МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности1–40 05 01-03 Информационные системы и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к дипломному проекту на тему:

«Веб-приложение для учета и анализа показателей работы персонала предприятия»

Дипломник Рауба Алексей Александрович

(Ф.И.О.)

Руководитель проекта к.т.н., доц., Буснюк Н.Н.

(уч. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц., Смелов В.В.

(уч. степень, звание, должность, подпись, Ф.И.О.)

Консультанты: к.т.н., доц., Буснюк Н.Н.

(уч. степень, звание, должность, подпись, Ф.И.О.)

к.э.н., доц., Россоха Е.В.

(уч. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер: асс. каф. Жигаровская С.А.

(уч. степень, звание, должность, подпись, Ф.И.О.)

Дипломный проект защищен с оценкой

Председатель ГЭК к.т.н., доц., Дюбков В.К.

(уч. степень, звание, подпись, Ф.И.О.)

Минск 2018

Реферат

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*ДП 00.00.ПЗ*

*Разраб.*

*Рауба А.А.*

*Провер.*

*.*

*Буснюк Н.Н.*

Консульт.

*Буснюк Н.Н.*

*Н. Контр.*

*Жигаровская С.А.*

*Утверд.*

*Смелов В.В.*

Реферат

*Лит.*

*Листов*

*1*

БГТУ 74419017, 2018

Пояснительная записка дипломного проекта содержит 73 страницы, 45 иллюстраций, 6 таблиц, 14 формул, 12 источников литературы, 10 приложений, 4 листинга.

WEB-ПРИЛОЕЖНИЕ, КОНТРОЛЬ ПЕРСОНАЛА, СИСТЕМА УЧЕТА РАБОЧЕГО ВРЕМЕНИ, SQL DATABASE, MVC 5, ASP.NET, С#, JAVASCRIPT, MORRIS.JS, СРЕДА РАЗРАБОТКИ VISUAL STUDIO

Основной целью дипломного проекта является разработка веб-приложения для учета и анализа показателей работы персонала предприятия.

Пояснительная записка состоит из введения, пяти разделов и заключения.

Во введение представлена информация о проблеме контроля персонала.

Первый раздел содержит результаты обзора теоретического материала, а также патентного поиска и аналогов.

Во втором разделе произведен выбор средств разработки программного средства.

В третьем разделе показаны процессы проектирования и разработки программного средства.

В четвертом разделе представлено руководство пользователя и проведено тестирование программного средства.

В пятом разделе представлены результаты себестоимости и отпускной цены разработанного программного средства.

В заключении представлены итоги дипломного проекта и задачи, которые были решены в ходе разработки программного средства.

У

**Содержание**

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*3*

*ДП 00.00.ПЗ*

*Разраб.*

*Рауба А.А.*

*Провер.*

*.*

*Буснюк Н.Н.*

Консульт.

*Буснюк Н.Н.*

*Н. Контр.*

*Жигаровская С.А.*

*Утверд.*

*Смелов В.В.*

Содержание

*Лит.*

*Листов*

*2*

БГТУ 74419017, 2018

У

[ВВЕДЕНИЕ 5](#_Toc515313068)

[1 Обзор теоретического материала и аналогов 6](#_Toc515313069)

[1.1 Контроль персонала 6](#_Toc515313070)

[1.2 Методы контроля персонала 6](#_Toc515313071)

[1.3 Обзор системы контроля StaffCounter 7](#_Toc515313072)

[1.4 Система учета рабочего времени Эпикур 8](#_Toc515313073)

[1.5 Система учета рабочего времени Kickidler 9](#_Toc515313074)

[1.6 Программа для учета рабочего времени Mipko PM 11](#_Toc515313075)

[1.7 Вывод по разделу 12](#_Toc515313076)

[2 Обоснование технических приемов программирования 13](#_Toc515313077)

[2.1 Платформа ASP.NET 13](#_Toc515313078)

[2.2 СУБД Microsoft SQL Server 14](#_Toc515313079)

[2.3 Язык программирования C# 14](#_Toc515313080)

[2.4 Язык SQL 16](#_Toc515313081)

[2.5 Язык разметки HTML5 16](#_Toc515313082)

[2.6 Библиотека jQuery 17](#_Toc515313083)

[2.7 Язык CSS 18](#_Toc515313084)

[2.8 Библиотека Morris 18](#_Toc515313085)

[2.9 Паттерн проектирования MVC 19](#_Toc515313086)

[2.10 Вывод по разделу 20](#_Toc515313087)

[3 Проектирование и разработка программного средства 21](#_Toc515313088)

[3.1 Проектирование программного средства 21](#_Toc515313089)

[3.2 Структура приложения 21](#_Toc515313090)

[3.3 Разработка диаграммы вариантов использования 22](#_Toc515313091)

[3.4 Структура базы данных 22](#_Toc515313092)

[3.5 Подключение к базе данных 24](#_Toc515313093)

[3.6 Главная страница приложения 24](#_Toc515313094)

[3.7 Авторизация пользователя 26](#_Toc515313095)

[3.8 Регистрация пользователя 27](#_Toc515313096)

[3.9 Страница создания новых опросов 29](#_Toc515313097)

[3.10 Учет рабочего времени сотрудников 31](#_Toc515313098)

[3.11 Публикация веб-приложения на веб-сервер 33](#_Toc515313099)

[3.12 Вывод по разделу 33](#_Toc515313100)

[4 Руководство пользователя и тестирование 34](#_Toc515313101)

[4.1 Регистрация пользователя в системе 34](#_Toc515313102)

[4.2 Авторизация пользователя 35](#_Toc515313103)

[4.3 Руководство для пользователя 36](#_Toc515313104)

[4.3.1 Работа с заданиями 36](#_Toc515313105)

[4.3.2 Работа с опросами 37](#_Toc515313106)

[4.3.3 Работа с дополнительной информацией 38](#_Toc515313107)

[4.3.4 Регистрация проработанного времени 38](#_Toc515313108)

[4.4 Руководство для пользователя с правами администратора 39](#_Toc515313109)

[4.4.1 Посещаемость сотрудников 39](#_Toc515313110)

[4.4.2 Управление системой опросов 40](#_Toc515313111)

[4.5 Тестирование программного средства 41](#_Toc515313112)

[4.5.1 Тестирование полей авторизации 42](#_Toc515313113)

[4.5.2 Тестирование полей регистрации 43](#_Toc515313114)

[4.5.3 Создание новых опросов для пользователей 43](#_Toc515313115)

[4.5.4 Выводы по разделу 44](#_Toc515313116)

[5 Экономическое обоснование цены программного средства 45](#_Toc515313117)

[5.1 Характеристика программного средства 45](#_Toc515313118)

[5.2 Исходные данные и маркетинговый анализ 45](#_Toc515313119)

[5.3 Методика обоснования цены 46](#_Toc515313120)

[5.3.1 Объем программного средства 47](#_Toc515313121)

[5.3.2 Основная заработная плата 47](#_Toc515313122)

[5.3.3 Дополнительная заработная плата 48](#_Toc515313123)

[5.3.4 Отчисления в фонд социальной защиты населения 48](#_Toc515313124)

[5.3.5 Расходы на материалы 48](#_Toc515313125)

[5.3.6 Расходы на оплату машинного времени 49](#_Toc515313126)

[5.3.7 Прочие прямые затраты 49](#_Toc515313127)

[5.3.8 Накладные расходы 49](#_Toc515313128)

[5.3.9 Сумма расходов на разработку программного средства 49](#_Toc515313129)

[5.3.10 Расходы на сопровождение и адаптацию 50](#_Toc515313130)

[5.3.11 Полная себестоимость 50](#_Toc515313131)

[5.3.12 Определение цены, оценка эффективности 50](#_Toc515313132)

[5.4 Вывод по разделу 51](#_Toc515313133)

[ЗАКЛЮЧЕНИЕ 52](#_Toc515313134)

[Список используемых источников 53](#_Toc515313135)

[ПРИЛОЖЕНИЕ А 54](#_Toc515313136)

[ПРИЛОЖЕНИЕ Б 58](#_Toc515313137)

[ПРИЛОЖЕНИЕ В 61](#_Toc515313138)

[ПРИЛОЖЕНИЕ Г 64](#_Toc515313139)

[ПРИЛОЖЕНИЕ Д 69](#_Toc515313140)

[ПРИЛОЖЕНИЕ Е 70](#_Toc515313141)

[ПРИЛОЖЕНИЕ Ж 71](#_Toc515313142)

[ПРИЛОЖЕНИЕ З 72](#_Toc515313143)

[ПРИЛОЖЕНИЕ И 73](#_Toc515313144)

[ПРИЛОЖЕНИЕ К 74](#_Toc515313145)

ВВЕДЕНИЕ

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*5*

*ДП 00.00.ПЗ*

*Разраб.*

*Рауба А.А.*

*Провер.*

*.*

*Буснюк Н.Н.*

Консульт.

*Буснюк Н.Н.*

*Н. Контр.*

*Жигаровская С.А.*

*Утверд.*

*Смелов В.В.*

ВВЕДЕНИЕ

*Лит.*

*Листов*

*1*

БГТУ 74419017, 2018

У

Проблема контроля персонала – это та проблема, с которой очень часто обращаются руководители предприятий. Они зачастую не понимают, что, кого, когда и как контролировать для того, чтобы эффект от контроля был должным, но сотрудники не чувствовали себя под постоянным прицелом руководства.

У каждого руководителя бизнеса свой взгляд на проблему контроля персонала. Большинство управленцев осознают тот факт, что контролировать сотрудников и бизнес-процессы необходимо, но подходы к мониторингу сильно разнятся. Одни убеждены, что необходимо спрашивать только результат, другие предпочитают отслеживать все этапы выполнения заданий, а третьи и вовсе воспринимают контроль как шпионаж за сотрудниками и ловлю нарушителей дисциплины.

Руководители крупных организаций с большим штатом сотрудников не успевают лично уследить за каждым подчиненным. Кроме функции контроля менеджер, начальник отдела или руководитель проекта исполняет и другие обязанности. Во время работы возникают непредвиденные трудности или обстоятельства, не предусмотренные планом, и рабочий процесс полностью предугадать и распланировать не удается даже лидерам тайм-менеджмента.

Цель дипломного проекта: создание web-приложения, которое предусматривает систему учета информации о сотрудниках, учет проделанной работы, контроль посещаемости, построение графиков на основе собранной информации.

Задачи дипломного проекта:

* проектирование и создание базы данных;
* разработка web-приложения;
* обеспечить возможность регистрации и авторизации пользователей;
* обеспечить анкетирование новых сотрудников;
* реализовать построение графиков на основе полученной информации (анкеты, посещаемость и т.д.);
* обеспечить возможность анализировать и контролировать выполненные задачи сотрудников.

Практическая значимость: разработка приложения позволит руководителю собирать данные о сотрудниках, проводить сбор информации о проделанной работе, возможность контролировать посещаемость.

# Обзор теоретического материала и аналогов

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*6*

*ДП 01.00.ПЗ*

*Разраб.*

*Рауба А.А.*

*Провер.*

*.*

*Буснюк Н.Н.*

Консульт.

