Частное учреждение образование

«Колледж бизнеса и права»

ОТЧЕТ по ПРАКТИКЕ

по РАЗРАБОТКЕ и сопровождению программного обеспечения

ОП Т.616011

Выполнил Е.В. Дубровский

Руководитель практики Н.В. Ржеутская

2018

Содержание

1 Программа практики4

2 Объектно-ориентированный анализ и проектирование системы5

2.1 Сущность задачи5

2.2 Проектирование модели5

3 Вычислительная система10

3.1 Требования к аппаратным и операционным ресурсам10

3.2 Инструменты разработки10

4 Проектирование задачи12

4.1 Требования к приложению12

4.2 Концептуальный прототип13

4.3 Организация данных14

4.4 Функции и элементы управления15

4.5 Проектирование справочной системы18

5 Описание программного средства19

5.1 Общие сведения19

5.2 Функциональное назначение19

5.3 Входные и выходные данные19

6 Методика испытаний20

6.1 Технические требования20

6.2 Функциональное назначение20

7 Применение23

7.1 Назначение программы23

7.2 Условия применения23

7.3 Справочная система23

Заключение24

Список используемых источников25

Приложение А26

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

ОП Т.616011.401 ПЗ

Разраб.

*Дубровский Е. В.*

Провер.

Ржеусткая Н.В.

Т. Контр.

Н. Контр.

Утверд.

Отчет по практике

по разработке и сопровождению

Лит.

Листов

37

КБП

## 1 Программа практики

Целями практики по разработке и сопровождению программного обеспечения являются:

- закрепление знаний, связанных с технологией обработки информации;

- приобретение навыков разработки программ с использованием современных средств обработки экономической и деловой информации;

- выработка умений применять средства стандартных библиотек в практических задачах.

Задачами практики по разработке и сопровождению ПО являются:

- углубленное изучение принципов организации программного обеспечения и технологии его проектирования;

- владение методами надежного программирования;

- умение разрабатывать программы в соответствии с промышленными требованиями, обеспечивая высокий уровень качества программного обеспечения и экономической эффективности;

- изучение способов определения экономической себестоимости и эффективности разработки программного обеспечения;

- оформление комплекта документации на созданное программное обеспечение.

##### Таблица 1 - Календарный график работы

|  |  |
| --- | --- |
| Период | Наименование и содержание работ |
| 09.02.18 | Изучение требований к рабочей дисциплине и технике безопасности.  Изучение нормативных документов. |
| 11.02.18-  17.02.18 | Ознакомление с различным программным обеспечением, используемым для разработки программ |
| 18.02.18-  24.02.18 | Получение индивидуальных заданий по созданию программ для обработки экономической и деловой информации. Обсуждение тем индивидуальных заданий. |
| 25.02.18-  02.03.18 | Изучение общих требований к программному обеспечению обработки экономической и деловой информации. Требования к интерфейсу пользователя, выходным документам (печатные, экранные формы, форматы результирующих файлов с результатами расчетов). |
| 03.03.18-  23.03.18 | Анализ предметной области индивидуального задания. Выбор и обоснование языка и среды программирования. Описание функциональных возможностей задачи. |
| 24.03.18-  06.04.18 | Технология создания программного обеспечения. Правила построения диаграмм UML. Проектирование ПО при помощи CASE-средства Rational Rose |
| 07.04.18-  20.04.18 | Конструирование алгоритма и его реализация в выбранной визуальной среде программирования. Разработка интерфейса .Проектирование классов, определение свойств и методов, определение связей между классами. |
| 21.04.18-  04.05.18 | Создание информационной базы задачи. Тестирование и отладка программы с использованием подготовленных тестовых примеров. Проектирование модулей, позволяющих осуществить защиту программного продукта от несанкционированного доступа.. |
| 05.05.18-  25.05.18 | Критерии оценки качества программного продукта. Изучение нормативных документов, регламентирующих оформление технического проекта. Создание документации по эксплуатации и сопровождению разработанного ПС. Проведение испытания ПС в нормальных и критических условиях.. |
| 26.05.18-  01.06.18 | Анализ возможности модернизации проекта с учетом альтернативных решений. Оформление аннотации на программное средство. |
| 02.06.18-  15.06.18 | Оформление отчета по практике. Подготовка листингов программ с дополнением комментариев. Распечатка экранных форм. |
| 16.06.18-  21.06.18 | Демонстрация программ по индивидуальным заданиям.  Защита отчета по практике. |

2 Объектно-ориентированный анализ и проектирование системы

2.1 Сущность задачи

У каждого из нас есть проблемы с памятью – можно пойти в магазин и забыть купить какой-то продукт или забыть напрочь о том, то завтра нужно сдавать нарисованный дизайн сайта. Конечно, можно писать в блокноте записи о важных задачах в жизни, а можно просто зайти на сайт в интернете и делать заметки там – это более экономично и просто, ведь не нужно постоянно покупать новые блокноты, таскать с собой дополнительную бумагу и ручку.

Целью разработки проекта на тему «Программная реализация органайзера» является создание программы, которая будет предоставлять пользователю возможность записи текстовой, графической и иного рода информации в виде заметок на сайте.

На данный момент существует много аналогов данного приложения, например: https://listick.ru, https://support.apple.com/ru-ru/ht205773, а также https://evernote.com/intl/ru

2.2 Проектирование модели

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями. Наиболее распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD), которые предназначены для графического представления моделей данных разрабатываемой программной системы и предлагают некоторый набор стандартных обозначений для определения данных и отношений между ними. С помощью этого вида диаграмм можно описать отдельные компоненты концептуальной модели данных и совокупность взаимосвязей между ними, имеющих важное значение для разрабатываемой системы. Основными понятиями данной нотации являются понятия сущности и связи. При этом под сущностью понимается произвольное множество реальных или абстрактных объектов, каждый из которых обладает одинаковыми свойствами и характеристиками. В этом случае каждый рассматриваемый объект может являться экземпляром одной и только одной сущности, должен иметь уникальное имя или идентификатор, а также отличаться от других экземпляров данной сущности. Связь определяется как отношение или некоторая ассоциация между отдельными сущностями. Примерами связей могут являться родственные отношения типа «отец-сын» или производственные отношения типа «начальник-подчиненный». Другой тип связей задается отношениями «иметь в собственности» или «обладать свойством».

Графическая модель данных строится таким образом, чтобы связи между отдельными сущностями отражали не только семантический характер соответствующего отношения, но и дополнительные аспекты обязательности связей, а также кратность участвующих в данных отношениях экземпляров сущностей. Информационная модель базы представлена на диаграмме «Сущность-связь». Данная диаграмма представлена на рисунке 2.1.

Рисунок 2.1 - Диаграмма «Cущность-связь» в нотации Баркера

**Пользователь**  
\*Главный email

\*Email для подтв. Пароля

\*Пароль

\*Секретный ключ

**Заметка**  
\*Заголовок

\*Описание

\*Дата

Пользователь

Заметка

Составляет

Пользователь

Заметка

Составляет

Рисунок 2.2 - Диаграмма ERD в нотации Ричарда Чена

Исходя из исследования предметной области, можно выделить следующие сущности разработки: пользователь и заметка.

Для сущности «Пользователь» можно выделить следующие атрибуты:

- Главный email

- Email для подтв. пароля

- Пароль

- Секретный ключ

Для сущности «Заметка» можно выделить следующие атрибуты:

- Заголовок

- Описание

- Дата

В рамках языка UML все представления о модели сложной системы фиксируются в виде специальных графических конструкций – диаграмм. В терминах языка UML определены следующие виды диаграмм: диаграмма вариантов использования, диаграмма классов, диаграмма деятельности, диаграмма последовательности, диаграмма компонентов.

Перечень этих диаграмм представляет собой неотъемлемую часть графической нотации языка UML, сам процесс объектно-ориентированного программирования (ООП) неразрывно связан с процессом построения этих диаграмм. Совокупность построенных таким образом диаграмм содержит всю информацию, необходимую для реализации проекта сложной системы.

Диаграмма – граф специального вида, состоящий из вершин в форме геометрических фигур, которые связаны между собой ребрами или дугами.

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы.

В данной проектируемой системе в качестве актера выступает пользователь, которому необходима производить crud операции с заметками.

К основным функциям разрабатываемой программы относятся следующие функции:

* добавление заметки;
* просмотр информации заметке;
* редактирование заметки;
* удаление заметки*;*
* небольшой текстовый редактор для описания заметки

Вспомогательных функция данная программа не имеет.

Диаграмма вариантов использования представлена на листе 1 графической части.

