: все данные для хранения хранятся в базе данных.

1. **Выбор технологии реализации проекта**

Для реализации инфраструктуры для системы управления проектами были выбраны: фреймворк : Entity Framework платформы .Net, технология WPF для создания UI , механизмы разбора формата JSON (Unity JSON Serialization и SimpleJSON), Bcrypt — адаптивная криптографическая функция формирования ключа, используемая для защищенного хранения паролей. А UI для платформы Android создан при помощи игрового движка Unity3d.

Кроме того в курсовой работе «Система управления проектами», в которой реализована веб-версия проекта, был использован фреймворк языка PHP Laravel. Laravel это бесплатный PHP фреймворк общего назначения с открытым кодом. Laravel предоставляет множество полезных инструментов для тестирования приложений, использующих базу данных.

Кроме того Структура кода Laravel framework соответствует популярному паттерну проектирования MVC, т.е. в нём можно выделить модели (models), представления (views) и контроллеры (controllers).

Также Laravel совместим с OpenServer, позволяет пользоваться его утилитами - консолью, MySQL.

Laravel позволяет легко работать с базами данных. Позволяет создать структуру базу данных посредством миграций и заполнить тестовыми данными посредством сидеров. Кроме того миграции позволяют изменять структуры таблиц уже после их создания. В Laravel можно прописать отношения между таблицами для легкого получения связанных данных. Также Laravel содержит в себе конструктор запросов, который упрощает написание запросов к базе данных, не используя синтаксис SQl запросов.

Также Laravel содержит модель Eloquent ORM, которая позволяет облегчить программистам работу с БД путём предоставления методов API для типичных операций (выборка, добавление, обновление, удаление и т.д.)

BootStrap - свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения. Bootstrap использует современные наработки в области CSS и HTML . На основе языка HTML написаны шаблоны Blade в проекте. JavaScript - язык JavaScript также был выбран для реализации проекта, с помощью него возможно "оживить" страницы, сделать их более привлекательными для пользователя.

Git позволил ускорить общую работу связанную с написанием кодом, использовать контроль версий для сравнения кода.

**2. Разработка структуры системы**

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища [20].

Центральной концепцией Entity Framework является понятие сущности или entity. Данная технология предполагает работу не с таблицами, а с объектами и их наборами. Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из базы данных [20].

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в базе данных. Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга). На концептуальном уровне происходит определение классов сущностей, используемых в приложении. Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных [20].

Windows Presentation Foundation— система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML [23].

WPF позволяет использовать традиционные языки .NET-платформы - C# и VB.NET для создания логики приложения. Также WPF имеет возможность декларативного определения графического интерфейса с помощью специального языка разметки XAML, основанном на xml и представляющем альтернативу программному созданию графики и элементов управления, а также возможность комбинировать XAML и C#/VB.NET. В WPF все элементы измеряются в независимых от устройства единицах, приложения на WPF легко масштабируются под разные экраны с разным разрешением.

WPF позволяет осуществить привязку данных, использование таких элементов, как стили, шаблоны, темы и др., создание приложений под множество ОС семейства Windows - от Windows XP до Windows 10.

Unity — межплатформенная среда разработки компьютерных игр. Unity позволяет создавать приложения, работающие под более чем 20 различными операционными системами, включающими персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов.Игровой движок Unity3d позволяет создавать кроссплатформенные UI пользователя и осуществляет доступ к ресурсам, например в формате json.

Так как в качестве основной технологии для веб-приложения разработки системы управления проектами был выбран фреймворк Laravel, структура разрабатываемого проекта была построена в соответствии с рекомендациями разработчиков данного фреймворка – использвался паттерн MVC «Модель-Представление-Контроллер» - — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Для десктопного приложения использовалась подобная система организации данных. Доступ к базе данных – данным, осуществлялся с помощью контекста данных, с помощью Entity Framework были построены модели данных – классы и определены взаимосвязи между этими моделями.

Визуальное отображение данных выполнено с помощью Window – WPF, которое использует разметку xaml. Логика взаимодействия между отображаемыми данными и контекстом данных реализована в классе, который унаследован от Window, который сопровождает файлы отображения xaml.

1. **Разработка алгоритмов функционирования системы**

Размещение базы данных для работы приложений выполнено на удаленном сервере. Это позволяет веб-приложению, десктопному приложению и мобильному приложению обращаться к единой базе и хранить данные в одном месте. Изменения данных отправляются в базу данных сразу.

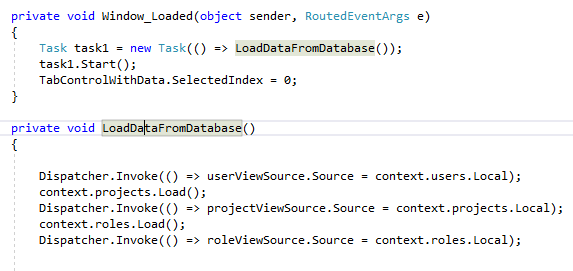
Согласно разработанной структуре системы и жизненного цикла приложения написанного с помощью WPF. Сначала происходит Initialized: это событие возникает при инициализации окна, когда у него устанавливаются все свойства, но до применения к нему стилей и привязки данных. Это общее событие для всех элементов управления в WPF, поэтому следует учитывать, что сначала возникают события вложенных элементов, а затем их контейнеров. То есть событие Initialized окна приложения генерируется только после того, как отработает событие Initialized для всех вложенных элементов.