*Буснюк Н.Н.*

*Н. Контр.*

*Жигаровская С.А.*

*Утверд.*

*Смелов В.В.*

*Лит.*

*Листов*

*7*

БГТУ 74419017, 2018

## Контроль персонала

Контроль работы персонала – это одна из функций менеджера, которая заключается в сборе сведений о выполнении поставленных задач подчиненным. Контроль является жизненной необходимостью при работе с персоналом, поскольку на основании данных контроля осуществляется планирование дальнейшей работы. От того на сколько правильно организован контроль работы персонала в компании зависит на сколько быстро компания придёт к намеченным целям и то сколько ресурсов потратит компания. Изучая методы управления персоналом с точки зрения продаж, функции контроля нужно уделить особое внимание [1].

Контроль персонала всегда является обязательным условием. Но его степень и вариации могут различаться в зависимости от ситуации – может быть иногда мягче, а порой жестче. Безделье персонала может быть обусловлено двумя причинами. Первая из них – отсутствие интереса к своей работе. Когда человек действительно в чем-то заинтересован, он будет на этом сконцентрирован, забывая про всё остальное. Вторая причина обусловлена возможностью сотрудников избежать работы. По своей натуре, люди не перестанут бездельничать в то время, когда другие выполняют работу за них. Работа любой компании должна быть построена на принципе, позволяющем контролировать работу каждого сотрудника.

## Методы контроля персонала

Есть много разных форм контроля. Можно контролировать как текущий процесс, так и полученный результат, существует предварительный, текущий и итоговый контроль. Так же при контроле очень важно понимать какие вы затрачиваете ресурсы на организацию контроля и развивать в коллективе самоконтроль.

Предварительный или начальный контроль – вариант контроля, который заключается в обозначении сотруднику какие итоговые результаты вы хотите получить по выполнению поставленной задачи, а также определение временных ограничений. Предварительный контроль является частью правильной постановки задач и прежде всего формирует у подчиненного осознание того что за его работой следят, и она будет проконтролирована по выполнению. Этап предварительного контроля снижает затраты на текущий и итоговый контроль.

Текущий контроль – вариант контроля, который осуществляется непосредственно во время проведения работы, путем наблюдения за работой

У

Обзор теоретического материала и аналогов

сотрудника. При этом наблюдение может носить форму как совместной работы, так и оценки работы по определенному «чек – листу». Основная задача текущего контроля скорректировать работу сотрудника и повлиять на достижение цели. Считается что текущий контроль наиболее эффективно проводить либо по истечении 2/3 от поставленного времени, либо, когда работа выполнена на 2/3. Поскольку вы сможете не только увидеть недочеты в работе, но и успеете внести корректировки [1].

Итоговый контроль – на этапе итогового контроля сравниваются поставленные цели с полученным результатом. Итоговый контроль необходимо проводить всегда. Если сотрудник поймет, что вы не контролируете его работу, он рано или поздно начнет работать «спустя рукава». У сотрудника всегда должна быть ответственность за результат. У итогового контроля есть большой недостаток, он вам не даёт возможности справить выполнение работы так как время ушло. Поэтому не нужно пренебрегать текущим и предварительным контролем.

При проведении контроля нужно понимать, что большинство рядовых сотрудников относятся к контролю как к карательной мере, многие начинают сразу искать скрытый смысл и даже боятся что их уволят. При контроле сотрудники испытывают определенный стресс и чувство отторжения. Хотя сам по себе контроль создан для того чтобы помогать руководителю понять зоны роста. Поэтому при контроле старайтесь более позитивно общаться с подчиненным. Самый высший пилотаж использовать контроль как возможность мотивировать сотрудника на выполнение работы. Кроме того, результаты контроля служат данными для того, чтобы осуществить анализ продаж [1].

На данный момент, так как почти все компании использую информационные ресурсы и технику, существует вариант контроля персонала посредством программного обеспечения. На данный момент многие компании используют различные программы, позволяющие контролировать определенные моменты в работе сотрудников.

## Обзор системы контроля StaffCounter

StaffCounter – инструмент контроля за сотрудниками. Приложение позволяет мониторить текущую деятельность персонала за компьютером или на мобильном на базе Android и вести записи разговоров. Собранные данные помогают оценивать реальную эффективность подчинённых и принимать управленческие решения.

Платформа подойдёт как небольшим, так и крупным компаниям. С её помощью HR-менеджеры могут более объективно оценивать сотрудников, вести учёт рабочего времени, в том числе переработок и сверхурочных, а также контролировать деятельность фрилансеров и специалистов на аутсорсе. Полученные данные позволяют предупредить увольнения ценных членов коллектива и предотвратить забастовки и утечку информации (рисунок 1.1).

Платформа имеет как свои особенности, которые скорее являются плюсами, так и минусы, которые в какой-то мере ухудшают данное программное средство, что сказывается на популярности и востребованности в данной платформе

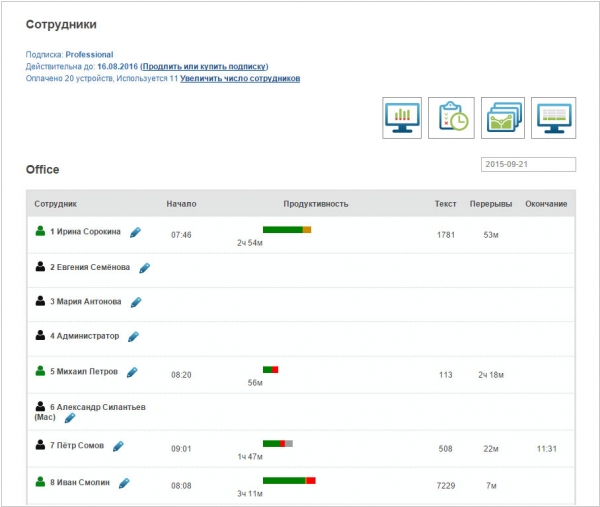


Рисунок 1.1 – Система контроля StaffCounter

StaffCounter автоматически фиксирует происходящее на компьютере или корпоративном смартфоне, включая время начала и окончания работы, а также перерывы. Уведомления о нарушениях режима формируются автоматически и избавляют от необходимости постоянной проверки отчётов. Мониторинг работы приложений, и их классификация по полезности исключает риск имитирования рабочейдеятельности. Софт позволяет делать снимки с экрана и веб-камеры, записывать звук с микрофона, а также фиксировать общение по SMS, Skype и Viber. Приложение работает в скрытом режиме, в том числе на мобильных и планшетах.

## Система учета рабочего времени Эпикур

**Программа учета рабочего времени Эпикур** предназначена для организации полнофункциональной системы контроля доступа на предприятии и учета рабочего времени персонала. Не требует покупки и подключения дополнительных модулей для расширения функционала, т.к. Эпикур не является модульной системой, все существующие функции (модули) доступны в стандартном комплекте. Простота установки, настройки, дружественный, интуитивно понятный интерфейс делают эту программу незаменимой в системах контроля доступа любой сложности. Данная программа является сетевой системой, работает под управлением Microsoft SQL Server 2005, 2008 и позволяет организовывать неограниченное число удаленных рабочих мест. Программа позволяет организовать полноценный учет рабочего времени на предприятии с поддержкой сложных графиков работы, смен, оправдательных документов, настраиваемых календарей, шаблонов и др. с итоговой выдачей **табеля учета рабочего времени** (рисунок 1.2)**.**



Рисунок 1.2 – Система учета Эпикур

Программное обеспечение Эпикур работает под управлением ОС Microsoft. Оно совместимо с мастером настройки импорта и экспорта, модулем интеграции и системой синхронизации баз данных. Также ПО совместимо с любыми необходимыми для обеспечения контроля доступа и учета рабочего времени, в том числе бухгалтерскими приложениями и расширениями. Обновление ПО осуществляется бесплатно через Интернет.

## Система учета рабочего времени Kickidler

Kickidler – система мониторинга деятельности персонала с возможностью наблюдения за действиями сотрудников на компьютерах и удаленного подключения к их рабочим столам. Правами на программу владеет сингапурская компания TeleLinkSoft, а эксклюзивной дистрибуцией Kickidler на территории России занимается компания «АйТи Сервис Менеджмент». Kickidler – относительно новый продукт на рынке программного обеспечения для учета рабочего времени сотрудников, обладающий уникальным функционалом, который не встречается у других подобных систем. Как и большинство других продуктов по контролю деятельности сотрудников, Kickidler умеет собирать информацию о том, с какими программами и веб-сайтами работал сотрудник и сколько времени он на это потратил. Но основной упор разработчики делают на возможность наблюдения в режиме реального времени за сотрудниками всей компании одновременно, а также записи видео их активности и фиксации нарушений (рисунок 1.3).

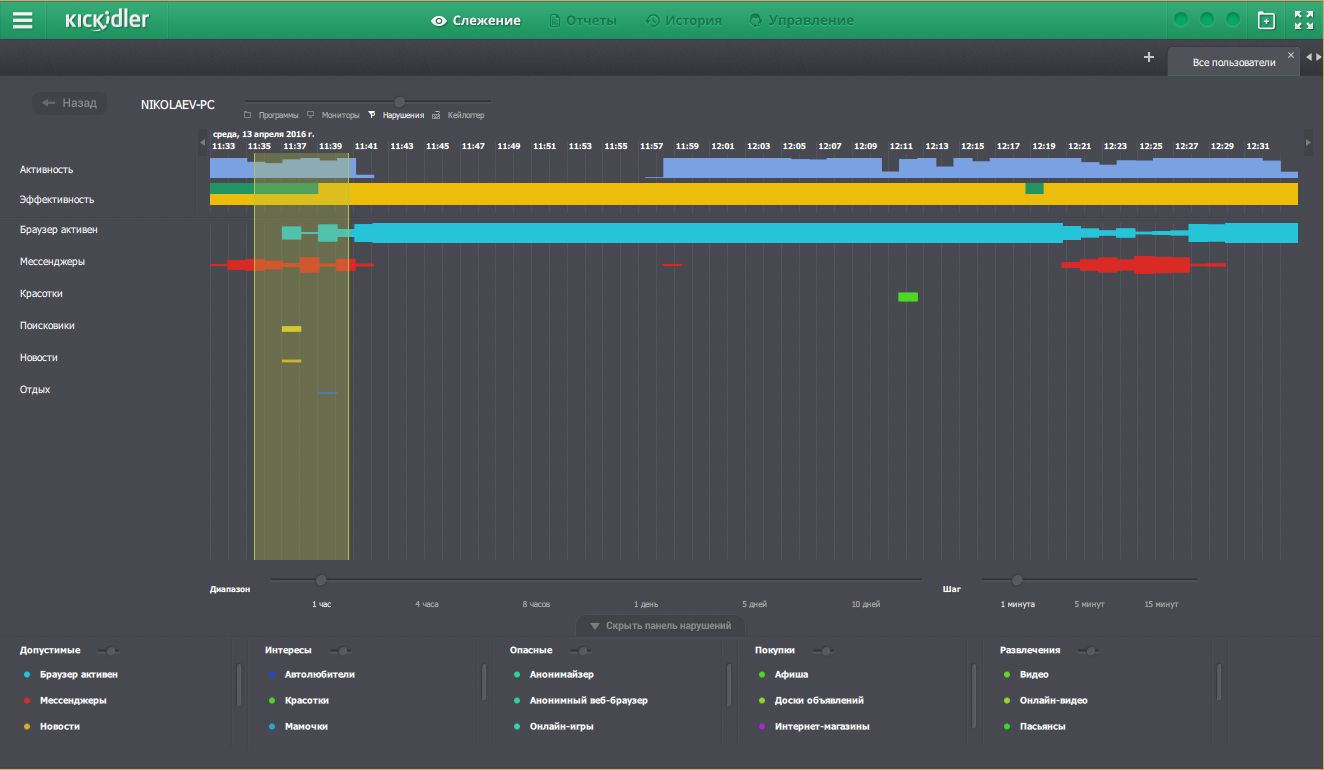


Рисунок 1.3 – Система учета Kickidler

Вся работа с системой Kickidler осуществляется через программу «Вьюер». Окно программы состоит из нескольких зон. Вверху находится панель, которая содержит четыре основных раздела – «Слежение», «Отчеты», «История» и «Управление», а также меню настройки комплекса и индикаторы состояния серверов системы. Слева доступно всплывающее меню выбора конфигурации отображения рабочих столов и настройки нарушений. Центральная область интерфейса программы предназначена для визуализации данных мониторинга и разноуровневой аналитики.