Центральное место в ООП занимает разработка логической модели системы в виде диаграммы классов. Нотация классов в языке UML проста и интуитивно понятна. Схожая нотация применяется и для объектов — экземпляров класса, с тем различием, что к имени класса добавляется имя объекта, и вся надпись подчеркивается.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Диаграмма классов представляет собой некоторый граф, вершинами кото­рого являются элементы типа «классификатор» и которые связаны различными типами структурных отношений. Следует заметить, что диаграмма классов может также содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры, такие как объекты и связи. Когда говорят о данной диаграмме, имеют в виду статическую структурную модель проектируемой системы. Поэтому диаграмму классов принято считать графическим представлением таких структурных взаимосвязей логической модели системы, которые не зависят от времени.

Класс в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линями на разделы или секции. Здесь указывается: имя класса, атрибуты и операции (методы).

Обязательным элементов обозначения класса является его имя. На начальных этапах разработки диаграммы отдельный класс может обозначаться простым прямоугольником с указанием только имени соответствующего класса. По мере проработки отдельных компонентов диаграммы классов дополняются атрибутами и операциями. Предполагается, что окончательный вариант диаграммы содержит наиболее полное описание классов, которые состоят из трех разделов или секций. Состав некоторых основных классов проекта представлен в таблице 2.1. Диаграмма классов представлена на листе 2 графической части.

Таблица 2.1 – Состав некоторых основных классов проекта

|  |  |  |
| --- | --- | --- |
| Имя класса | Имя поля | Методы |
| NoteController | db | Index; Add; GetNote; Edit; Delete; |
| UserController | db | Exit; Home; IsRestoringEmail; IsMainEmail; IsSecretKeyByMainEmail; IsPassByMainEmail; IsValidRecapcha; Registration; GetIdByMainEmail; ConfReg; Login; SendMesForPassRec; PassRec; SaveNewPass; |
| IsData |  | IsRestoringEmail; IsMainEmail; IsSecretKeyByMainEmail; IsPassByMainEmail; |
| DBContext | Notes; Users; |  |

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, переход в следующее состояние срабатывает только при завершении этой операции. Графически диаграмма деятельности представляется в форме графа, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому.

Диаграммы деятельности - частный случай диаграмм состояний. Основная цель использования таких диаграмм - визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности представлена на листе 3 графической части.

Для моделирования взаимодействия объектов в UML используется диаграмма последовательности. Диаграмма последовательности представлена на листе 4 графической части.

Все предыдущие диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления и оперировали понятиями, которые не имеют самостоятельного материального воплощения. Для описания реальных сущностей предназначен другой аспект модельного представления, а именно физическое представление модели. Для физического представления модели используются диаграмма компонентов.

Диаграмма компонентов описывает объекты реального мира – компоненты программного обеспечения. Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код.

Диаграмма компонентов представлена на листе 5 графической части и содержит следующие компоненты:

* программные компоненты, созданные в среде VStudio 2017;
* файл базы данных «MyDatabaseForTesting.mdf»;

3 Вычислительная система

3.1 Требования к аппаратным и операционным ресурсам

Конфигурация компьютера, на котором будет разрабатываться программное приложение:

- процессор Intel Core i5 6300 3,2 Ггц;

- оперативная память DDR4 8 Гб;

- жёсткий диск HDD Samsung 500 Гб;

- видеокарта NVIDIA GeForce GTX 950M 512 Мб;

- материнская плата Asus;

- монитор Samsung 22”;

- клавиатура A4Tech Bloody B2100;

- мышь A4Tech Bloody V8.

3.2 Инструменты разработки

Инструментами разработки будут являться:

* операционная система Windows 10;
* среда Microsoft Visual C# 2017;
* язык программирования С#;
* язык программирования JS
* язык текстовой разметки HTML 5
* язык стилизации текстовой разметки CSS 3
* программа для построения диаграмм UMLet;
* база данных MySQL;

Операционная система – это набор управляющих программ, предназначенных для управления ресурсами вычислительной системы как единого комплекса, другими словами операционная система – это набор программного обеспечения, который обеспечивает работу компьютера. Основными функциями операционной системы являются:

1. Управление файловой системы

2. Управление выполнением программ

3. Управление памятью

4. Диалог с пользователем

При разработке программного средства использовалась операционная система Windows 10, так как эта система является самой распространённой системой. В Windows 10 были исправлены практически все недостатки предыдущих операционных систем. Так же аппаратные требования 10-ки скромнее, она способна работать на маломощных ПК. [18]

Microsoft Visual Studio 2017 — Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight. Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server). [15]

C# (произносится си шарп) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .Net Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с С-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщенные типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет C#, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на VB.NET, J#. [13]

UMLet – это инструмент UML с открытым исходным кодом на основе Java, предназначенный для обучения унифицированному языку моделирования и для быстрого создания диаграмм UML. Это инструмент рисования, а не инструмент моделирования, поскольку нет базового словаря или каталога объектов многократного использования.

UMLet может экспортировать диаграммы в изображения (eps, jpg), форматы чертежа (SVG), форматы документов (PDF). Буфер обмена можно использовать для копирования-вставки диаграммы в виде изображений в другие приложения. Можно создавать пользовательские элементы UML. [17]

База данных MySQL – система хранения и обработки данных, для доступа к котором используется язык SQL. Существует множество различных систем управления базами данных, но в условиях хостинга как правило применяется СУБД MySQL. Причинами тому являются фактическая ориентация этой СУБД на хостинговые задачи, доступность на всех популярных серверных операционных системах, а также простота настройки и администрирования. [16]

JavaScript – это язык программирования, который добавляет интерактивность на ваш веб-сайт (например: игры, отклик при нажатии кнопок или при вводе данных в формы, динамические стили, анимация).  [18]

HTML5 — последняя версия стандарта HTML. Термин имеет два определения:

Новая версия языка HTML, с новыми элементами, атрибутами и новым поведением.

Набор технологий, позволяющий создавать разнообразные сайты и Web-приложения.

[19]

CSS3 это последнее эволюционное изменение языка Cascading Style Sheets, и оно направлено на расширение CSS2.1. Оно привносит давно ожидаемые новшества, такие как закруглённые углы, тени, градиенты, переходы или анимация, а также новые макеты, такие как multi-columns, «резиновый» дизайн или сеточный макет. Экспериментальный функционал помечен специальными префиксами разработчика, и он не должен использоваться в производственной среде, либо должен использоваться с особой осторожностью, так как его синтаксис и поведение может быть изменено в будущем. [20]

4 Проектирование задачи

4.1 Требования к приложению

Данное приложение требует авторизации пользователя.

Разрабатываемое приложение должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено. Для того чтобы интерфейс был понятен пользователю, необходимо оформить диалог приложения с пользователем, обладающий кнопочным меню, позволяющим перемещаться по UI элементам управления. Программное приложение должно обеспечивать оперативный поиск необходимой информации по предоставленным критериям, для этого будут разработаны процедуры отбора информации.

Все входные данные должны проверяться на ошибки. При совершении пользователем неверных действий или введении неверных данных будут выводиться сообщения с пояснением ошибок.

Функциональные требования представлены на диаграмме SADT, которая отображает вначале всю систему в виде простейшей компоненты – одного блока и дуг, изображающих интерфейсы с функциями вне системы. Единственный блок представляет всю систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных дуг – они также представляют полный набор внешних интерфейсов системы в целом. Диаграмма нулевого уровня представлена на рисунке 4.1.

Пользователь

Запросы пользователя

Заметки

Реализовать органайзер

Информация о пользователе

Рисунок 4.1 – Функциональная SADT диаграмма нулевого уровня

Далее блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых представлена как блок, границы которого определены интерфейсными дугами. Каждая из этих подфункций может быть декомпозирована подобным образом для более детального представления. Диаграмма первого уровня детализации представлена на рисунке 4.2.

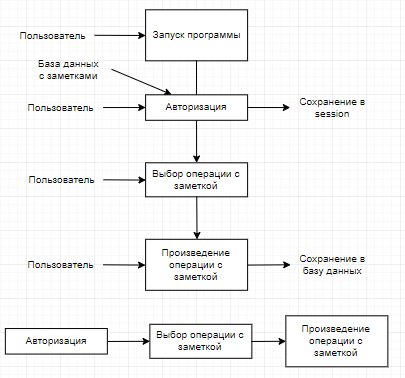


Рисунок 4.2 – Функциональная SADT диаграмма первого уровня детализации

Кроме этого, при разработке UI необходимо соблюдать определённые требования: интерфейс в приложении должен быть выполнены в едином стиле, сдержанной цветовой гамме, иметь стандартные элементы управления данными на всех страницах, не должны быть перегружены информацией, и в свое время понятны простому пользователю.