При создании главного окна определяется контекст данных – DataContext, загружаются данные о пользователях, так как нам необходимо проверить логи и пароль пользователя для входа пользователя в систему.

Далее Loaded: возникает после полной инициализации окна и применения к нему стилей и привязки данных. После генерации этого события происходит визуализация элемента, и окно отображается на экране и становится видимым для пользователя.

По наступлению этого события отдельным потоком начинают подгружаться остальные необходимые данные в контекст данных, а также по мере подгружения – передаются для отображаются на экране.

Эти данные подгружаются с помощью отдельного потока – пока пользователь вводит данные для входа в систему. Такая система загрузки данных со вторым потоком позволила ускорить загрузку окна и быстрее отобразить пользователю необходимые данные, смотри рисунок 1:

Рис.1 Обработчик события на загруженное окно и метод, загружающий данные в контекст данных.

У главного окна к некоторым элементам применены стили на основе триггеров событий, также применены к некоторым компонентам CommandBindings, объявлены Resource. При работе с окном действия пользователя обрабатывают триггеры событий, а также обработчики событий. Например, при нажатии на кнопки перехода по таблице вызываются следующие обработчики событий (рисунок 2):

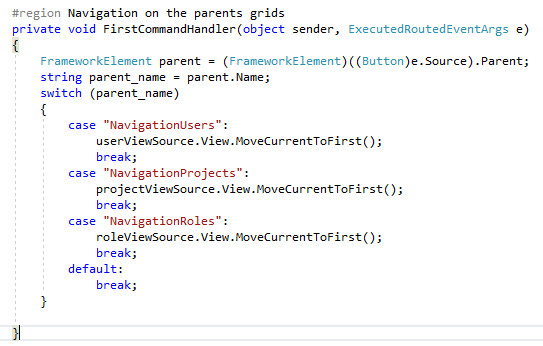


Рис. 2 Обработчик события нажатия навигационной кнопки «Вперед»

Далее происходит определение на какой таблице был вызван этот обработчик событий и происходит перемещение по необходимой таблице. При нажатии на кнопки, например “New Role”, вызывается следующий обработчик событий (рисунок 3):

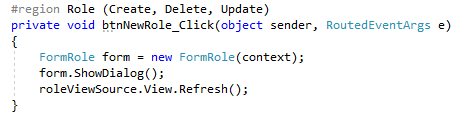


Рис. 3 Обработчик событий на нажатие кнопки “New Role”

При выполнении этого обработчика события создается новое окно и открывается. Новое окно содержит форму для редактирования свойств объекта “role”. При этом в контекст данных нового окна передается новый созданный объект “role”. При сохранении изменений в новом окне – для создания новой роли, необходимо нажать кнопку “Save”. При нажатии этой кнопки вызовется обработчик события (рисунок 4):

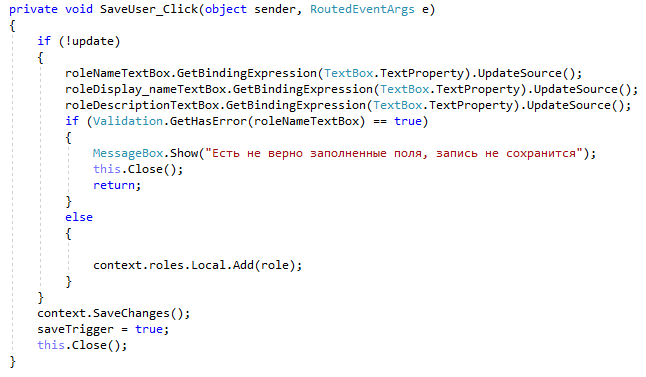


Рис. 4 Обработчик события на нажатие кнопки “Save”

При выполнении этого обработчика событий проверяется, если это создание новой записи, то сначала обновляются все свойства, которые добавлялись, далее объект проверяется на наличие ошибок валидации. Если есть неверно заполненные поля – запись не сохраниться. Если все поля заполнены верно, то запись будет добавлена сначала в локальный контекст данных, а далее командой для контекста SaveChanges() – данные будут переданы на сервер.

**4. Разработка базы данных для системы**

Для функционирования системы необходимо хранить данные о:

* пользователях;
* ролях пользователей;
* разрешениях – права доступа для определенной роли;
* контактах пользователей;
* проектах;
* задачах проектов;
* файлах для задач;
* технологиях (пользователя, проекта, задачи) ;
* комментариях к задачам.

В соответствии с этим были разработаны таблицы данных и взаимосвязи между таблицами.

Разработана таблица “Users”, которая хранит основные данные о пользователе. Каждый пользователей может иметь несколько типов контактов – организовываем связь один ко многим, к таблице “Contacts”.

Каждый контакт хранит свой тип данных Contact\_type (почта, телефон, Viber и т.д.), поэтому организовываем связь между таблицами “Contacts” “Contact\_type” один ко многим, т.к. один тип контакта может иметь несколько вариантов контактов.