Как и большинство других систем, Kickidler умеет собирать информацию о том, с какими программами и сайтами работал сотрудник и сколько времени у него заняла эта работа.

Для каждого человека или группы людей (менеджеры, программисты, бухгалтерия) существует свой список продуктивных и непродуктивных программ и сайтов. Например, Facebook для SMM менеджера рабочий инструмент, а для бухгалтера – развлечение. На основе этого деления Kickidler отображает отчет об эффективности отдельных сотрудников, подразделений и всей компании и табель учета рабочего времени.

## Программа для учета рабочего времени Mipko PM

Mipko – абсолютный старожил на рынке систем контроля. Правда, изначально была создана программа для контроля домашних пользователей (в первую очередь детей) за ПК и только потом функционал был расширен до использования в коммерческом секторе.

Разработка ведется в Пскове. Помимо российского рынка, активно продается на западе, но почему-то под другим брендом Refog.

Интерфейс программы словно застыл в начале 2000-х. Именно тогда были популярны крупные кнопки в тулбарах, яркие иконки и меню в стиле Microsoft Office 2003.

На рынке родительского контроля компания признанный лидер, остальные системы домашнего мониторинга или слабее по функционалу (многочисленные кейлоггеры), или сложнее для установки (Kickidler), или вообще бесплатные (Стахановец). Разработке корпоративной версии уделяется меньше усилий. Например, Personal Monitor может контролировать компьютеры Mac, а Employee Monitor работает только под Windows. Последнее (на момент написания обзора) обновление для персональной версии вышло 20 мая, а для корпоративной 14 марта (рисунок 1.4).

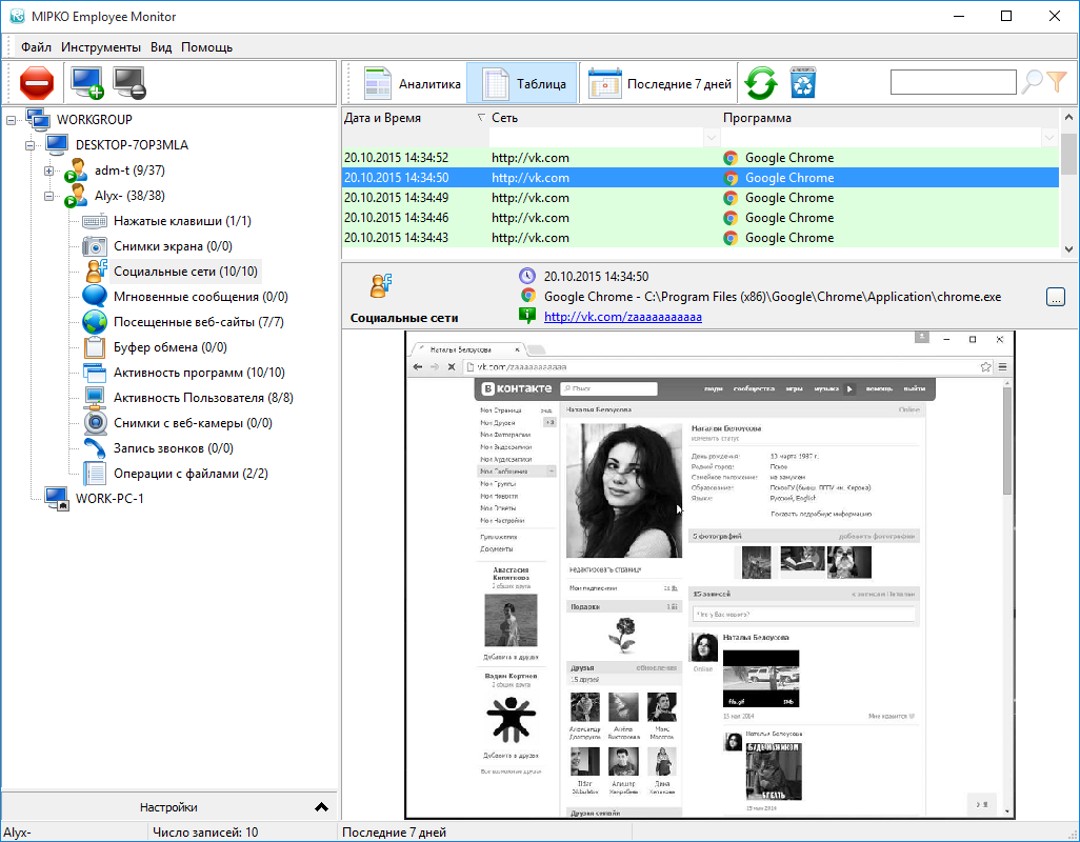


Рисунок 1.4 – Система учета рабочего времени Mipko

При этом Mipko честно не пытается казаться равным по функционалу таким системам, как Kickidler, StaffCop или Стахановец (а вот NeoSpy пытается). Его главный козырь простота для малых компаний. Поэтому его можно порекомендовать к использованию тем организациям, где число ПК не более 5 (в крайнем случае 10), все компьютеры находятся в одной сети, а системный администратор отсутствует.

Принцип работы программы: агент, устанавливаемый на каждый компьютер, собирает информацию о работе сотрудников. В отличие от большинства систем, данные никуда не передаются, а хранятся на ПК пользователей. Чтобы просмотреть отчеты, руководителю необходимо установить клиентскую программу, которая подключится ко всем наблюдаемым компьютерам и построит отчет.

Эта программа позволяет просматривать самую разнообразную информацию. Более подробно стоит остановиться на фиксировании скриншотов. Здесь все просто и в то же время очень интересно. В настройках можно выставлять значения периодичности совершения снимков, или они будут делаться при каждом переходу. Также здесь есть возможность настраивать качество снимков.

Программа создает отчеты в самых разнообразных форматах, что расширяет поле ее деятельности. В отчетах могут быть зафиксированы все действия, введенные пароли, посещенные сайты и т.д.

## Вывод по разделу

В данном разделе был произведен обзор теоретического материала о возможном сборе информации о сотрудниках и средствах контроля персонала.

Данный раздел также содержит информацию о аналогах разрабатываемого программного средства, был приведен кратки обзор к каждому из них, рассмотрены их возможности и недостатки, выделены плюсы и минусы.

В итоге был сделан вывод, что некоторые особенности схожих по функционалу уже созданных программных средств недостаточны для достижения требуемого результата, в результате чего было принято решение на создание собственной разработки в рамках дипломного проекта.

# Обоснование технических приемов программирования

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*13*

*ДП 02.00.ПЗ*

*Разраб.*

*Рауба А.А.*

*Провер.*

*.*

*Буснюк Н.Н.*

Консульт.

*Буснюк Н.Н.*

*Н. Контр.*

*Жигаровская С.А.*

*Утверд.*

*Смелов В.В.*

Обоснование технических приемов программирования

*Лит.*

*Листов*

*8*

БГТУ 74419017, 2018

У

## Платформа ASP.NET

Считается, что платформа .NET Framework явилась ответом компании Microsoft на набравшую к тому времени большую популярность платформу Java. ASP.NET основывается на Common Language Runtime: разработчики могут писать код для ASP.NET, используя практически любые языки программирования, некоторые из которых входят в комплект .NET Framework (C#, Visual Basic.NET и JavaScript. NET), а другие могут быть установлены дополнительно (IronRuby, IronPython, PHP, Perl, Smalltalk, Haskell и др.). Тем не менее, нередко используется не вполне корректный термин «NET-программист», который, видимо, характеризует разработчика на любом языке программирования, который может быть использован в .NET-проекте.

Некоторые особенности ASP.NET:

* компилируемый код выполняется быстрее, а большинство ошибок отлавливается ещё на стадии разработки;
* расширяемый набор элементов управления и библиотек классов, ускоряющий разработку;
* возможность кэширования всей страницы, её частей или данных, используемых на странице;
* возможность разделения визуальной части и бизнес-логики по разным файлам;
* расширяемые модели событий, обработки запросов и серверных элементов управления;
* поддержка CRUD-операций при работе с таблицами через GridView;
* возможно создание веб-приложений, которые реализуют шаблон Model-View-Controller (ASP.NET MVC Framework).

Платформа .NET достаточно широко распространена в сфере разработки внутрикорпоративных программных продуктов, но в веб-разработке всё же встречается относительно редко, как и другие программные продукты корпорации Microsoft. Поэтому найти разработчиков для веб-проекта бывает достаточно непросто. Использование .NET «тянет» за собой покупку и иного ПО от корпорации Microsoft (серверной ОС, СУБД и т.п.). Технология достаточно дорогая в разработке и сопровождении: кроме затрат на покупку лицензий на необходимое ПО существенный вклад в бюджет проектов вносят высокие зарплаты разработчиков [3].

## СУБД Microsoft SQL Server

Microsoft SQL Server – система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз, данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Сервер баз, данных Microsoft SQL Server в качестве языка запросов использует версию языка SQL, получившую название Transact-SQL (сокращённо T-SQL). Язык T-SQL является реализацией SQL-92 (стандарт ISO для языка SQL) с множественными расширениями. T-SQL позволяет использовать дополнительный синтаксис для хранимых процедур и обеспечивает поддержку транзакций (взаимодействие базы данных с управляющим приложением) [4].

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 16, которая вышла в 2016 году и которая будет использоваться в текущем руководстве.

SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.

SQL Server характеризуется такими особенностями как:

* SQL Server работает очень быстро;
* SQL Server предоставляет шифрование данных;
* с данной СУБД относительно легко работать и вести администрирование.

Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз, данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД [4].

Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта [4].

## Язык программирования C#

Язык C# и связанную с ним среду .NET Framework можно без преувеличения назвать самой значительной из предлагаемых в настоящее время технологий для разработчиков. Среда .NET является такой средой, которая была создана для того, чтобы в ней можно было разрабатывать практически любое приложение для запуска в Windows, а C# является языком программирования, который был специально создан для использования в .NET Framework. Например, с применением C# и .NET Framework можно создавать динамические веб-страницы, приложения Windows Presentation Faundation, веб-службы XML, компоненты для распределенных приложений, компоненты для доступа к базам данных, классические настольные приложения Windows и даже клиентские приложения нового интеллектуального типа, обладающие возможностями для работы в оперативном и автономном режимах [5].