4.2 Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – систему меню и страниц.

Все действия можно будет выполнить с помощью главной страницы, которая будет появляться после авторизации. Главная страница будет иметь кнопочное меню программы позволяющее вызывать все функции программы, а также осуществлять различные манипуляции с данными. Используя меню, пользователь должен иметь возможность завершить работу всего приложения с сохранением сделанных изменений.

Главное меню программы будет представлено страницей на которой находятся основные элементы управления программой. С помощью кнопок на главной странице можно получить доступ к операциям с заметкой. Структура меню представлена в   
таблице 4.1.

Таблица 4.1 – Структура меню

|  |  |  |
| --- | --- | --- |
| Операции с поиском |  | |
| 1.1 | | Выбрать заметку |
| 1.2 | | Удалить заметку |
| 1.3 | | Сохранить изменения |
| 1.4 | | Добавить заметку |
| 1.5 | | Очистить поля |

4.3 Организация данных

SQL Server Management Studio (SSMS) — утилита из Microsoft SQL Server 2005 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Утилита включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера.

Главным инструментом SQL Server Management Studio является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью ими управлять.

Таблица «Пользователи» хранит информацию о пользователях приложения, структура приведена в таблице 4.2

Таблица 4.2 — Структура таблицы «Пользователи»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер,  байт | Описание |
| ID | Счетчик | 4 | Идентификационный номер пользователя |
| Главный Email | Текстовый | 50 | Email для связи с пользователем |
| Email для подтверждения пароля | Текстовый | 50 | Email для подтверждения пароля |
| Пароль | Текстовый | 50 | Пароль |
| Секретный ключ | Текстовый | 50 | Секретный ключ для подтверждения регистрации и изменения пароля |

Таблица «Заметки» хранит информацию о заметках пользователя, структура приведена в таблице 4.3

Таблица 4.3 — Структура таблицы «Заметки»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер,  байт | Описание |
| ID | Счетчик | 4 | Идентификационный номер заметки |
| Заголовок | Текстовый | 50 | Заголовок заметки |
| Дата | Дата | 3 | Дата используемая в целях пользователя |
| Описание | Текстовый | 100 | Описание заметки (html код) |
| ID пользователя | Вторичный ключ | 4 | Идентификационный номер пользователя |

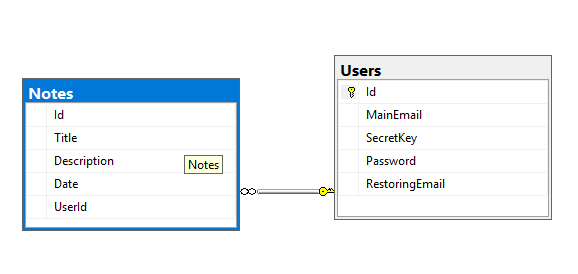


Рисунок 4.3 - Схема данных

4.4 Функции и элементы управления

На основании диаграммы вариантов использования в программном приложении «Программная реализация органайзера» были реализованы следующие функции:

* создание заметки;
* редактирование заметки;
* удаление заметки;
* страница для администратора.

Рассмотрим основные функции программы.

Функция создание заметки. Программный код реализации данной функции представлен ниже.

[HttpPost]

public JsonResult Add()

{

string Title = Request.Form["Title"];

string Description = Request.Unvalidated().Form["Description"];

string Date = Request.Form["Date"];

DateTime date = Convert.ToDateTime(Date);

string MainEmail = Session["MainEmail"].ToString();

int UserId = DB.db.Users.Where(a => a.MainEmail == MainEmail).FirstOrDefault().Id;

Note NewNote = new Note

{

Title = Title,

Description = Description,

Date = date,

UserId = UserId

};

DB.db.Notes.Add(NewNote);

DB.db.SaveChanges();

return Json(NewNote.Id);

}

//добавление заметки

$(document).on("click", "#AddNote", function () {

var Title = $('#Note-Title').val();

CKEDITOR.instances['Note-Discription'].updateElement();//обновляем данные об элементе

var Description = CKEDITOR.instances['Note-Discription'].getData();

var Date = $('#Note-Date').val().replace('T',' ');

//alert(typeof (Title)); alert(typeof (Description)); alert(typeof(Date));

$.ajax({

type: "POST", url: "http://localhost/Notes/Note/Add",

data: { Title: Title, Description: Description, Date: Date },

success: function (Id)

{

var NewNote = "<a class=\"list-group-item list-group-item-action\" note-id=\"" + Id + "\">" +

"<div class=\"w-100\">" +

"<div class=\"float-right\"><img src = \"../Content/Images/delete-ico.png\" alt=\"Удалить\" title=\"Удалить\" " +

"class=\"Notes-List-Item-Controll-Delete\"/></div>" +

"<div class=\"Note-List-Item-Date\">" + Date + "</div>" +

"</div>" +

"<div class=\"Notes-List-Item-Title\">" + Title + "</div>" +

"</a>";

$(NewNote).prependTo('#list-note-container');

},

crossDomain: true,

dataType: 'json'

});

});

Функция редактирования заметки. Программный код реализации данной функции представлен ниже.

//редактирование заметки

$(document).on("click", "#SaveNote", function () {

var Id = $('.list-group-item.active').attr('note-id');

if (Id == undefined) {

$('#messageBoxText').html('<div class="text-center mb-3">Выберите заметку!</div>');

$('#signupFormOk').modal('show');

return;

}

var Title = $('#Note-Title').val();

CKEDITOR.instances['Note-Discription'].updateElement();//обновляем данные об элементе

var Description = CKEDITOR.instances['Note-Discription'].getData();

var MyDate = $('#Note-Date').val().replace('T', ' ');

$.ajax({

type: "POST", url: "http://localhost/Notes/Note/Edit",

data: { Id: Id, Title: Title, Description: Description, Date: MyDate },

success: function () {

var MyFormatDate = GetSimpleDate(new Date(MyDate.replace(' ', 'T')));

$('.list-group-item.active .Note-List-Item-Date').text(MyFormatDate);

$('.list-group-item.active .Notes-List-Item-Title').text(Title);

},

crossDomain: true

});

});

[HttpPost]

public void Edit()

{

int Id = int.Parse(Request.Form["Id"]);

string Title = Request.Form["Title"];

string Description = Request.Unvalidated().Form["Description"];

string Date = Request.Form["Date"];

var note = DB.db.Notes.Find(Id);

note.Title = Title;

note.Description = Description;

note.Date = Convert.ToDateTime(Date);

DB.db.SaveChanges();

}

Функция удаления заметки. Программный код реализации данной функции представлен ниже.

[HttpPost]

public void Delete(int Id)

{

Note note = DB.db.Notes.Find(Id);

DB.db.Notes.Remove(note);

DB.db.SaveChanges();

}

//удаление заметки

$(document).on("click", ".Notes-List-Item-Controll-Delete", function () {

var Note = $(this).closest(".list-group-item");

var note\_id = Note.attr("note-id");

$.ajax({

type: "POST", url: "http://localhost/Notes/Note/Delete?Id=" + note\_id,

crossDomain: true,

dataType: 'json'

});

Note.remove();

});

Весь программный код представлен в приложении А.

5 Описание программного средства

5.1 Общие сведения

Программа «Программная реализация органайзера» создана для обеспечения удобной системы записи информации о планах и\или задачах пользователя. Данная программа была разработана на персональном компьютере со следующей конфигурацией:

* процессор Intel Core i5 6300 3,2 Ггц;
* оперативная память DDR4 8 Гб;
* жёсткий диск HDD Samsung 500 Гб;
* видеокарта NVIDIA GeForce GTX 950M 512 Мб.

Программа создана в средстве разработки Microsoft Visual Studio 2017 на языках программирования C# и JavaScript в операционной системе Windows 10. Она может работать в любых операционных системах, которые поддерживают браузер.

Программа не требовательна к системным ресурсам. Инсталляция не требуется.

5.2 Функциональное назначение

Программа «Программная реализация органайзера» создана для обеспечения удобной системы записи информации о планах и\или задачах пользователя.

В данном приложении будет присутствовать несколько классов необходимых для работы программы, а именно:

* class Authentication осуществляет авторизацию;
* class NoteController осуществляет crud операции с заметками;
* class AdminController осуществляет crud операции со всеми таблицами базы данных.