Каждый пользователь может иметь несколько ролей (таблица “Roles”), и каждая роль может содержать в себе несколько пользователей, поэтому создаем связь многие ко многим с помощью вспомогательной таблицы “role\_user”. Аналогичная взаимосвязь многие ко многим между ролями и правами пользователя – создаем таблицы “Permissions” и “Permission\_Role”.

Каждый проект может иметь только одного клиента, поэтому связь между таблицами “Users” и “Projects” по полю client\_id – один ко многим, так как у одного клиента может быть несколько проектов. В тоже время к проектам можно добавлять пользователей, с ролями разработчиков, проджект-менеджеров. Поэтому создаем также связь многие ко многим (много проектов у разработчика – много разработчиков в проекте) – вспомогательная таблица “project\_user”.

Связь многие ко многим у таблиц технологий “Technologies” – c проектами, пользователями и задачами.

Один проект может содержать в себе несколько задач, поэтому создаем связь один ко многим и таблицу “Tasks”. Задача может содержать в себе несколько файлов, поэтому также создаем связь один ко многим , таблицу “Task file”.

Задача может иметь несколько комментариев – связь один ко многим, а также комментарий может иметь – автора - пользователя, а пользователь может оставлять много комментариев – также связь один ко многим.

Разработка и разворачивание базы данных осуществлено с помощью механизмов фреймворка Laravel. Laravel позволил разработать модели данных, прописать взаимосвязи между данными – определить primary key и foreign key в таблицах. База данных была развернута на системе управления базами MySQL c помощью миграций баз данных и сидеров данных.

В итоге были разработаны в первоначальном виде таблицы указанные на рисунке 6.

Для разработки десктопной версии проекта – необходимо подключение к базе данных и получение модели базы данных на основе уже существующей базы данных. Подключение и получение контекста данных выполнено с помощью Entity Framework - Database First. В итоге автоматически были созданы классы и взаимосвязи между классами данных.

А также был создан контекст данных – показанный на рисунке 5.

Автоматически сгенерированные классы моделей данных содержат в себе виртуальные свойства – указывающие на связи с другими таблицами. Виртуальное свойство означает ленивую загрузку данных. Ленивая загрузка - это процесс, при котором объект или совокупность объектов автоматически загружается из базы данных при первом доступе к объекту, относящемуся к объекту / объектам.