Не стоит поддаваться заблуждению из-за наличия в названии Framework слова «NET» и думать, что данная среда предназначена только для создания приложений, ориентированных на Интернет. Слово "NET" здесь является лишь показателем того, что, по мнению Microsoft, распределенные приложения, в которых обработка распределяется между клиентом и сервером, являются шагом вперед. Однако важно понимать, что C# представляет собой язык, предназначенный не только для написания приложений, способных работать в Интернете и в сети. Он предоставляет средства для кодирования практически любого типа программного обеспечения или компонентов для платформы Windows. Язык C#, и среда .NET привели к революционным изменениям в способе написания разработчиками программ и сделали программирование приложений для Windows гораздо более простым, чем когда-либо [5].

C# – это относительно новый язык программирования, который характеризуется двумя следующими преимуществами:

Этот язык спроектирован и разработан специально для применения с Microsoft.NET Framework (развитой платформой разработки, развертывания и выполнения распределенных приложений).

Язык, основанный на современной объектно-ориентированной методологии проектирования, при разработке которого специалисты из Microsoft опирались на опыт создания подобных языков, построенных в соответствии с предложенными около 20 лет назад объектно-ориентированными принципами [5].

Нужно подчеркнуть то важное обстоятельство, что C# – это полноценный язык программирования. Хотя он и предназначен для генерации кода, выполняемого в среде .NET, сам по себе он не является частью .NET. Существует ряд средств, которые поддерживаются .NET, но не поддерживаются C#, и, возможно, вас удивит, что есть также средства, поддерживаемые C# и не поддерживаемые .NET (например, некоторые случаи перегрузки операций). Однако поскольку язык C# предназначен для применения на платформе .NET, вам, как разработчику, важно иметь представление о .NET Framework, если вы хотите эффективно разрабатывать приложения на C#.

Для понимания важности .NET не помешает вспомнить о природе многих технологий Windows, которые появились в последние примерно 18 лет. Хотя на первый взгляд все они могут выглядеть довольно разными, на самом деле все операционные системы Windows, начиная с Windows 3.1 (которая вышла в 1992 г.) и заканчивая Windows 7 и Windows Server 2008 R2, в основе своей имеют один и тот же хорошо знакомый API-интерфейс Windows. По мере появления новых версий Windows в этот API-интерфейс добавлялось много новых функций, но это был скорее процесс совершенствования и расширения API-интерфейса, а не его замена [5].

## Язык SQL

SQL (Structured Query Language – Структурированный язык запросов) – язык управления базами данных для реляционных баз данных. Сам по себе SQL не является Тьюринг-полным языком программирования, но его стандарт позволяет создавать для него процедурные расширения, которые расширяют его функциональность до полноценного языка программирования.

Язык был создан в 1970х годах под названием «SEQUEL» для системы управления базами данных (СУБД) System R. Позднее он был переименован в «SQL» во избежание конфликта торговых марок. В 1979 году SQL был впервые опубликован в виде коммерческого продукта Oracle V2.

Первый официальный стандарт языка был принят ANSI в 1986 году и ISO – в 1987. С тех пор были созданы еще несколько версий стандарта, некоторые из них повторяли предыдущие с незначительными вариациями, другие принимали новые существенные черты.

Несмотря на существование стандартов, большинство распространенных реализаций SQL отличаются так сильно, что код редко может быть перенесен из одной СУБД в другую без внесения существенных изменений. Это объясняется большим объемом и сложностью стандарта, а также нехваткой в нем спецификаций в некоторых важных областях реализации.

SQL создавался как простой стандартизированный способ извлечения и управления данными, содержащимися в реляционной базе данных. Позднее он стал сложнее, чем задумывался, и превратился в инструмент разработчика, а не конечного пользователя. В настоящее время SQL (по большей части в реализации Oracle) остается самым популярным из языков управления базами данных, хотя и существует ряд альтернатив [6].

## Язык разметки HTML5

Hyper Text Markup Language (HTML) – язык разметки гипертекста – предназначен для написания гипертекстовых документов, публикуемых в World Wide Web.

Гипертекстовый документ – это текстовый файл, имеющий специальные метки, называемые тегами, которые впоследствии опознаются браузером и используются им для отображения содержимого файла па экране компьютера.

С помощью этих меток можно выделять заголовки документа, изменять цвет, размер и начертание букв, вставлять графические изображения и таблицы. Но основным преимуществом гипертекста перед обычным текстом является возможность добавления к содержимому документа гиперссылок – специальных конструкций языка HTML, которые позволяют щелчком мыши перейти к просмотру другого документа.

HTML-документ состоит из двух частей: собственно, текста, т.е. данных, составляющих содержимое документа, и тегов – специальных конструкций языка HTML, используемых для разметки документа и управляющих его отображением. Теги языка HTML определяют, в каком виде будет представлен текст, какие его компоненты будут исполнять роль гипертекстовых ссылок, какие графические или мультимедийные объекты должны быть включены в документ.

Графическая и звуковая информация, включаемая в HTML-документ, хранится в отдельных файлах. Программы просмотра HTML-документов (браузеры) интерпретируют флаги разметки и располагают текст и графику на экране соответствующим образом. Для файлов, содержащих HTML-документы приняты расширения .htm или .html [7].

## Библиотека jQuery

Если вкратце охарактеризовать то, что делает библиотека jQuery, то это прозвучит довольно обыденно: данная библиотека позволяет изменять содержимое HTML-документов путем манипулирования объектами модели, создаваемой браузерами в процессе обработки HTML-кода (так называемые DOM-манипуляции).

Часто приходится манипулировать объектами DOM (Document Object Model – объектная модель документа) с помощью других библиотек JavaScript или встроенных функций API (Application Programming Interface – интерфейс программирования приложений), которые поддерживаются большинством современных браузеров. Однако библиотека jQuery работает намного лучше. Она превращает манипулирование объектами DOM в увлекательное занятие, временами напоминающее настоящее развлечение.

В работе с jQuery есть нечто настолько элегантное и притягательное, что задачи, решение которых обычно требует выполнения множества скучных рутинных операций, внезапно чрезвычайно упрощаются, так что, начав использовать jQuery, вы уже не сможете от этого отказаться.

Средства jQuery необычайно выразительны. Эта библиотека позволяет добиться гораздо большего при намного меньшем объеме кода, чем в случае использования программных DOM-интерфейсов браузеров.

Методы jQuery применимы к целым группам элементов. Предлагаемый в DOM-модели стандартный подход, основанный на шаблонной цепочке действий "выбрать-повторить-изменить", больше не требуется. Следствием этого является уменьшение количества циклов for в коде, а значит, и снижение вероятности появления в нем ошибок.

Библиотека jQuery справляется с различиями в реализации DOM в различных браузерах (проблемы кросс-браузерности). Например, меня не должна беспокоить мысль об особенностях поддержки того или иного средства, чем печально славится браузер Internet Explorer (IE). Достаточно всего лишь сформулировать jQuery свои пожелания, и библиотека самостоятельно обеспечит совместимость с конкретным браузером.

Библиотека jQuery имеет открытый исходный код. Если принципы работы какого-либо средства для меня не совсем ясны или получаемый результат не совпадает с ожидаемым, я могу обратиться непосредственно к коду библиотеки на JavaScript и, если это необходимо, внести соответствующие изменения.

Здесь также рассматриваются библиотека jQuery UI – это набор инструментальных средств, который предназначен для создания универсальных пользовательских интерфейсов и может применяться на любых устройствах [8].

## Язык CSS

CSS это язык стилей, определяющий отображение HTML-документов. Например, CSS работает с шрифтами, цветом, полями, строками, высотой, шириной, фоновыми изображениями, позиционированием элементов и многими другими вещами.

HTML может (неправильно) использоваться для оформления Web-сайтов. Но CSS предоставляет большие возможности и более точен, и проработан. CSS, на сегодняшний день, поддерживается всеми браузерами (программами просмотра).

Прочитав лишь несколько уроков этого учебника, вы сможете создавать собственные таблицы стилей и использовать CSS для придания вашему Web-сайту великолепного вида.

HTML используется для структурирования содержимого страницы. CSS используется для форматирования того же структурированного содержимого.

Согласен, это звучит как-то заумно. Но, пожалуйста, читайте дальше. Вскоре всё начнёт проясняться.

По мере развития Web дизайнеры начали искать возможности форматирования онлайновых документов. Чтобы удовлетворить возросшим требованиям потребителей, производители браузеров (тогда - Netscape и Microsoft) изобрели новые HTML-тэги, такие, например, как <font>, которые отличались от оригинальных HTML-тэгов тем, что они определяли внешний вид, а не структуру.

Это также привело к тому, что оригинальные тэги структурирования, такие как <table>, стали всё больше применяться для дизайна страниц вместо структурирования текста. Многие новые тэги дизайна, такие как <blink>, поддерживались только одним браузером. "Вам необходим браузер X для просмотра этой страницы" - такой отказ стал обычным явлением на web-сайтах.

CSS был создан для исправления этой ситуации путём предоставления web-дизайнерам возможностей точного дизайна, поддерживаемых всеми браузерами. Одновременно произошло разделение представления и содержимого документа, что значительно упростило работу.

## Библиотека Morris

Morris– это легковесный JavaScript (всего-то 3052 байта после миниатюризации) с открытым исходным кодом (распространяемым по упрощённой лицензии BSD), который для работы требует jQuery и Raphaël и строит с их помощью графики на простой сетке горизонтальных линий (рисунок 2.1).

Эта библиотека используя JavaScript строит различного вида графики, диаграммы, которые программист может сам настраивать под себя и под заказчика. Библиотека пользуется большой популярностью.

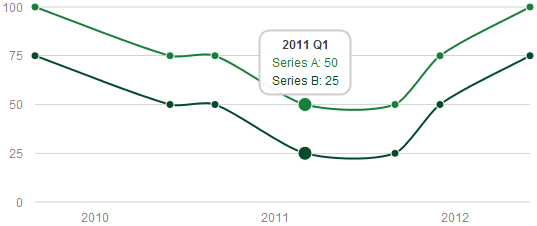


Рисунок 2.1 – Пример построения линейного графика

По оси абсцисс откладывается время, по оси ординат – какие-нибудь зависящие от времени значения. (Morris изначально разрабатывался для сайта, показывающего общее число автомашин той или иной марки в Великобритании, так что для него естественно, что ось абсцисс – это ось времени).

Графики реагируют на мышь: вспучиваются точки, соответствующие указанному мышью моменту во времени, и подле них появляются подсказки. Достоинство скрипта – простота API [9].

## Паттерн проектирования MVC

Платформа ASP.NET MVC представляет собой Framework для создания сайтов и веб-приложений с помощью реализации паттерна MVC.

Концепция паттерна (шаблона) MVC (model-view-controller) предполагает разделение приложения на три компонента:

Контроллер представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.

Представление – это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.

Модель – это элемент приложения, который представляет класс, описывающий логику используемых данных.

Благодаря этому реализуется концепция разделение ответственности, в связи с чем легче построить работу над отдельными компонентами. Кроме того, вследствие этого приложение обладает лучшей тестируемостью. И если нам, допустим, важна визуальная часть или фронтэнд, то мы можем тестировать представление независимо от контроллера. Либо мы можем сосредоточиться на backend и тестировать контроллер.

Конкретные реализации и определения данного паттерна могут отличаться, но в силу своей гибкости и простоты он стал очень популярным в последнее время, особенно в сфере веб-разработки.