5.3 Входные и выходные данные

Входной информацией являются данные о пользователе.

Выходная информация будет представлена в виде информации на экране, а также в файле базы данных. В файле будет храниться вся информация о заметках.

6 Методика испытаний

6.1 Технические требования

Минимальные системные требования к приложению:

* 100 Мб свободного места на винчестере;
* 30 Мб свободной оперативной памяти;
* процессор Intel Pentium 3 Ггц или совместимый аналог;
* видеокарта с объемом памяти не менее 16 Мб.
* операционная система семейства Microsoft Windows NT;
* клавиатура, мышь.

6.2 Функциональное тестирование

В процессе написания программного продукта необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование программы будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

Форма регистрации представлена на рисунке 6.1.

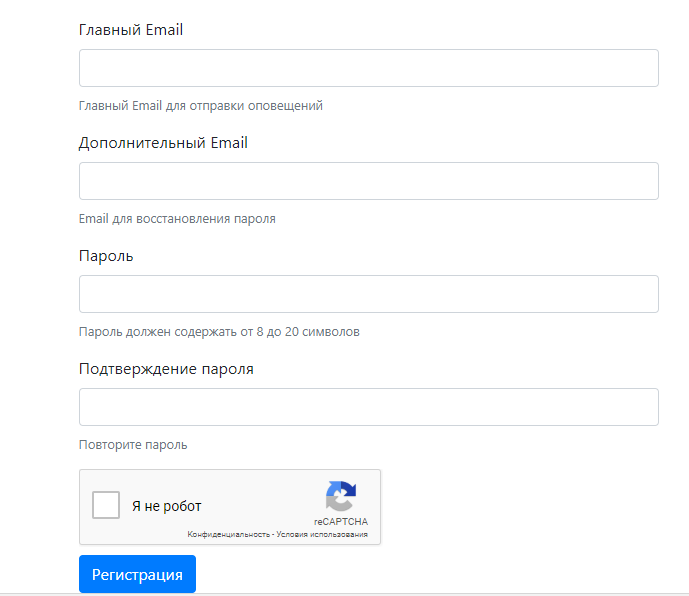


Рисунок 6.1 - Форма регистрации

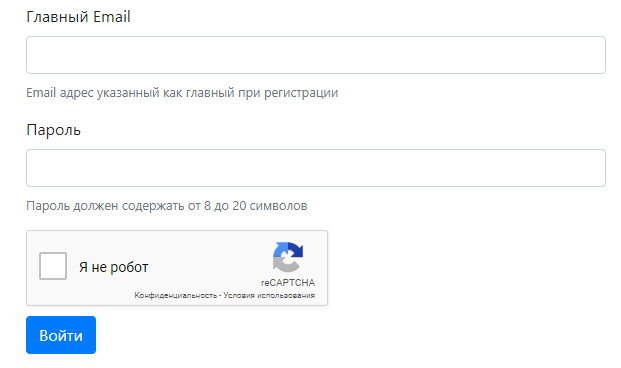


Рисунок 6.2 – Форма входа

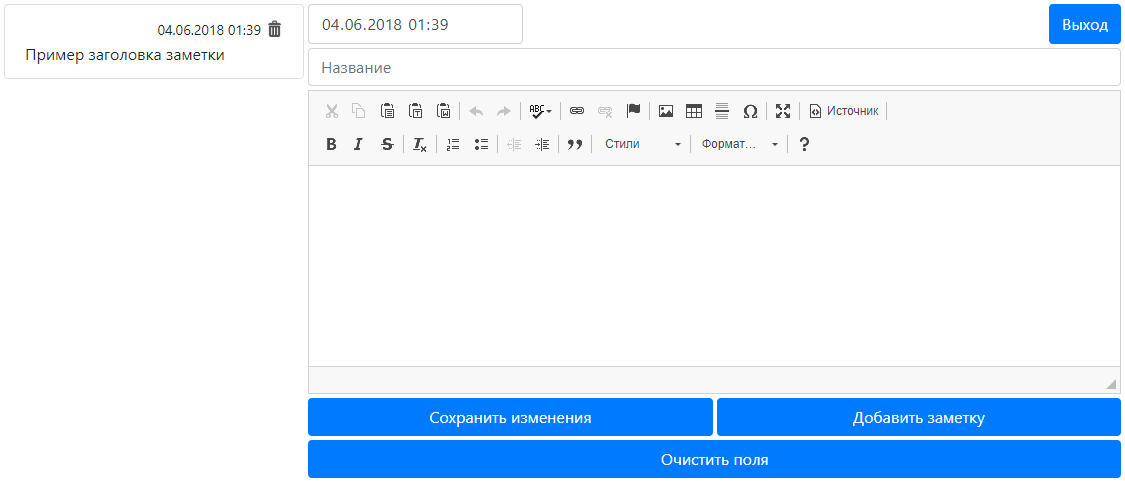


Рисунок 6.3 – Главная страница

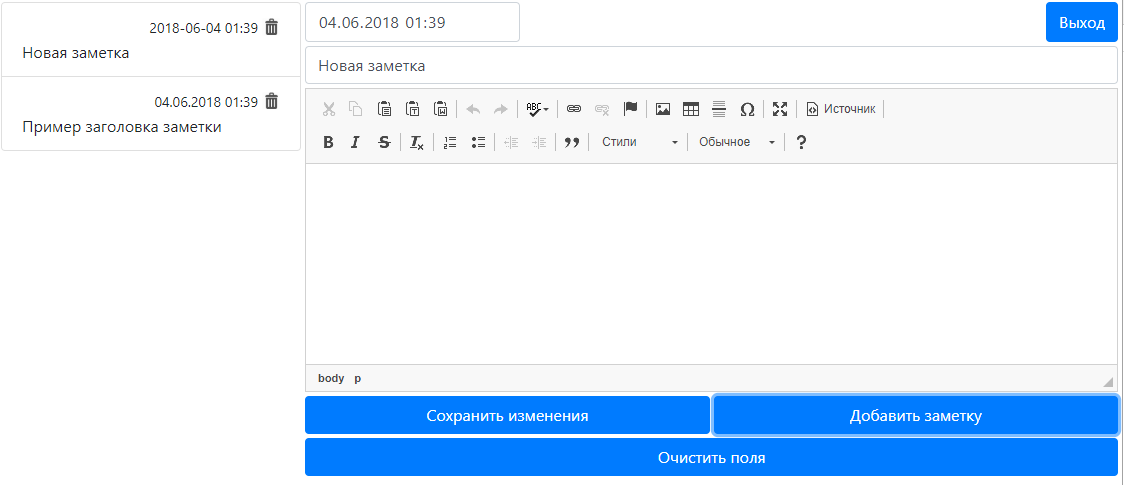


Рисунок 6.4 – Добавление заметки

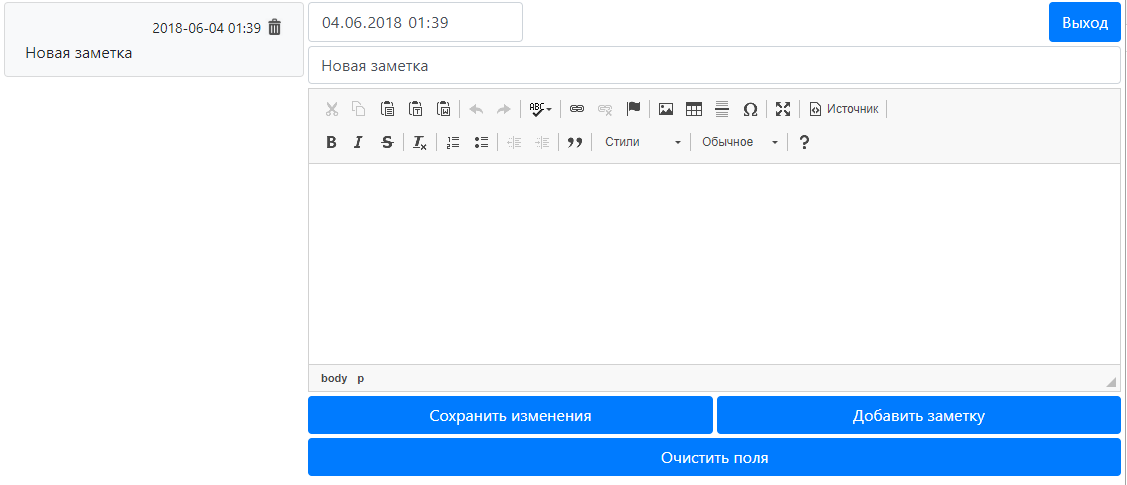


Рисунок 6.5 – Удаление заметки

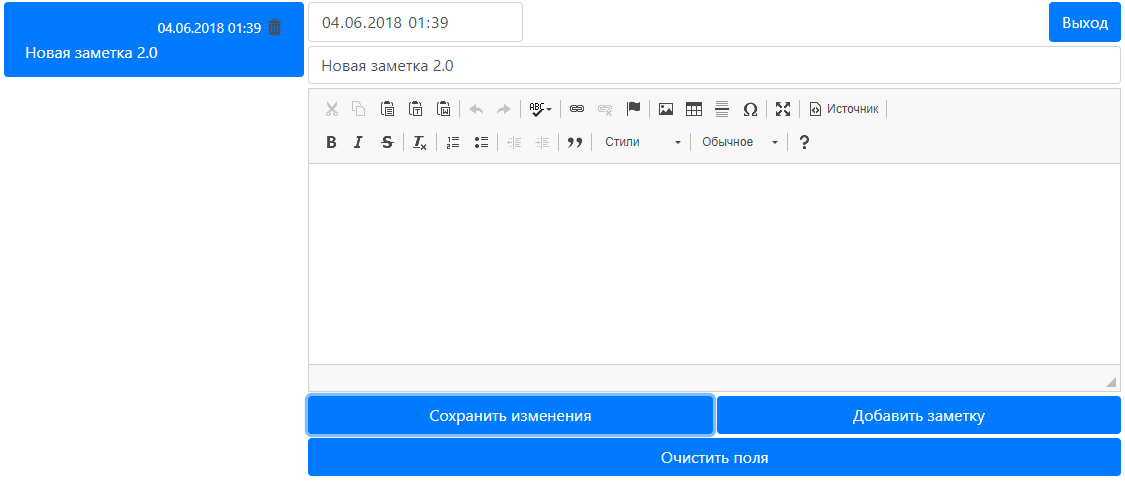


Рисунок 6.5 – Редактирование заметки

7 Применение

7.1 Назначение программы

Программа «Программная реализация органайзера» создана для обеспечения удобной системы записи информации о планах и\или задачах пользователя.

7.2 Условия применения

Для применения данного программного средства необходимы следующие технически требования:

* процессор Intel Core 2 Duo или выше;
* минимальный объем оперативной памяти — 512 Мбайт;
* операционная система Windows 7 и выше;
* Framework v4.0
* рекомендуется монитор типа VGA или с лучшей разрешающей способностью;
* устройство для чтения дисков;
* клавиатура;
* мышь.

Список используемых источников

1 Багласова Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г.Багласова. – Минск: КБП, 2017

2 Багласова Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г.Багласова, К.О.Якимович. – Минск: КБП, 2013

3 Вендров, А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. / А.М. Вендров. – 2-е изд., перераб. и доп.– М. : Финансы и статистика, 2006. — 544 с.

4 Мацяшек. Анализ и проектирование информационных систем с помощью UML 2.0 / Мацяшек, А. М. Лешек: – М. – ООО “И.Д. Вильямс”, 2008. — 816 с.

5 Орлов С. А. Технологии разработки программного обеспечения: Учебник для вузов. 4-е изд. / С. А. Орлов, Б. Я. Цилькер – СПб. : Питер, 2012. — 608 с.

6 Рудаков А. В., Технология разработки программных продуктов. Практикум: учеб. пособие для студ. учреждений сред. проф. образования / А.В. Рудаков, Г.Н. Федорова М. : Издательский центр «Академия»; 2014. — 192 с.

7 ГОСТ 19701-90 Схемы алгоритмов, программ, данных. – М.: Из-во стандартов, 1990

8 ГОСТ 19301-2000 Программа и методика испытаний. – М.: Из-во стандартов, 2000

9 ГОСТ 19401-2000 Текст программы. – М.: Из-во стандартов, 2000

10 ГОСТ 19402-2000 Описание программы. – М.: Из-во стандартов, 2000

11 ГОСТ 2.105-95 Общие требования к тестовым документам. – М.: Из-во   
стандартов, 1995

12 Справочные материалы по информационным технологиям [Электронный ресурс] / Режим доступа : http://itteach.ru/bpwin/

13 C# [Электронный ресурс] / Режим доступа : msdn.microsoft.com/ru-ru

14 Dr.Explain [Электронный ресурс] / Режим доступа : https://www.drexplain.ru

15 Microsoft Visual Studio 2017 [Электронный ресурс] / Режим доступа : https://www.microsoft.com/ru-ru/SoftMicrosoft/vs2017