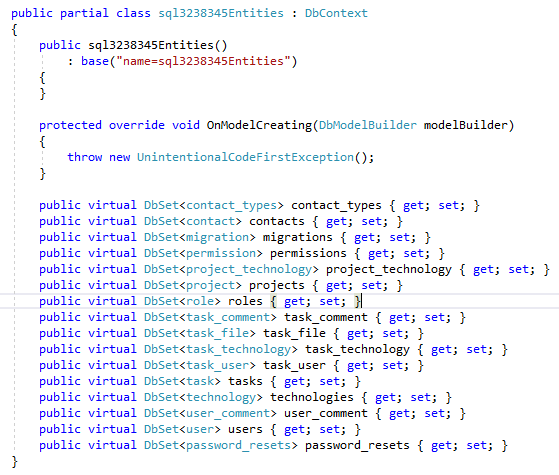


Рис.5 Класс - контекст данных

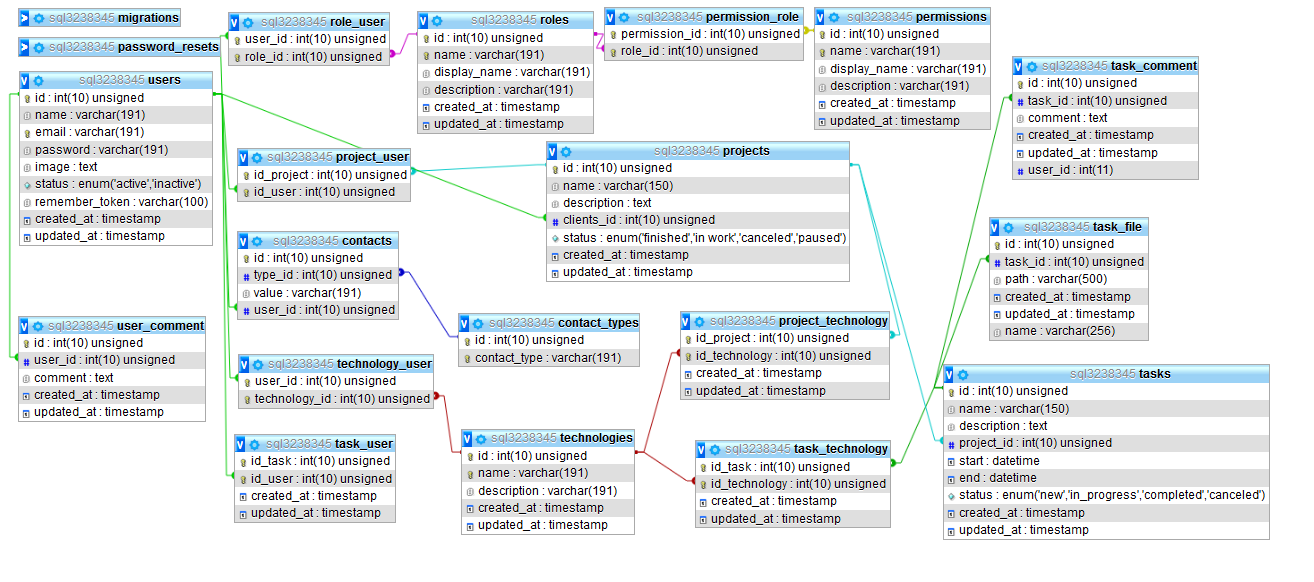


Рис.6 Структура базы данных MySQL

1. **Архитектура проекта**

Для реализации технического задания для десктопного проекта была разработана USE CASE диаграмма, показанная на рисунке 7. Данная диаграмма отображает возможности взаимодействия Админа с программой.

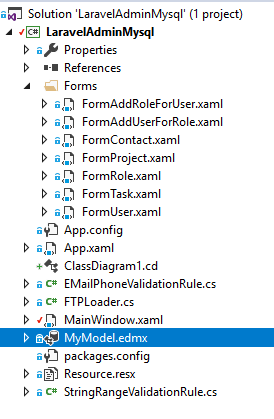
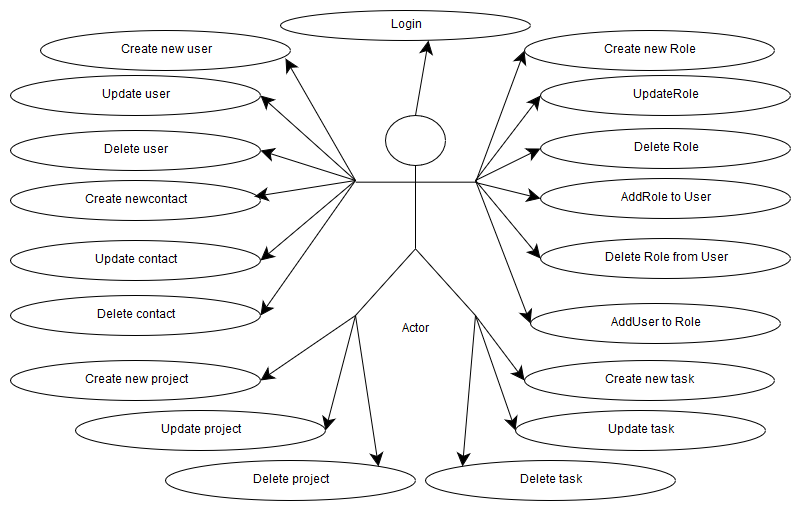
****

Рис.7 Use Case диаграмма для пользователя-Админа, десктопного проекта

Рис.8 Структура декстопного проекта – Solution Exploer

В ходе внедрения Use Case диаграммы была создана структура проекта, показанная на рисунке 8.

Для работы с базой данных MySQL был установлен коннектор – провайдер и были подключены следующие библиотеки: MySql.Data.dll;

MySql.Data.Entity.EF6.dll; MySql.Web. Для работы с Entity Framework, через NuGet Package Manager был установлен EntityFramework 6.2.

Для работы с зашифрованными паролями через NuGet Package Manager был установлен BCrypt.Net-Next 2.1.3.

В App.Config – прописан провайдер поключения и строка подключения к удаленной базе данных MySQL

Основное окно проекта – Main Window – содержит в себе таблицы данных. Диалоговые окна – Form\*.xaml – для добавления – редактирования данных таблиц.

FTPLoader – для загрузки и получения файлов удаленного сервера с помощью FTP соединений.

Правила валидации для заполнения форм данных прописаны в \*ValidationRule.cs.

MyModel.edmx – хранит автоматически сгенерированный контекст данных. Модель базы данных отображена на рисунке 10 .

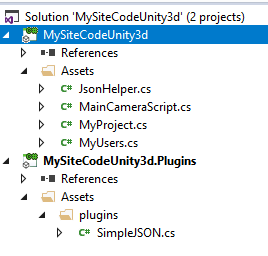


Рис. 9 Структура скриптов для мобильного проекта.