Свою реализацию паттерна представляет платформа ASP.NET MVC. 2013 год ознаменовался выходом новой версии ASP.NET MVC - MVC 5, а также релизом Visual Studio 2013, которая предоставляет инструментарий для работы с MVC5.

Хотя во многих аспектах MVC 5 не слишком сильно будет отличаться от MVC 4, многое из одной версии вполне применимо к другой, но в то же время есть и существенные отличия:

* вместо SimpleMembershipProvider была внедрена система ASP.NET Identity, которая использует компоненты OWIN и Katana;
* для создания адаптивного и расширяемого интерфейса в MVC 5 используется css-фреймворк Bootstrap;
* добавлены фильтры аутентификации, а также появилась функциональность переопределения фильтров;
* в MVC 5 также добавлены атрибуты маршрутизации.

Это наиболее важные нововведения в MVC 5. Кроме того, есть еще ряд менее значимых, например, использование по умолчанию Entity Framework 6, некоторые изменения при создании проекта (концепция One ASP.NET), дополнительные компоненты и т.д.

В любом случае все полученные при работе с MVC 4 навыки можно успешно применять при использовании MVC 5, учитывая, конечно, нововведения [10].

## Вывод по разделу

В результате обзора технических средств и анализа поставленной задачи для реализации дипломного проекта была выбрана платформа ASP.NET. В качестве системы управления базой данных выбор был сделан в пользу Microsoft SQL Server. Основой архитектуры проекта был выбран паттерн проектирования MVC. Выбранный стек технологий позволит разработать гибкую систему способную работать без высоких нагрузок на систему.

# Проектирование и разработка программного средства

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*21*

*ДП 03.00.ПЗ*

*Разраб.*

*Рауба А.А.*

*Провер.*

*.*

*Буснюк Н.Н.*

Консульт.

*Буснюк Н.Н.*

*Н. Контр.*

*Жигаровская С.А.*

*Утверд.*

*Смелов В.В.*

Проектирование и разработка программного средства

*Лит.*

*Листов*

*13*

БГТУ 74419017, 2018

У

## Проектирование программного средства

Для реализации дипломного проекта были поставлены следующие задачи:

* спроектировать и разработать базу данных проекта;
* спроектировать и разработать пользовательский интерфейс;
* спроектировать и разработать интерфейс администратора.

## Структура приложения

Перед тем, как приступить к разработке программного средства, необходимо продумать структуру проекта, это позволит быстрее ориентироваться в коде, редактировать, вносить поправки, дописывать функционал. Файловая структура основных директорий проекта представлена на рисунке 3.1.

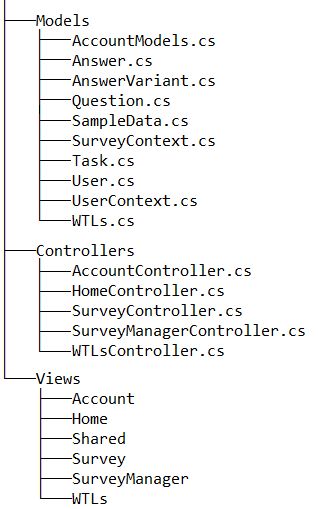


Рисунок 3.1 – Структура программного средства

Каталог models содержит классы, которые описывают логику используемых данных, controllers – каталог, который содержит файлы функционала приложения, views содержит страницы пользовательского интерфейса приложения.

## Разработка диаграммы вариантов использования

Перед тем как приступить к разработке программного средства необходимо определить цели, задачи и возможности использования ПО. Чтобы определить возможности программы, необходимо построить диаграмму, которая в свою очередь будет отображать возможности пользователей с разными привилегиями.

Диаграмма ролей (диаграмма вариантов использования) в UML – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы (часть ее функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов ее использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

Основное назначение диаграммы – описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

В результате была разработана диаграмма ролей, на которой изображен функционал разрабатываемого программного средства. С ее использованием будет проходить дальнейшая разработка веб-приложения (рисунок 3.2).

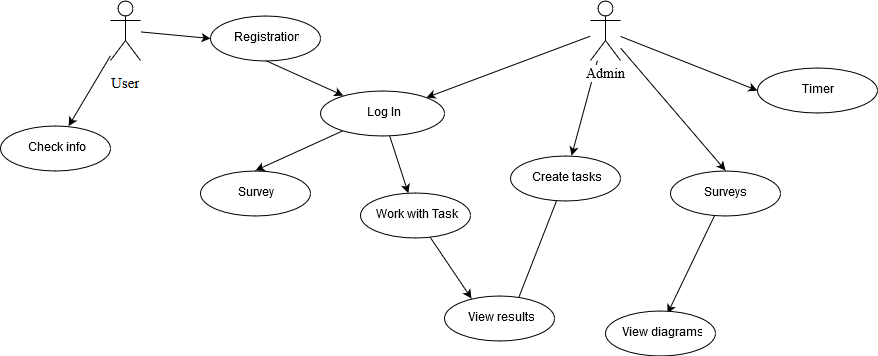


Рисунок 3.2 – Диаграмма ролей

Такая диаграмма полностью раскрывает все возможности всех пользователей с разными правами.

## Структура базы данных

При разработке программного средства, вся информация, которая заполняется пользователями и регистрируется программой должна храниться в определенных таблицах базы данных. Учитывая эту ситуацию были созданы таблицы базы данных, которые будут включать определенные записи:

* таблица Tasks для информации о заданиях;
* таблица Users для информации о пользователях;
* таблица WTLs для информации о посещаемости.

Таблица «Tasks» используется для хранения данных о заданиях.

Все таблицы хранятся в базе данных на Microsoft SQL Server 2014, что позволяет удаленно и независимо работать с отдельными записями таблиц, или самими таблицами.

Таблица 3.1 – Структура таблицы «Tasks»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | int | ID пользователя |
| Name | nvarchar | Наименование задания |
| E-mail | nvarchar | E-mail пользователя |
| TheTask | nvarchar | Принадлежность |
| Status | nvarchar | Статус задания |

Таблица «Users» будет содержать информацию о данных регистрации каждого из пользователей.

Таблица 3.2 – Структура таблицы «Users»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | int | ID пользователя |
| Name | nvarchar | Имя пользователя |
| E-mail | nvarchar | E-mail пользователя |
| Password | nvarchar | Пароль |
| Age | int | Возраст |

Таблица «WTLs» будет содержать информацию о данных посещаемости каждого из пользователей. При авторизации пользователя будет происходить запись и обновление данных.

Таблица 3.3 – Структура таблицы «WTLs»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | int | ID пользователя |
| Name | nvarchar | Имя пользователя |
| E-mail | nvarchar | E-mail пользователя |
| Come | datetime | Вошел в систему |
| Gone | datetime | Вышел из системы |
| TSpanH | int | Время в часах |
| TSpanM | int | Время в минутах |

После создания таблиц базы данных, можно объединить их в структуру, которая отобразит все связи между таблицами (рисунок 3.3).

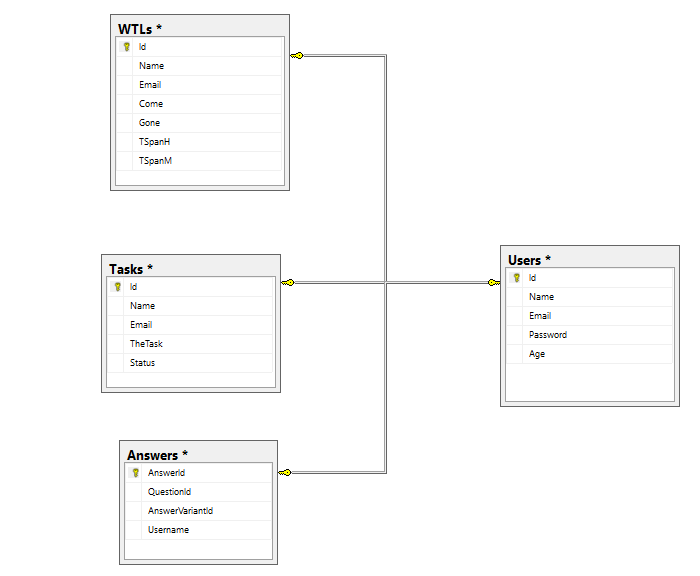


Рисунок 3.3 – Структура базы данных приложения

Все таблицы были созданы на Microsoft SQL Server 2014, что позволит работать с базой данных вне зависимости от приложения. Все таблицы соответствуют созданным в приложении классам моделей.

## Подключение к базе данных

Подключение к базе данных в программе будет происходить посредством технологии Entity Framework. Эта технология позволяет быстро и удобно получать данные и управлять ими в созданной базе. База данных для приложения уже создана и добавлена в приложение с помощью подхода Database first.

Для реализации данного подхода необходимо добавить в проект компонент «ADO.NET Entity Data Model». После указания используемых таблиц, будет автоматически создана модель базы данных, соответствующие классы и контекст базы данных.

Сама база данных будет хранится на MS SQL Server 2014, в которую можно войти только под определенным логином и паролем, что обеспечит защиту от несанкционированного доступа.

## Главная страница приложения

Главная страница приложения содержит в себе данные о посещаемости пользователей, однако, чтобы ее увидеть, необходимо сперва зарегистрироваться или авторизоваться в приложении (рисунок 3.4). Функционал на странице тоже доступен только авторизованным пользователям. Ограничение доступа сделано для того, чтобы никто, кроме пользователей с правами администратора пользователей не смог получить доступ к информации о посещаемости сотрудников.



Рисунок 3.4 – Экранная копия главной страницы

Данная страница использует контроллер HomeController. Сам контроллер реализован с использованием методов для работы с заданиями, каждый метод несет отдельный функционал.

Метод Add позволяет администратору создавать новые задачи для сотрудников, тем самым создавая занятость.

Метод EditTask позволяет пользователям менять статус выполнения задания, а администратору редактировать условия.

Метод Delete позволит удалить уже завершенные задания.

При добавлении нового задания у нас создается объект базы данных, в котором идет добавление новой записи. При редактировании задания, сперва происходит проверка на уже существующие записи, что позволит произвести поиск данных для обновления (рисунок 3.5).

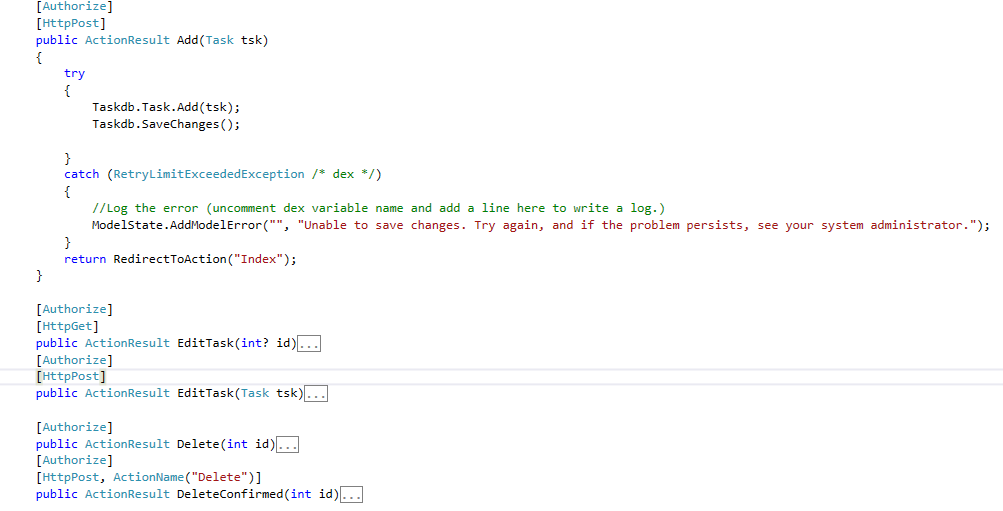


Рисунок 3.5 – Функционал работы с заданиями сотрудников

Для реализации данной страницы была построена таблица с не которым функционалом для пользователей (рисунок 3.6). Вывод результата на странице реализовано с помощью таблицы и модели данных, которая создавалась для работы с данными в таблице на сервере.