16 MySQL [Электронный ресурс] / Режим доступа : https://www.mysql.com

17 UMLet [Электронный ресурс] / Режим доступа : http://www.umlet.com

18 Windows 10 [Электронный ресурс] / Режим доступа : https://www.microsoft.com/ru-ru/windows

Приложение А

(Обязательное)

Текст программы

AdminController

using Notes.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Notes.Controllers

{

public class AdminController: Controller

{

static string PrKey = "im\_admin";

public ActionResult Index(string Key)

{

if (Session["AdminKey"] != null) return View("Adminka");

if (PrKey == Key)

{

Session["AdminKey"] = Key;

return View("Adminka");

}

return View();

}

//users начало

public ActionResult Users()

{

return View();

}

public ActionResult EditUser(int id)

{

var User = DB.db.Users.Find(id);

ViewBag.User = User;

return View();

}

[HttpPost]

public ActionResult EditUser(User user)

{

var userNew = DB.db.Users.Find(user.Id);

userNew.MainEmail = user.MainEmail;

userNew.Password = user.Password;

userNew.SecretKey = user.SecretKey;

userNew.RestoringEmail = user.RestoringEmail;

DB.db.SaveChanges();

return RedirectPermanent("~/Admin/Users");

}

public ActionResult DeleteUser(int id)

{

DB.db.Users.Remove(DB.db.Users.Find(id));

DB.db.SaveChanges();

return RedirectPermanent("~/Admin/Users");

}

[HttpPost]

public ActionResult AddUser(User user)

{

Aes aes = new Aes(Aes.SAMPLE\_KEY, Aes.SAMPLE\_IV);

user.Password = aes.EncryptToBase64String(user.Password);

DB.db.Users.Add(user);

DB.db.SaveChanges();

return RedirectPermanent("~/Admin/Users");

}

public ActionResult AddUser()

{

return View();

}

//users конец

//notes начало

public ActionResult Notes()

{

return View();

}

public ActionResult EditNote(int id)

{

var note = DB.db.Notes.Find(id);

ViewBag.Note = note;

return View();

}

[HttpPost]

[ValidateInput(false)]

public ActionResult EditNote(Note note)

{

var noteNew = DB.db.Notes.Find(note.Id);

noteNew.Title = note.Title;

noteNew.Description = note.Description;

noteNew.Date = note.Date;

noteNew.UserId = note.UserId;

DB.db.SaveChanges();

return RedirectPermanent("~/Admin/Notes");

}

public ActionResult DeleteNote(int id)

{

DB.db.Notes.Remove(DB.db.Notes.Find(id));

DB.db.SaveChanges();

return RedirectPermanent("~/Admin/Notes");

}

[HttpPost]

public ActionResult AddNote(Note note)

{

note.User = DB.db.Users.Find(note.UserId);

DB.db.Notes.Add(note);

DB.db.SaveChanges();

return RedirectPermanent("~/Admin/Notes");

}

public ActionResult AddNote()

{

return View();

}

//notes конец

}

}

NoteController

using Notes.Models;

using System.Web.Mvc;

using System.Linq;

using System.Net;

using Newtonsoft.Json.Linq;

using System.IO;

using System.Configuration;

using System.Threading.Tasks;

using System;

using Notes.Controllers;

using System.Web.Helpers;

namespace Заметки.Controllers

{

public class NoteController : Controller

{

public ActionResult Index()

{

return View();

}

[HttpPost]

public JsonResult Add()

{

string Title = Request.Form["Title"];

string Description = Request.Unvalidated().Form["Description"];

string Date = Request.Form["Date"];

DateTime date = Convert.ToDateTime(Date);

string MainEmail = Session["MainEmail"].ToString();

int UserId = DB.db.Users.Where(a => a.MainEmail == MainEmail).FirstOrDefault().Id;

Note NewNote = new Note

{

Title = Title,

Description = Description,

Date = date,

UserId = UserId

};

DB.db.Notes.Add(NewNote);

DB.db.SaveChanges();

return Json(NewNote.Id);

}

[HttpPost]

public JsonResult GetNote()