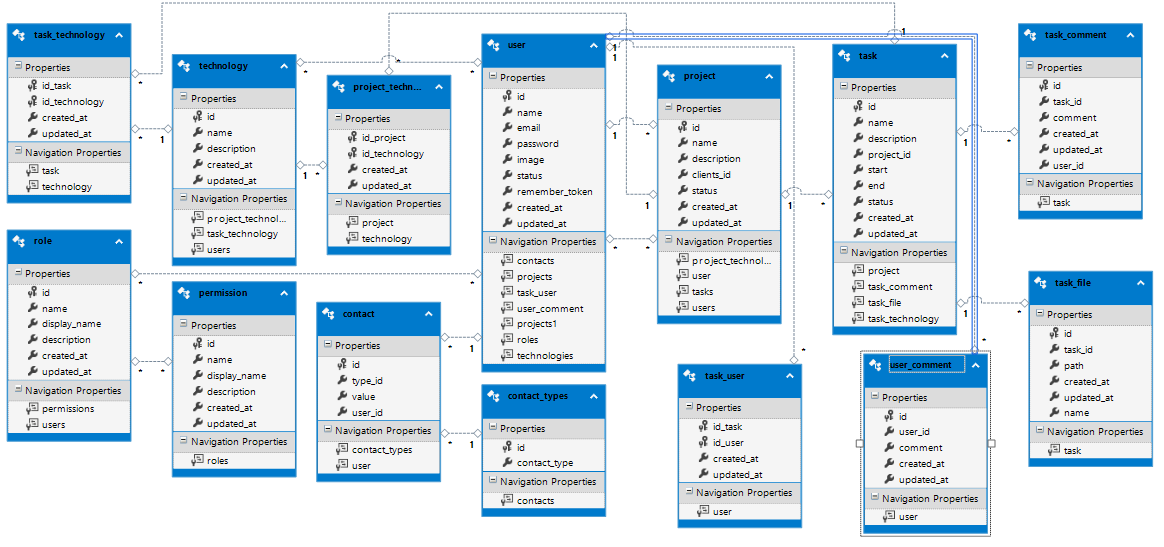


Рис.10 Модель базы данных

**6. Разработка интерфейса системы**

Для входа в десктопное приложение и доступа к данным необходимо ввести логин - почтовый адрес и пароль. Если пользователь обладает правами администратора, то ему будут доступны данные для просмотра и редактирования, смотри рисунок 11. Если пользователь не обладает правами администратора, ему будет выведено сообщение о том, у него нет прав доступа и, что необходимо обратиться к администратору. Если комбинация логина-пароля не найдена в базе данных, будет выдано сообщение об ошибке.

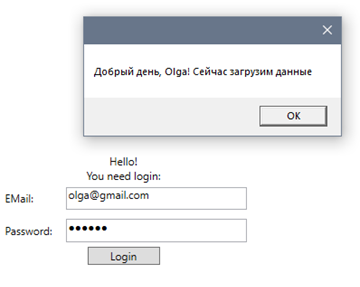


Рис.11 Логин – получение доступа к данным

Страница, на которую вы попадаете сразу после входа в аккаунт показана на рисунке 12. На этой странице находится таблица пользователей с данными о пользователе. Внизу таблицы отображается детальная информация о контактах пользователя. Можно использовать навигационные кнопки для перемещения. Можно добавлять нового пользователя – кнопка “New User”, обновлять данные кнопка “Update” возле строки пользователя, удалять пользователя - кнопка “Delete”. Также можно добавить контакты к выделенному пользователю, нажав кнопку “New Contact”, .обновить данные и удалить контакт, нажав кнопки “Update” и “Delete” соответственно.

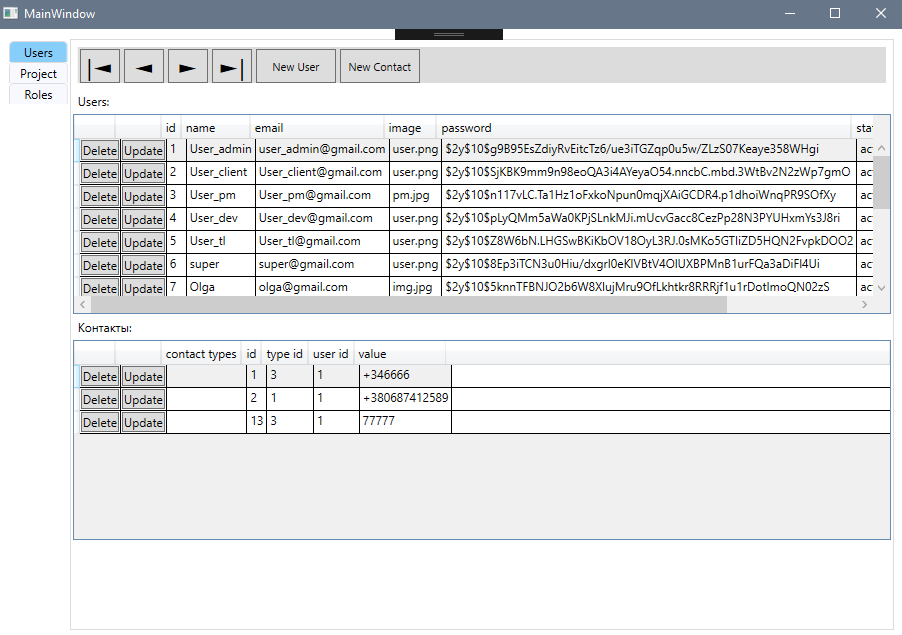


Рис.12 Таблица пользователей и контакты выбранного пользователя

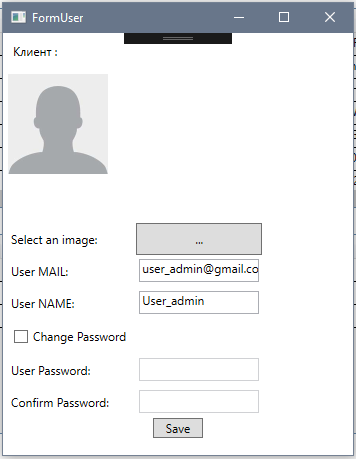


Рис.13 Форма добавления – редактирования профиля пользователя

Форма добавления/редактирования пользователя отображена на рисунке 13. Данная форма позволяет обновить данные пользователя – загрузить новый аватар, обновить пароль у сохраненного пользователя. Для полей включена валидация данных, поэтому, если поле заполнено не верно будет выведено сообщение об ошибке. При нажатии на кнопку сохранить проверка на корректность данных будет проведена еще раз, и если есть некорректно заполненные поля, то записи не сохраняться. Во всех формах приложения реализована такая схема проверки данных на корректность и сохранение изменений.