Рисунок 3.6 – Код страницы с заданиями сотрудников

При использовании User.Identity мы ограничиваем возможность доступа к странице неавторизованным пользователям, или пользователям с ограниченным доступом. Исходный код контроллера для работы с заданиями представлен в приложении Б.

Данная страница создана таким образом, чтобы обычный незарегистрированный пользователь не смог получить доступ к данным на этой странице. Также каждый пользователь имеет право изменять статус только своего задания. Данные о заданиях хранятся в таблице базы данных.

## Авторизация пользователя

Авторизация пользователя содержит поля для ввода почтового адреса и пароль, который он задавал при регистрации в приложении (рисунок 3.7).

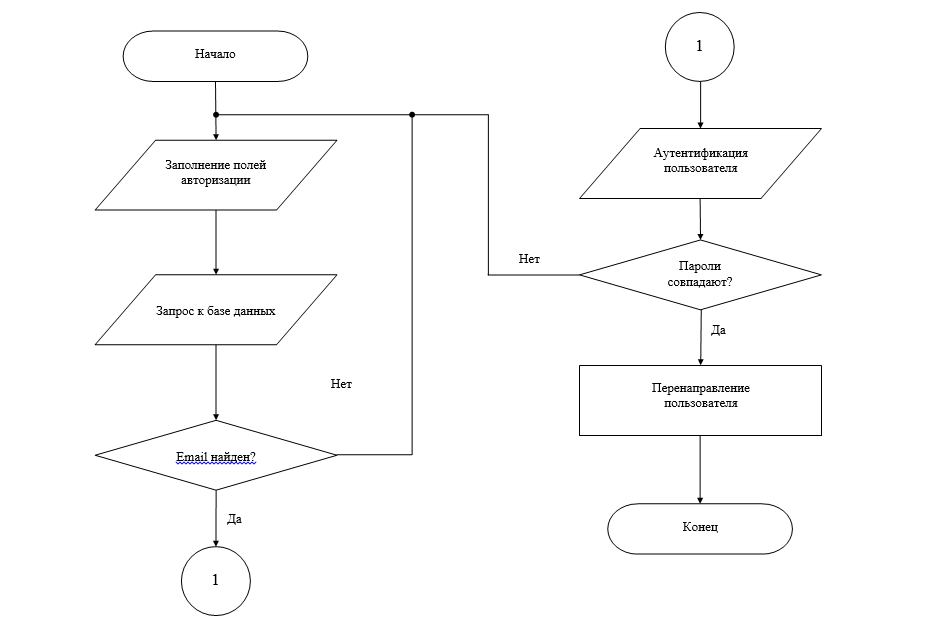


Рисунок 3.7 – Авторизация пользователя в системе

Сама же страница содержит код для ввода двух полей: имени пользователя и пароля. При вводе неверных данных пользователь получит информацию о неверно введенных данных (рисунок 3.8).



Рисунок 3.8 – Разметка авторизации в системе

Авторизация содержит сообщения, которые отобразятся в случае неверных данных. Также для удобства пользователя тут содержится кнопка регистрации. Исходный код контроллера с авторизацией представлен в приложении А.

## Регистрация пользователя

Регистрация пользователя выглядит примерно так же, как и авторизация, за исключением некоторых новых полей для ввода (рисунок 3.9).

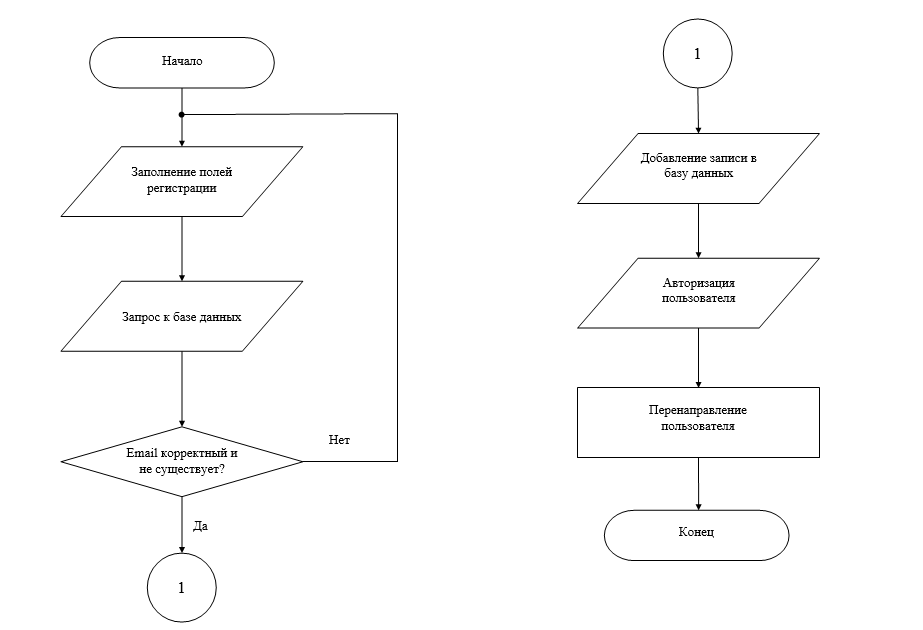


Рисунок 3.9 – Регистрация пользователя в системе

Исходный код регистрации содержит валидацию вводимых значений для каждого из полей. Это позволяет предупредить пользователей, если они вводят некорректные данные в заполняемы поля для регистрации в системе (рисунок 3.10).

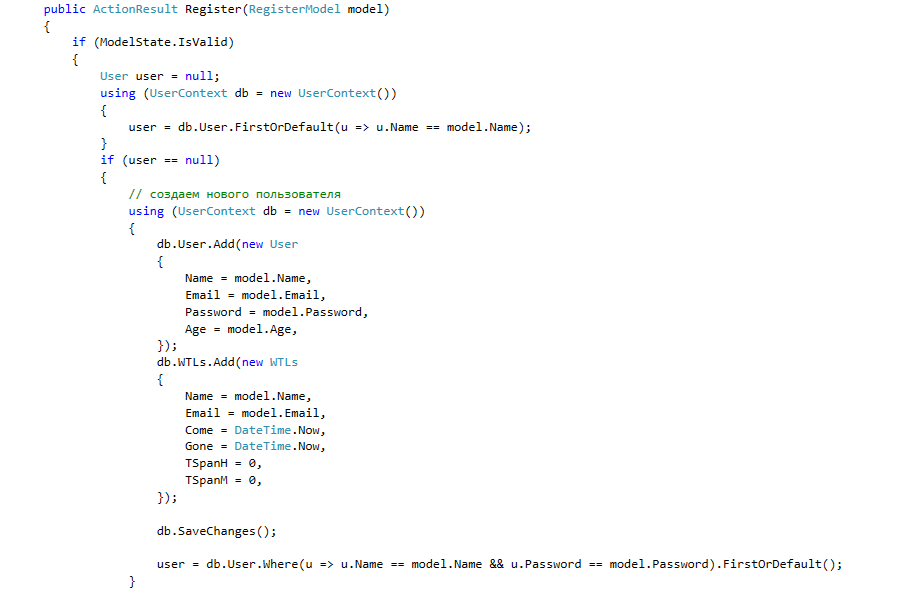


Рисунок 3.10 – Исходный код регистрации пользователя

Данный метод в контроллере сохраняет введенные данные с полей в таблицу нашей базы данных, при этом учитывает время регистрации, потому что после регистрации будет происходить авторизация в системе, а это позволяет начать регистрировать начало рабочего времени. Разметка страницы регистрации представлена на рисунке 3.11.

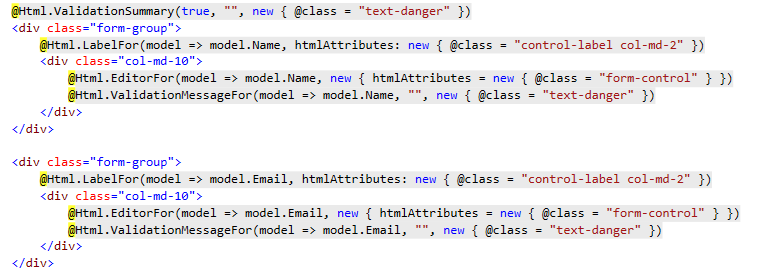


Рисунок 3.11 – Разметка страницы регистрации

Страница регистрации содержит поля для ввода данных пользователя с проверкой для оповещения о неверно введенных данных. Исходный код контроллера с регистрацией представлен в приложении Ж.

## Страница создания новых опросов

Каждый пользователь после регистрации в программе должен пройти определенный опрос, который содержит некоторое количество вопросов, которые может создавать администратор приложения (рисунок 3.12).

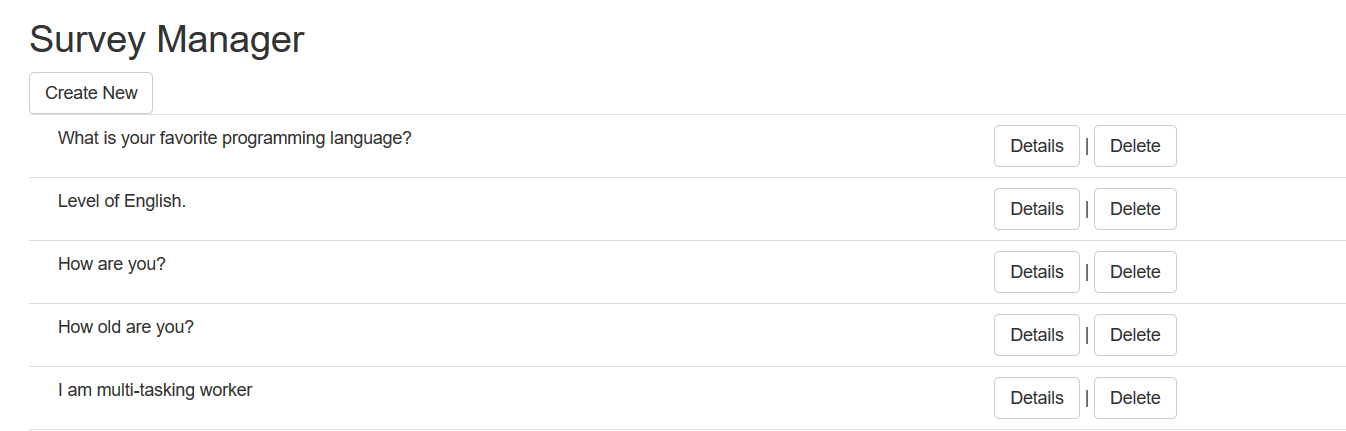


Рисунок 3.12 – Страница опроса

Количество создаваемых вопросов неограничено, что дает возможность собрать полную информацию о сотрудниках. Сами вопросы могут иметь различное количество вариантов ответа, но минимальное количество равно двум. Максимально количество вариантов ответа администратор выбирает сам, но не более 9, это позволит обезопасить создание бесконечного количества вариантов ответа. При вводе менее двух вариантов ответа, пользователь получит предупреждение о необходимости заполнения полей (рисунок 3.13).