{

int Id = int.Parse(Request.Form["Id"].ToString());

return Json(DB.db.Notes.Find(Id).Clone());

}

[HttpPost]

public void Edit()

{

int Id = int.Parse(Request.Form["Id"]);

string Title = Request.Form["Title"];

string Description = Request.Unvalidated().Form["Description"];

string Date = Request.Form["Date"];

var note = DB.db.Notes.Find(Id);

note.Title = Title;

note.Description = Description;

note.Date = Convert.ToDateTime(Date);

DB.db.SaveChanges();

}

[HttpPost]

public void Delete(int Id)

{

Note note = DB.db.Notes.Find(Id);

DB.db.Notes.Remove(note);

DB.db.SaveChanges();

}

}

}

UserController

using Notes.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Notes.Controllers

{

public class UserController : Controller

{

DBContext db = new DBContext();

public ActionResult Exit()

{

Session["MainEmail"] = null;

return RedirectPermanent("~/User/Home");

}

public ActionResult Home()

{

//Session["MainEmail"] = "egordubrovskij76@gmail.com";

if (Session["MainEmail"] == null) return View("~/Views/Home/Index.cshtml");

string MainEmail = Session["MainEmail"].ToString();

int UserId = DB.db.Users.Where(a => a.MainEmail == MainEmail).FirstOrDefault().Id;

List<Note> notes = db.Notes.Where(a=>a.UserId==UserId).OrderByDescending(a=>a.Id).ToList();

ViewBag.Notes = notes;

return View("~/Views/User/Home.cshtml");

}

[HttpPost]

public JsonResult IsRestoringEmail(string Email) => Json(db.Users.Any(a => a.RestoringEmail == Email));

[HttpPost]

public JsonResult IsMainEmail(string Email) => Json(db.Users.Any(a => a.MainEmail == Email));

[HttpPost]

public JsonResult IsSecretKeyByMainEmail(string Email) => Json(db.Users.Any(a => a.MainEmail == Email && a.SecretKey != ""));

[HttpPost]

public JsonResult IsPassByMainEmail(string Pass, string Email)

{

Aes aes = new Aes(Aes.SAMPLE\_KEY, Aes.SAMPLE\_IV);

Pass = aes.EncryptToBase64String(Pass);

return Json(db.Users.Any(a => a.MainEmail == Email && a.Password == Pass));

}

[HttpPost]

public JsonResult IsValidRecapcha(string response) => Json(ReCaptchaClass.Validate(response) == "True" ? true : false);

[HttpPost]

public void Registration(string MainEmail, string RestoringEmail, string Password)

{

Aes aes = new Aes(Aes.SAMPLE\_KEY, Aes.SAMPLE\_IV);

string EncryptToBase64Password = aes.EncryptToBase64String(Password);

Random random = new Random();

User user = new User

{

MainEmail = MainEmail,

RestoringEmail = RestoringEmail,

Password = EncryptToBase64Password,

SecretKey = aes.EncryptToBase64String(MyRandom.RandomString(random.Next(5, 10)))

};

db.Users.Add(user);

db.SaveChanges();

//отправка письма для подтверждения на почту

string Link = "<a href=\"http://localhost/Notes/User/ConfReg?MainEmail=" + MainEmail +

"&SecretKey=" + user.SecretKey+"\"><b>ссылка</b></a>";

string Message = "<h3>Для подтверждения регистрации на сайте Notes перейдите по ссылке: " +

Link+"</h3>";

Email.SendMessage("egordubrovskij76@gmail.com", "Админ сайта Notes ","123456789egor",

MainEmail, "Уважаемый пользователь", "Уважаемый пользователь", Message);

}

private int GetIdByMainEmail(string MainEmail)

{

List<User> user = db.Users.Where(a=>a.MainEmail == MainEmail).ToList();

if (user.Count == 0) return -1;

else return user[0].Id;

}

//Подтверждение регистрации при переходе по ссылки с почты

public ActionResult ConfReg(string MainEmail)

{

//обычным методом не достать переменную из url тк там есть символ - "+" который пропускаеться

string SecretKey = Request.RawUrl.Substring(Request.RawUrl.IndexOf("SecretKey") + 10);

int UserId = GetIdByMainEmail(MainEmail);

if (UserId == -1) return View("~/Views/Shared/Error.cshtml");

if (SecretKey != db.Users.Find(UserId).SecretKey) return View("~/Views/Shared/Error.cshtml");

//удаляем секретный ключ из бд

db.Database.ExecuteSqlCommand("UPDATE Users SET SecretKey='' WHERE Id=" + UserId);

//сохраняем пользователя в сессию

Session["MainEmail"] = MainEmail;

return RedirectPermanent("~/User/Home");

}

public void Login(string MainEmail)

{

Session["MainEmail"] = MainEmail;

}

//отрпавка письма для восстановления пароля

public void SendMesForPassRec(string RestoringEmail)

{

Aes aes = new Aes(Aes.SAMPLE\_KEY, Aes.SAMPLE\_IV);

string SecretKey = aes.EncryptToBase64String(MyRandom.RandomString(new Random().Next(5, 10)));

//отправка письма для подтверждения на почту

string Link = "<a href=\"http://localhost/Notes/User/PassRec?RestoringEmail=" + RestoringEmail +

"&SecretKey=" + SecretKey + "\"><b>ссылка</b></a>";

string Message = "<h3>Для восстановления пароля на сайте Notes перейдите по ссылке: " +

Link + "</h3>";

Email.SendMessage("egordubrovskij76@gmail.com", "Админ сайта Notes ", "123456789egor",

RestoringEmail, "Уважаемый пользователь", "Уважаемый пользователь", Message);

db.Database.ExecuteSqlCommand("UPDATE Users SET SecretKey='"+ SecretKey + "' WHERE RestoringEmail='" + RestoringEmail+"'");

}

public ActionResult PassRec(string RestoringEmail)

{

//обычным методом не достать переменную из url тк там есть символ - "+" который пропускаеться

string SecretKey = Request.RawUrl.Substring(Request.RawUrl.IndexOf("SecretKey") + 10);

if (db.Users.Any(a => a.RestoringEmail == RestoringEmail && a.SecretKey == SecretKey))

return View("~/Views/User/PassChange.cshtml");

else return View("~/Views/Shared/Error.cshtml");

}

public void SaveNewPass(string NewPassword, string RestoringEmail)

{

Aes aes = new Aes(Aes.SAMPLE\_KEY, Aes.SAMPLE\_IV);

NewPassword = aes.EncryptToBase64String(NewPassword);

db.Database.ExecuteSqlCommand("UPDATE Users SET Password='"+ NewPassword +

"' WHERE RestoringEmail='" + RestoringEmail + "'");

db.Database.ExecuteSqlCommand("UPDATE Users SET SecretKey='' WHERE RestoringEmail='" + RestoringEmail + "'");

string Message = "<h3>Пароль успешно изменён!</h3>";

Email.SendMessage("egordubrovskij76@gmail.com", "Админ сайта Notes ", "123456789egor",

RestoringEmail, "Уважаемый пользователь", "Уважаемый пользователь", Message);

}

}

}

ListNotesItemHelper

using Notes.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Notes.Helpers

{

public static class ListNotesItemHelper

{

public static MvcHtmlString CreateList(this HtmlHelper html, List<Note> notes)

{

string a = "";

if(notes.Count==0)

{

a += "<a class=\"list-group-item list-group-item-action Example\" \">" +

"<div class=\"w-100\">" +

"<div class=\"float-right\"><img src = \"../Content/Images/delete-ico.png\" alt=\"Удалить\" title=\"Удалить\" " +

"class=\"Notes-List-Item-Controll-Delete-Example\"/></div>" +

"<div class=\"Note-List-Item-Date\">" + MyLibrary.GetSimpleDateFormat2(DateTime.Now) + "</div>" +

"</div>" +

"<div class=\"Notes-List-Item-Title\">Пример заголовка заметки</div>" +

"</a>";

}

else

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

{

a += "<a class=\"list-group-item list-group-item-action\" note-id=\""+notes[i].Id+"\">"+

"<div class=\"w-100\">" +

"<div class=\"float-right\"><img src = \"../Content/Images/delete-ico.png\" alt=\"Удалить\" title=\"Удалить\" " +

"class=\"Notes-List-Item-Controll-Delete\"/></div>" +

"<div class=\"Note-List-Item-Date\">"+ MyLibrary.GetSimpleDateFormat2(notes[i].Date)+"</div>" +

"</div>" +

"<div class=\"Notes-List-Item-Title\">"+ notes[i].Title+"</div>" +

"</a>";

}

string res =

"<div class=\"row m-0\">" +

"<div class=\"col-12 p-0\">" +

"<div class=\"list-group\" id=\"list-note-container\">"+

a+

"</div>" +

"</div>" +

"</div>";

return new MvcHtmlString(res);

}

}

}

Authentication

let RegValPass = new ValidationPassword();

let RegValPassConf = new ValidationPasswordConf();

let RegValMainEmail = new ValidationForDoubleEmail();

let RegValRestEmail = new ValidationForDoubleEmail();

let AutValMainEmail = new ValidationExistMainEmail();

let AutValPass = new ValidationPassForEmail();

let PassRecValRecEmail = new SimpleValidationRecEmail();

var RecapchaReg; var RecapchaAut; var RecapchaPassRec;

var onloadCallback = function(){

//recapcha begin

// Renders the HTML elements reCAPTCHA widgets.