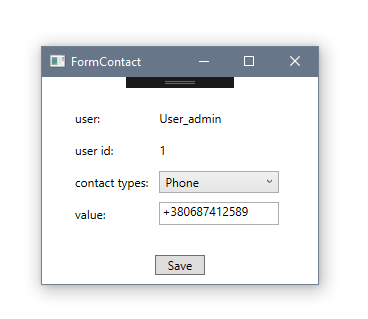
Форма добавления/редактирования контакта пользователя показана на рисунке 14. Выбор типа контакта осуществляется с помощью Combobox. 

Рис.14 Форма добавления – редактирования контакта пользователя

Вкладки “Users” “Project” “Roles” в левом верхнем углу позволяют осуществить переход между таблицами данных.

На рисунке 15 показана таблица проектов, с детализацией – дополнительной таблицей задач, относящихся к выбранному проекту. Доступный интерфейс аналогичный интерфейсу с таблицей пользователей.

Есть возможность добавить проект, удалить проект, обновить проект. Также есть возможность добавить задачу к выбранному проекту, удалить задачу, обновить задачу.

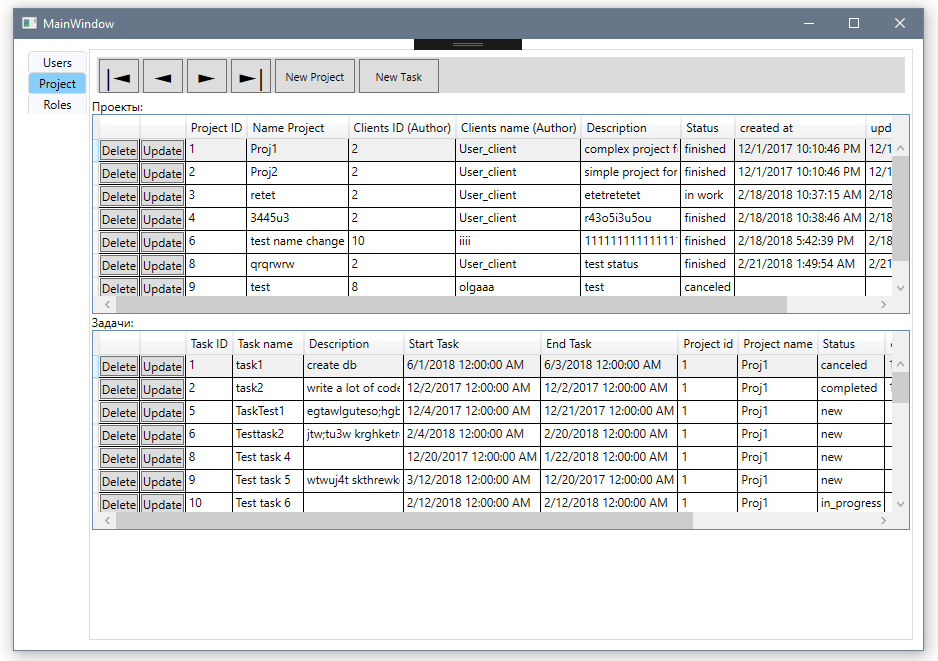


Рис.15 Таблица проектов и задач к выбранному проекту.

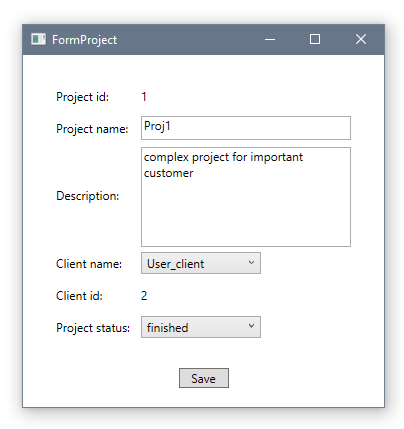


Рис.16 Форма добавления – редактирования проекта

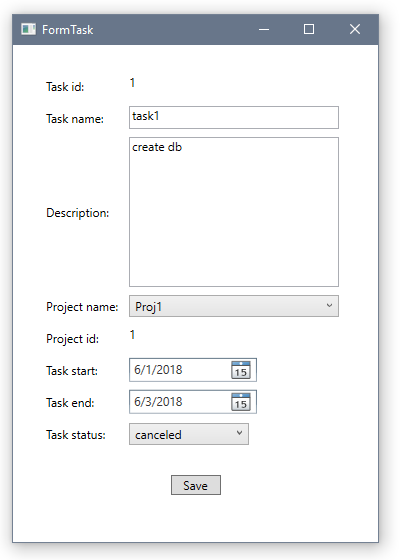


Рис.17 Форма добавления – редактирования задачи

На рисунке 17 показана таблица ролей, с детализацией – дополнительной таблицей пользователей, которые обладают такой ролью. Доступный интерфейс аналогичный интерфейсу с таблицей пользователей.