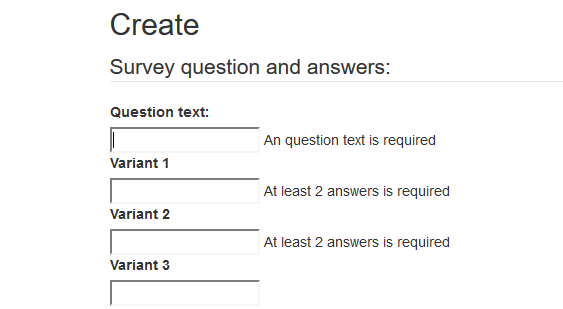


Рисунок 3.13 – Создание нового опроса для пользователей

Файл с кодом для создания определенного вопроса содержит созданные поля для самого вопроса и для некоторого количества вариантов ответа. (рисунок 3.14). Исходный код контроллера для работы с заданиями представлен в приложении Г

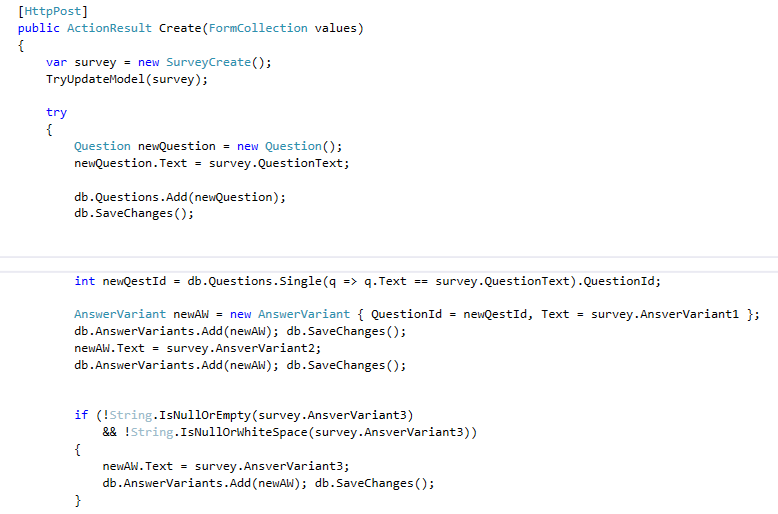


Рисунок 3.14 – Исходный код создания вопроса

При открытии страницы по созданию опроса нам предлагает ввести данные вопроса и его вариантов ответа. Для этого в файле мы программно создаем поля для ввода, указывая определенные ограничения и валидацию для обязательного заполнения двух вариантов ответа (рисунок 3.15).

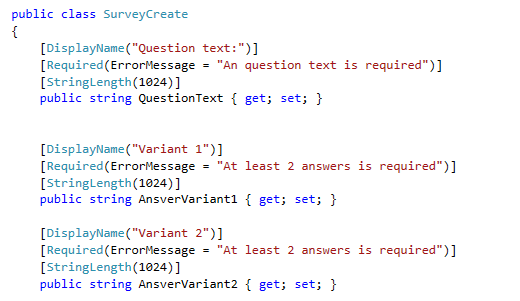


Рисунок 3.15 – Код создания нового вопроса

Сами вопросы хранятся в файле базы данных, для того, чтобы все варианты ответов для каждого из пользователей были сохранены и отображены при повторном запуске приложения. Для работы с базой данных были созданы модели, на основе которых и строится сама база (рисунок 3.16).

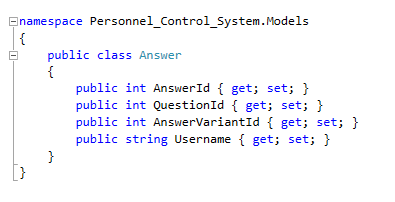


Рисунок 3.16 – Модель данных создаваемого вопроса

Все поля, которые создаются на сервере БД в таблице, будут аналогичны тем, что указаны в модели данных. Это позволит напрямую соединять приложение с базой. Исходный код контроллера для работы опросами представлен в приложении В.

## Учет рабочего времени сотрудников

Для обычного пользователя данная функция не доступна, т.к. только руководитель обязан контролировать такие данные (рисунок 3.17).

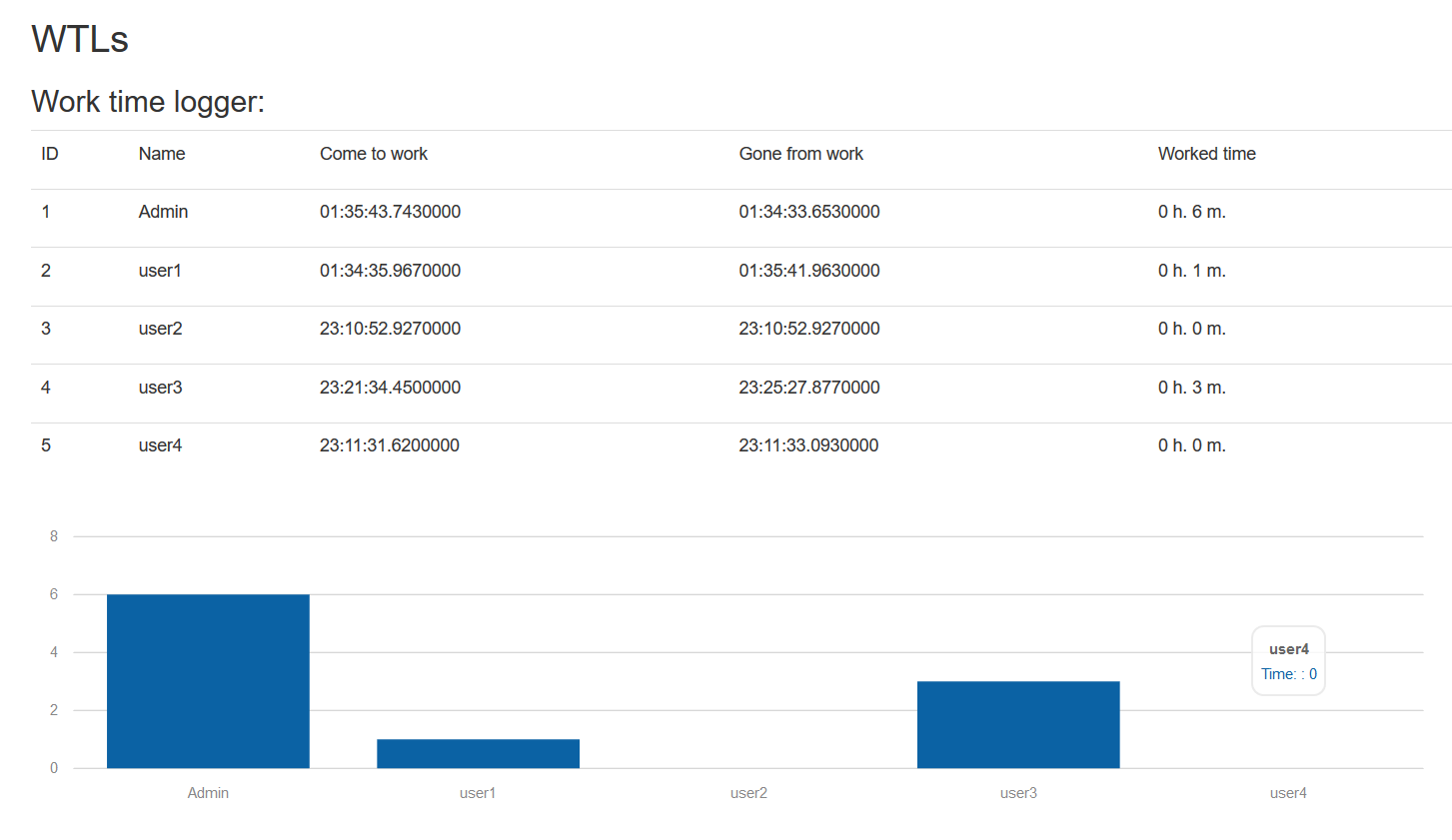


Рисунок 3.17 – Учет рабочего времени сотрудников

Данный скриншот показывает, что данные о посещаемости сохраняются в таблицу базы данных и на основе этой таблицы у нас строится график посещаемости для каждого пользователя за текущий день.

Исходный код для построения такого графика содержит JS, в который передаются данные из таблицы (рисунок 3.18).

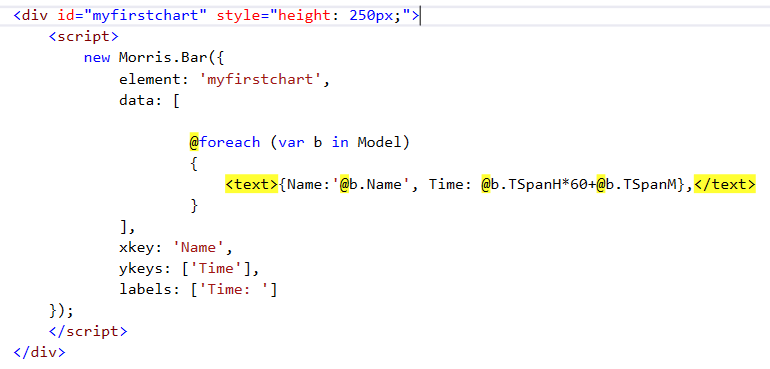


Рисунок 3.18 – Построение графика посещаемости сотрудников

В скрипт мы передаем имя и время в часах и минутах, при этом учитываем, чтобы минуты сбрасывались при достижении значения 60, но при этом значение часа будет увеличиваться на единицу. График будет содержать имя каждого пользователя и его посещаемость за текущий день.

График можно создавать любого вида, от обычного линейного графика, до диаграммы. На рисунке 19 показана схема работы приложения.

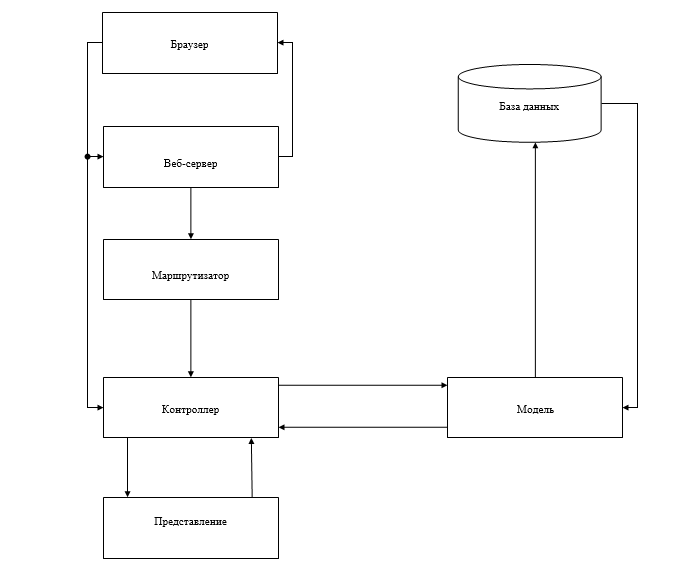


Рисунок 3.19 – Схема работы приложения

В итоге получили веб-приложение, которое будет работать по определенной схеме благодаря использованию паттерна MVC5.