// The id of the reCAPTCHA widget is assigned to 'widgetId1'.

RecapchaReg = grecaptcha.render('g-recaptcha-response-reg', {

'sitekey': '6LeaWFUUAAAAAAa32wacPyJwDIXPpHCp05iJbpJG'

});

RecapchaAut = grecaptcha.render(document.getElementById('g-recaptcha-response-aut'), {

'sitekey': '6LeaWFUUAAAAAAa32wacPyJwDIXPpHCp05iJbpJG'

});

RecapchaPassRec = grecaptcha.render('g-recaptcha-response-passRecovery', {

'sitekey': '6LeaWFUUAAAAAAa32wacPyJwDIXPpHCp05iJbpJG'

});

//recapcha end

}

$(document).ready(function () {

$('.notSpace').keypress(function (key) {

if (key.charCode == 32) return false;

});

$('.notSpace').on('paste', function (e) { return false; });

});

function Registr() {

if (RegValMainEmail.IsVal && RegValRestEmail.IsVal && RegValPass.IsVal &&

RegValPassConf.IsVal && IsValidRecapcha(RecapchaReg))

{

var MainEmail = $('#form\_signup\_email').val();

var RestEmail = $('#form\_signup\_restoring\_email').val();

var Password = $('#form\_signup\_pass').val();

$.ajax({

type: "POST", url: "http://localhost/Notes/User/Registration?MainEmail=" + MainEmail + "&RestoringEmail=" +

RestEmail + "&Password=" + Password,

success: function (data)

{

$('#messageBoxText').html('<div class="text-center">Поздравляем, вы успешно зарегистрираны!</div><p>В течении нескольких минут на указанную вами почту придёт письмо! Для завершения регистрациии перейдите по ссылке в письме!</p>');

$('#signupFormOk').modal('show');

}

});

}

else $('#errorAreaForReg').removeClass("d-none");

}

function UserLogin() {

if (AutValMainEmail.IsVal && AutValPass.IsVal && IsValidRecapcha(RecapchaAut)) {

var MainEmail = $('#form\_authorization\_email').val();

$.ajax({ type: "POST", url: "http://localhost/Notes/User/Login?MainEmail=" + MainEmail });

window.location.replace("http://localhost/Notes/User/Home");

}

else $('#errorAreaLogin').removeClass("d-none");

}

function PassRecSendToEmail() {

if (PassRecValRecEmail.IsVal && IsValidRecapcha(RecapchaPassRec)) {

var Email = $('#passRecoveryEmail').val();

$.ajax({

type: "POST", url: "http://localhost/Notes/User/SendMesForPassRec?RestoringEmail=" + Email,

success: function ()

{

$('#passRecoveryModal').modal('hide');

$('#messageBoxText').html('<p>В течении нескольких минут на указанную вами почту придёт письмо! Для восстановления пароля перейдите по ссылке в письме</p>');

$('#signupFormOk').modal('show');

}

});

}

else $('#errorAreaPassRecovery').removeClass("d-none");

}

function IsValidRecapcha(RecapWidget) {

var IsVal;

var response = grecaptcha.getResponse(RecapWidget);

$.ajax({

type: "POST", url: "http://localhost/Notes/User/IsValidRecapcha?response=" + response, async: false,

success: function (data) { IsVal = Boolean(data); }

});

grecaptcha.reset(RecapWidget);

return IsVal;

}

//модальное окно для отправки письма на вост. пароля

$('#passRecoveryModal').modal({

backdrop: "static",//запрежаем закрытие при клике вне формы мышью

show: false

});

//модальное окно для уведомления пользователя о успешной регистрации

$('#signupFormOk').modal({

backdrop: "static",

show: false

});

function PassRecoveryClick() { $('#passRecoveryModal').modal('show'); }

UserHome

var IsShowListNones = Boolean(false);

//показать список заметок

$(document).on("click", "#Notes-List-Show", function () {

if (IsShowListNones) {

IsShowListNones = Boolean(false);

$('.Notes-List-Container').removeClass("active");

} else {

IsShowListNones = Boolean(true);

$('.Notes-List-Container').addClass("active");

}

});

//выбор заметки

$(document).on("click", ".list-group-item:not(.Example)", function () {

$('.list-group-item.active').removeClass('active');

$(this).addClass('active');

var NoteId = $(this).attr("note-id");

$.ajax({

type: "POST", url: "http://localhost/Notes/Note/GetNote",

data: { Id: NoteId},

success: function (data) {

$('#Note-Date').val(data.Date);

$('#Note-Title').val(data.Title);

CKEDITOR.instances['Note-Discription'].setData(data.Description);

},

crossDomain: true,

dataType: 'json'

});

});

//выход

$(document).on("click", "#user-exit", function () { window.location.replace("http://localhost/Notes/User/Exit");});

//удаление заметки

$(document).on("click", ".Notes-List-Item-Controll-Delete", function () {

var Note = $(this).closest(".list-group-item");

var note\_id = Note.attr("note-id");

$.ajax({

type: "POST", url: "http://localhost/Notes/Note/Delete?Id=" + note\_id,

crossDomain: true,

dataType: 'json'

});

Note.remove();

});

//удаление тестовой заметки

$(document).on("click", ".Notes-List-Item-Controll-Delete-Example", function () {

if ($(".list-group-item").length < 2) return;

var Note = $(this).closest(".list-group-item").remove();

});

//Очистка

$(document).on("click", "#ClearNote", function () { ClearNoteEditPanel();});

function ClearNoteEditPanel() {

$('#Note-Date').val(GetSurDate(new Date()));

$('#Note-Title').val('');

CKEDITOR.instances['Note-Discription'].setData('');

}

//добавление заметки

$(document).on("click", "#AddNote", function () {

var Title = $('#Note-Title').val();

CKEDITOR.instances['Note-Discription'].updateElement();//обновляем данные об элементе

var Description = CKEDITOR.instances['Note-Discription'].getData();

var Date = $('#Note-Date').val().replace('T',' ');

//alert(typeof (Title)); alert(typeof (Description)); alert(typeof(Date));

$.ajax({

type: "POST", url: "http://localhost/Notes/Note/Add",

data: { Title: Title, Description: Description, Date: Date },

success: function (Id)

{

var NewNote = "<a class=\"list-group-item list-group-item-action\" note-id=\"" + Id + "\">" +

"<div class=\"w-100\">" +

"<div class=\"float-right\"><img src = \"../Content/Images/delete-ico.png\" alt=\"Удалить\" title=\"Удалить\" " +

"class=\"Notes-List-Item-Controll-Delete\"/></div>" +

"<div class=\"Note-List-Item-Date\">" + Date + "</div>" +

"</div>" +

"<div class=\"Notes-List-Item-Title\">" + Title + "</div>" +

"</a>";

$(NewNote).prependTo('#list-note-container');

},

crossDomain: true,

dataType: 'json'

});

});

//редактирование заметки

$(document).on("click", "#SaveNote", function () {

var Id = $('.list-group-item.active').attr('note-id');

if (Id == undefined) {

$('#messageBoxText').html('<div class="text-center mb-3">Выберите заметку!</div>');

$('#signupFormOk').modal('show');

return;

}

var Title = $('#Note-Title').val();

CKEDITOR.instances['Note-Discription'].updateElement();//обновляем данные об элементе

var Description = CKEDITOR.instances['Note-Discription'].getData();

var MyDate = $('#Note-Date').val().replace('T', ' ');

$.ajax({

type: "POST", url: "http://localhost/Notes/Note/Edit",

data: { Id: Id, Title: Title, Description: Description, Date: MyDate },

success: function () {

var MyFormatDate = GetSimpleDate(new Date(MyDate.replace(' ', 'T')));