Есть возможность добавить роль, удалить роль, обновить роль. Также есть возможность добавить роль к выбранному пользователю, удалить роль пользователя, добавить пользователя к определенной роли.

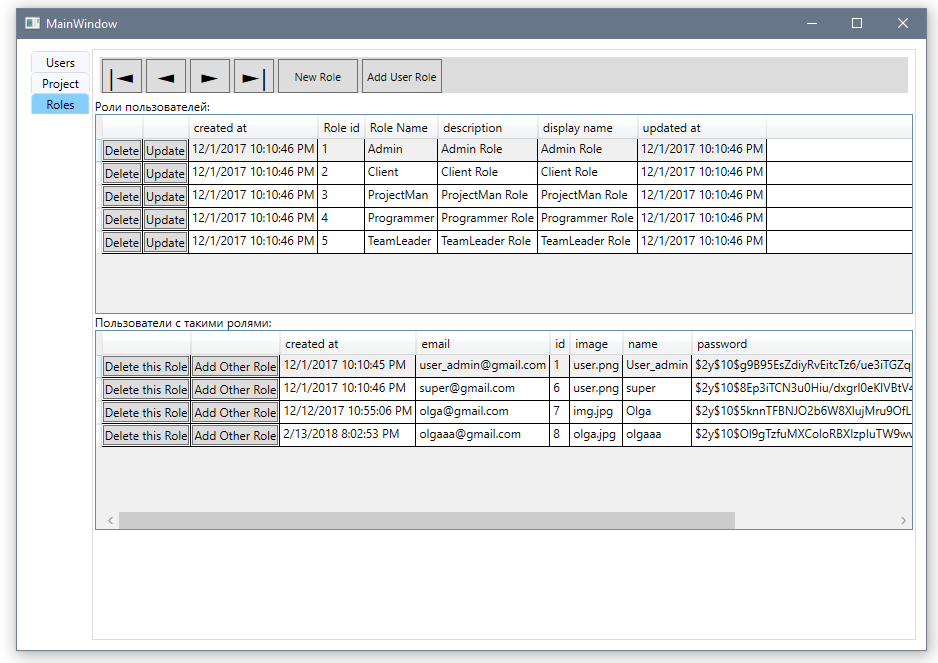


Рис.18 Таблица ролей пользователей и пользователей имеющих выбранную роль

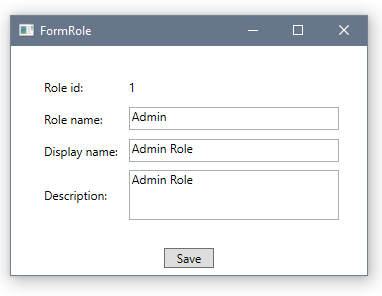


Рис.19 Форма добавления – редактирования роли пользователя

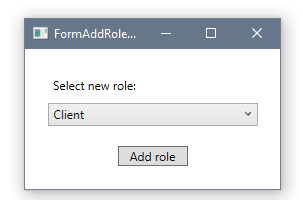


Рис.20 Форма добавления роли для пользователя

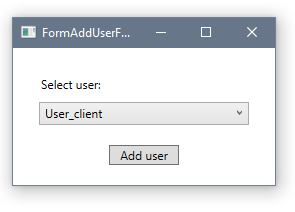


Рис.21 Форма добавления пользователя к определенной роли.

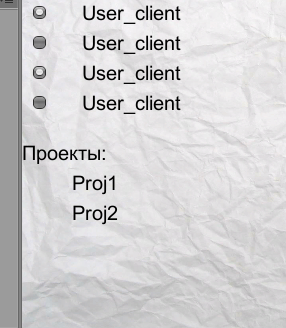


Рис 22. Интерфейс мобильной версии приложения

На рисунке 22 показан интерфейс мобильного приложения – в верхней части идет список клиентов-пользователей. При выборе клиента внизу отображается список проектов – этих пользователей.

**Выводы**

В итоговом проекте мы получили сайт, который позволяет организовать работу по управлению проектами и десктопное приложение, которое позволяет администрировать сайт. А также мобильное приложение, которое отображает статистическую информацию о пользователях и их проектах.

Основными технологиями для написания десктопного приложения стал Entity Framework - для организации работы с базой данных, и WPF - для построения графических интерфейсов, а также технология BCript.Net-Next - адаптивная криптографическая функция формирования ключа, используемая для защищенного хранения паролей. Основной технологией для мобильного приложения на Android, стал игровой движок Unity3d – для интерфейса. Логика работы мобильного приложения построена с помощью скриптов на языке с#. Основной технологией для реализации сайта была использована технология Laravel Framework. С помощью этой технологии удалось реализовать принцип MVC - выделены модели, контроллеры и представления.