$('.list-group-item.active .Note-List-Item-Date').text(MyFormatDate);

$('.list-group-item.active .Notes-List-Item-Title').text(Title);

},

crossDomain: true

});

});

function GetSimpleDate(D)

{

var MyDate = "";

var Year = D.getFullYear();

var Month = D.getMonth() + 1;

var Day = D.getDate();

var Hours = D.getHours();

var Min = D.getMinutes();

if (Day < 10) MyDate += "0" + Day + ".";

else MyDate += Day+".";

if (Month < 10) MyDate += "0" + Month + ".";

else MyDate += Month + ".";

if (Year < 10) MyDate += "0" + Year;

else MyDate += Year;

MyDate += " ";

if (Hours < 10) MyDate += "0" + Hours + ":";

else MyDate += Hours + ":";

if (Min < 10) MyDate += "0" + Min;

else MyDate += Min;

return MyDate;

}

//для даты

function zeroPadded(val) {

if (val >= 10)

return val;

else

return '0' + val;

}

function GetSurDate(d) {

return (d.getFullYear() + "-" + zeroPadded(d.getMonth() + 1) + "-" + zeroPadded(d.getDate()) +

"T" + zeroPadded(d.getHours()) + ":" + zeroPadded(d.getMinutes()));

}

//настройки при старте страницы

jQuery(document).ready(function ($) {

$('#Note-Date').val(GetSurDate(new Date()));

$(".Notes-List-Item-Description").dotdotdot();

CKEDITOR.replace("Note-Discription");

});

//модальное окно для уведомления пользователя

$('#signupFormOk').modal({

backdrop: "static",

show: false

});

Home/Index.cshtml

<!-- recaptcha -->

<script src="https://www.google.com/recaptcha/api.js?onload=onloadCallback&render=explicit" async defer></script>

<ul class="nav nav-pills mb-3 mt-1 ml-1" id="pills-tab" role="tablist">

<li class="nav-item">

<a class="nav-link active" data-toggle="pill" href="#reg-container" role="tab" aria-controls="pills-reg" aria-selected="true">Регистрация</a>

</li>

<li class="nav-item">

<a class="nav-link" data-toggle="pill" href="#authorization-container" role="tab" aria-controls="pills-authorization" aria-selected="false">Вход</a>

</li>

<li class="nav-item nav-link" title="восстановление пароля">

<img src="~/Content/Images/ForGuest/passRecovery.png" onclick="PassRecoveryClick()" alt="восстановление пароля" id="PassRecovery">

</li>

</ul>

<!-- модальное окно для уведомления пользователя-->

<div class="modal fade" id="signupFormOk" tabindex="-1" role="dialog" aria-labelledby="signupFormOk" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-body">

<div class="row">

<div class="col-12">

<h5 class="modal-title" id="messageBoxText"></h5>

</div>

</div>

<div class="row">

<div class="col-12"><button type="button" class="btn btn-primary w-100" data-dismiss="modal" aria-label="Close">ок</button></div>

</div>

</div>

</div>

</div>

</div>

<!--Регистрация-->

<div class="tab-content d-flex justify-content-center" id="pills-tabContent">

<div class="tab-pane fade show active" id="reg-container" role="tabpanel" aria-labelledby="reg-container-tab">

<form id="form\_signup">

<div class="row">

<div class="col-12">

<div class="form-group" id="signup-form-group-email">

<label class="form-control-label">Главный Email</label>

<input type="email" class="form-control notSpace" id="form\_signup\_email" oninput="RegValMainEmail.EmailChange('form\_signup\_email', 'form\_signup\_restoring\_email', 'signup-form-email-feedback')" required>

<h6 id="signup-form-email-feedback"></h6>

<small class="form-text text-muted">Главный Email для отправки оповещений</small>

</div>

</div>

<div class="col-12">

<div class="form-group" id="signup-form-group-restoring\_email">

<label class="form-control-label">Дополнительный Email</label>

<input type="email" class="form-control notSpace" id="form\_signup\_restoring\_email" oninput="RegValRestEmail.EmailChange('form\_signup\_restoring\_email', 'form\_signup\_email', 'signup-form-restoring\_email-feedback')" required>

<h6 id="signup-form-restoring\_email-feedback"></h6>

<small class="form-text text-muted">Email для восстановления пароля</small>

</div>

</div>

</div>

<div class="row">

<div class="col-12">

<div class="form-group" id="signup-form-group-password">

<label class="form-control-label">Пароль</label>

<input type="password" class="form-control notSpace" id="form\_signup\_pass" oninput="RegValPass.PassChange('form\_signup\_pass', 'signup-form-pass-feedback')" required>

<h6 id="signup-form-pass-feedback"></h6>

<small class="form-text text-muted">Пароль должен содержать от 8 до 20 символов</small>

</div>

</div>

<div class="col-12">

<div class="form-group" id="signup-form-group-password2">

<label class="form-control-label">Подтверждение пароля</label>

<input type="password" class="form-control notSpace" id="form\_signup\_pass2" oninput="RegValPassConf.PassConfChange('form\_signup\_pass2', 'form\_signup\_pass', 'signup-form-pass2-feedback')" required>

<h6 id="signup-form-pass2-feedback"></h6>

<small class="form-text text-muted">Повторите пароль</small>

</div>

</div>

</div>

<div class="row">

<div class="col-12">

<div>

<div id="g-recaptcha-response-reg" class="g-recaptcha"></div>

<h6 id="passRecoveryRecaptchaError"></h6>

</div>

</div>

</div>

<button type="button" class="btn btn-primary mr-1" onclick="Registr()">Регистрация</button>

<span class="d-none text-danger" id="errorAreaForReg">Введите корректные данные!</span>

</form>

</div>

<!--Авторизация-->

<div class="tab-pane fade" id="authorization-container" role="tabpanel" aria-labelledby="authorization-container-tab">

<form id="form\_authorization">

<div class="form-group" id="authorization-form-group-email">

<label class="form-control-label">Главный Email</label>

<input type="text" class="form-control notSpace" id="form\_authorization\_email" oninput="AutValMainEmail.EmailChange('form\_authorization\_email', 'login-form-authorization-email-feedback')" required>

<h6 id="login-form-authorization-email-feedback"></h6>

<small class="form-text text-muted">Email адрес указанный как главный при регистрации</small>

</div>

<div class="form-group" id="authorization-form-group-password">

<label class="form-control-label">Пароль</label>

<input type="password" class="form-control notSpace" id="form\_authorization\_pass" name="password" oninput="AutValPass.PassChange('form\_authorization\_pass', 'form\_authorization\_email', 'form-authorization-pass-invalid')" required>

<h6 id="form-authorization-pass-invalid"></h6>

<small class="form-text text-muted">Пароль должен содержать от 8 до 20 символов</small>

</div>

<div class="row">

<div class="col-12">

<div>

<div id="g-recaptcha-response-aut" class="g-recaptcha"></div>

<h6 id="passRecoveryRecaptchaError"></h6>

</div>

</div>

</div>

<button type="button" class="btn btn-primary mr-1" onclick="UserLogin()">Войти</button>

<span class="d-none text-danger" id="errorAreaLogin">Введите корректные данные!</span>

</form>

</div>

</div>

<!-- модальное окно для отправки письма на восстановление пароля -->

<div class="modal fade" id="passRecoveryModal" tabindex="-1" role="dialog" aria-labelledby="passRecoveryModal" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title">Восстановление пароля</h5>

<button class="close" type="button" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span><!-- крестик -->

</button>

</div>

<div class="modal-body">

<form id="passRecoveryForm">

<div class="row">

<div class="col-12">

<div class="form-group">

<label class="form-control-label">Адрес почты для восстановления пароля</label>

<input type="email" id="passRecoveryEmail" class="form-control notSpace" oninput="PassRecValRecEmail.EmailChange('passRecoveryEmail','passRecoveryEmailError')" required>

<h6 id="passRecoveryEmailError"></h6>

</div>

</div>

</div>

<div class="row">

<div class="col-12">

<div>

<div id="g-recaptcha-response-passRecovery" class="g-recaptcha"></div>

<h6 id="passRecoveryRecaptchaError"></h6>

</div>

</div>

</div>

<button type="button" onclick="PassRecSendToEmail()" class="btn btn-primary mr-1">Отправить письмо</button>

<span class="d-none text-danger" id="errorAreaPassRecovery">Введите корректные данные!</span>

</form>

</div>

</div>

</div>

</div>

<script src="~/Scripts/Libraries/MyLibrary.js"></script>

<script src="~/Scripts/Authentication.js"></script>