База данных MySQL была разработана и развернута на удаленном сервере баз данных. Прописаны взаимосвязи между таблицами. Организован обмен данными с удаленной базой данных и десктопным, и мобильным приложениями с помощью Connector/Net. Connector/Net - это полностью управляемый драйвер ADO.NET для MySQL. Для обмена данными с мобильным приложением использованы JSON-файлы и механизмы разбора формата JSON : Unity JSON Serialization и SimpleJSON.

Веб-приложение-сайт – выложен на бесплатный хостинг. Организована загрузка файлов данных (таких как аватар пользователя) с помощью методов протокола FTP, которые могут использоваться с FTP-запроса.

В проекте реализован подход, в котором права пользователя распределяются в зависимости от того, какой ролью обладает пользователь. Выделено шесть основных ролей Гость, Клиент, Администратор, Программист, ТимЛид, Проджект-Менеджер. Кроме того реализован механизм добавления и удаления ролей, назначения ролей пользователям.

Для входа и доступа к веб-приложению сайту реализован механизм регистрации нового пользователя и логина существующего пользователя. В ходе отправки на сервер данных используется csrf\_field для верификации пользователя и чтобы не подделывались запросы.

После того как пользователь “залогинился” на сайте, ему открывается доступ к основному функционалу сайта, в зависимости от роли, которой обладает пользователь.

Доступ в десктопном приложении к данным разрешен только пользователям, имеющим роль “Админ”, остальным зарегистрированным пользователям рекомендуется обратиться к администратору. Для “залогинивания” пользователя используется почта и пароль. Администратор обладает неограниченными правами. Админ может добавлять пользователей, менять данные пользователя: смена фото-аватара, смена пароля, добавления контактов, изменения имени пользователя, изменение почты пользователя. Также админ может создавать проекты, закреплять за ними нового клиента, добавлять к проектам задачи, редактировать данные проекта и задач, просматривать статус проектов и задач. Админ имеет возможность создать новые роли пользователей, удалять роли пользователей, распределять роли между пользователями.

**Список использованной литературы**

1. Laravel по-русски [Электронный ресурс] // Режим доступа: https://laravel.ru
2. LARAVEL.SU [Электронный ресурс] // Режим доступа: laravel.su
3. Stackoverflow [Электронный ресурс] // Режим доступа: https://stackoverflow.com
4. W3Schools.com [Электронный ресурс] // Режим доступа: https://www.w3schools.com
5. habr [Электронный ресурс] // Режим доступа: https://habr.com/
6. Laracasts [Электронный ресурс] // Режим доступа: https://laracasts.com
7. laravel [Электронный ресурс] // Режим доступа: https://laravel.com/
8. Хостинг на сайте [Электронный ресурс] // Режим доступа: https://www.000webhost.com, сайт: https://ssamyraika.000webhostapp.com
9. CRUD [Электронный ресурс] // Вікіпедія. Вільна енциклопедія. – Режим доступа: https://uk.wikipedia.org/wiki/CRUD
10. Руководство по WPF [Электронный ресурс] //Metanit.com Сайт о программировании. – Режим доступа: https://metanit.com/sharp/wpf/
11. MySQL Documentation[Электронный ресурс] // MySQL . – Режим доступа: https://dev.mysql.com/doc/
12. Entity Framework [Электронный ресурс] //Microsoft. Developer Network. – Режим доступа:https://msdn.microsoft.com/en-us/library/gg696 172%28v=vs.103%29.aspx?f=255&MSPPError=-2147217396
13. Bcrypt [Электронный ресурс] //Википедия. Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Bcrypt
14. Unity Documentation [Электронный ресурс] //Режим доступа: https://docs.unity3d.com/Manual/index.html
15. SimpleJSON [Электронный ресурс] //Unify Community” . – Режим доступа: http://wiki.unity3d.com/index.php/SimpleJSON#Download
16. Parse JSON data in unity3d [Электронный ресурс] // Gyanendu shekhar’s Blog”. – Режим доступа: http://gyanendushekhar.com/2016/06/15 /parse-json-data-in-unity3d/
17. ADO.NET Entity Framework [Электронный ресурс]// Википедия. Свободная энциклопедия. – Режим доступа: https://ru.wikipedia. org/wiki/ADO.NET \_Entity\_Framework
18. Introduction to Entity Framework [Электронный ресурс]//Microsoft. Developer Network. – Режим доступа: https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx
19. Entity Framework 6.0 Introduction: [Электронный ресурс]// Entity Framework Tutorial– Режим доступа: http://www.entityframeworktutorial.net/entityframework6/introduction.aspx
20. Введение в Entity Framework [Электронный ресурс]//Metanit.com Сайт о программировании. – Режим доступа: https://metanit.com/sharp/entityframework/1.1.php
21. Взаимодействие с данными. Подходы [Электронный ресурс]//Metanit.com Сайт о программировании. – Режим доступа: https://metanit.com/sharp/entityframework/2.2.php
22. EntityFrameworkMrf6\_0 [Электронный ресурс]//Microsoft. Developer Network. – Режим доступа: https://msdn.microsoft.com/en-us/library/mt154346(v=vs.113).aspx
23. Windows Presentation Foundation [Электронный ресурс]// Википедия. Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Windows\_Presentation\_Foundation
24. Unity (игровой движок) [Электронный ресурс]// Википедия. Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Unity\_(%D0%B8%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9\_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)