## Публикация веб-приложения на веб-сервер

Традиционно somee.com применялся для развертывания веб-приложений. Для хостирования веб-приложений ASP.NET также может применяться IIS, только в отличие от предыдущих версий ASP.NET теперь его роль будет сводиться к прокси-серверу. Хостирование приложений ASP.NET на somee.com происходит с после регистрации аккаунта на сайте и ввода первичной информации о приложении, которое в свою очередь будет опубликовано через VS2015. Публикация через VS2015 позволит опубликовать приложение на удаленный сервер, указав регистрационные данные и данные для публикации.

После завершения создания приложения оно должно быть опубликовано на сервере, чтобы все пользователи смогли получить доступ. Для публикации будет использован бесплатный сервер somee.com (рисунок 3.20).

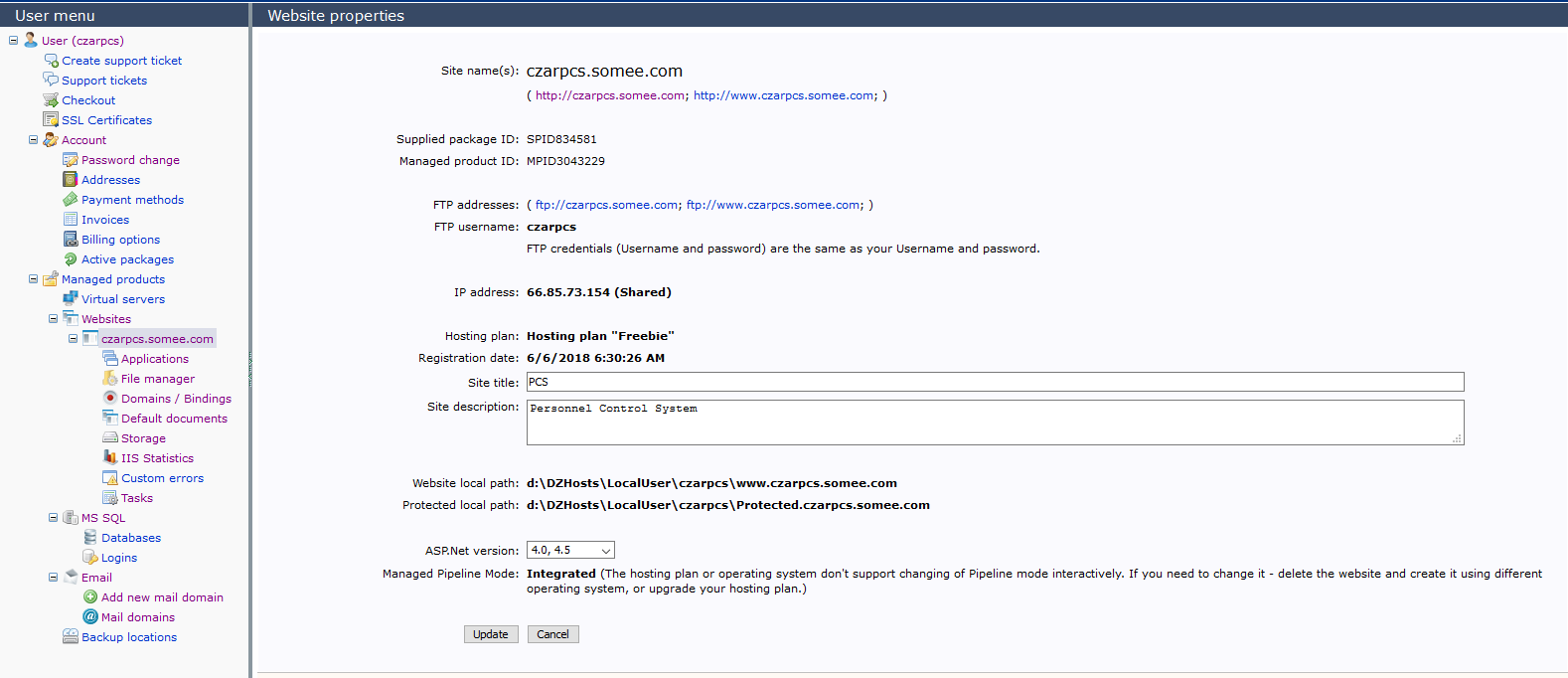


Рисунок 3.20 – Публикация на сервер somee.com

После публикации приложение будет доступно пользователям по локальной сети по соответствующему адресу, который был получен после публикации на сервере.

Такой способ настройки приложения позволяет быстро развернуть программное средство на одном сервере, для того, чтобы каждый пользователь смог подключиться к нему со своего рабочего компьютера.

## Вывод по разделу

Данный раздел содержит все этапы моделирования и разработки приложения. В результате было разработано web-приложение, которое позволяет собирать информацию о сотрудниках и проводить некоторый анализ на основе собранной информации. Сравнивая с рассматриваемыми аналогами данное приложение отличается повышенной скоростью работы. Это приложение не нужно устанавливать на все рабочие компьютеры, достаточно развернуть его на сервере.

Приложение легко расширяется и дополняется новыми возможностями и функционалом благодаря паттерну MVC.

# Руководство пользователя и тестирование

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*34*

*ДП 04.00.ПЗ*

*Разраб.*

*Рауба А.А.*

*Провер.*

*.*

*Буснюк Н.Н.*

Консульт.

*Буснюк Н.Н.*

*Н. Контр.*

*Жигаровская С.А.*

*Утверд.*

*Смелов В.В.*

Руководство пользователя и тестирование

*Лит.*

*Листов*

*10*

БГТУ 74419017, 2018

У

## Регистрация пользователя в системе

Регистрация – очень важна для многих веб-приложений. Для организации различных манипуляций на сайте, каждому пользователю необходимо предоставить возможность зарегистрировать свой аккаунт в системе (рисунок 4.1).

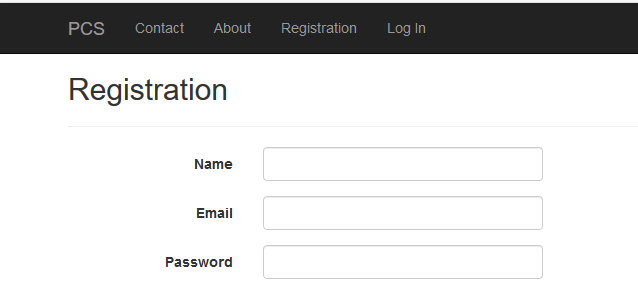


Рисунок 4.1 – Регистрация пользователя

При регистрации пользователь проходит несколько этапов, чтобы создать собственную учетную запись в системе, это позволит в дальнейшем без проблем получать доступ к необходимой информации.

К этапам можно отнести:

* ввод регистрационных данных;
* запрос к базе данных приложения;
* проверка на корректность и повтор email адреса;
* добавление записи в таблицу на сервере;
* перенаправление пользователя.

Все поля контролируются на заполнение данными, если не ввести данные в поле, у нас появится предупреждение напротив незаполненного поля (рисунок 4.2). Такое предупреждение оповестит пользователя, что он ввел неверно, или какое поле забыл заполнить.

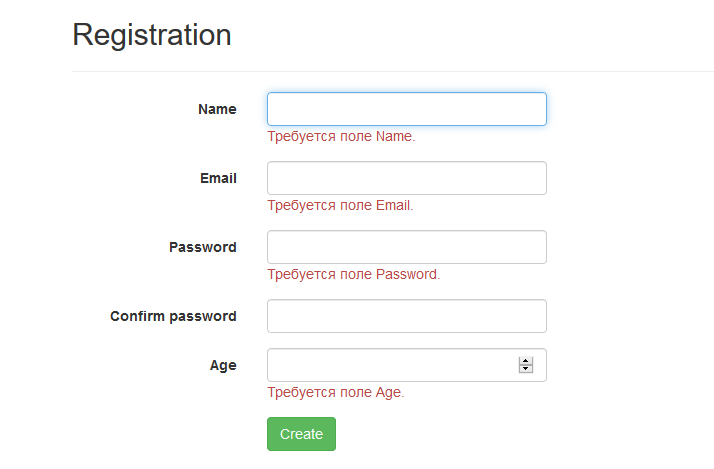


Рисунок 4.2 – Валидация заполняемых полей

Валидация полей оповещает пользователя о неправильности введенных данных. Это позволит подсказать пользователю, что делать.

## Авторизация пользователя

После регистрации нового пользователя, система автоматически авторизует его. Но отдельная страница авторизации позволит пользователю самостоятельно входить в систему в дальнейшем (рисунок 4.3).

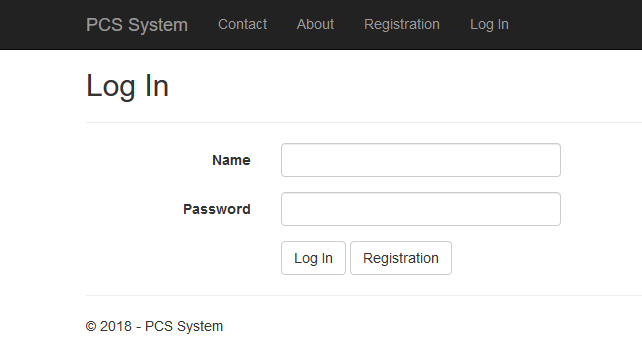


Рисунок 4.3 – Авторизация пользователя в системе

После авторизации пользователю станут доступны некоторые дополнительные возможности, которые отсутствуют без входа в систему.

## Руководство для пользователя

### Работа с заданиями

После входа в систему, пользователь автоматически попадает на главную страницу, которая содержит список созданных заданий для сотрудников. Возможность редактирования и создания новых заданий, для обычного пользователя частично ограничена (рисунок 4.4).

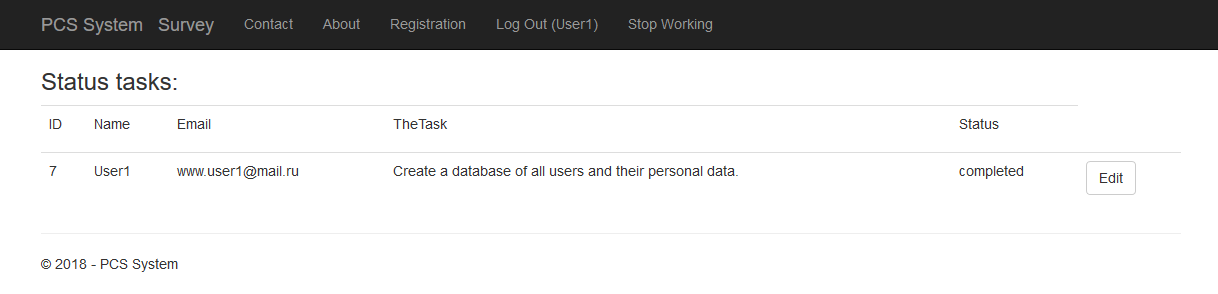


Рисунок 4.4 – Главнач страница с заданиями

После выполнения определенного персонального задания, пользователь должен зайти на страницу редактирования и отметить прогресс выполнения (рисунок 4.5).

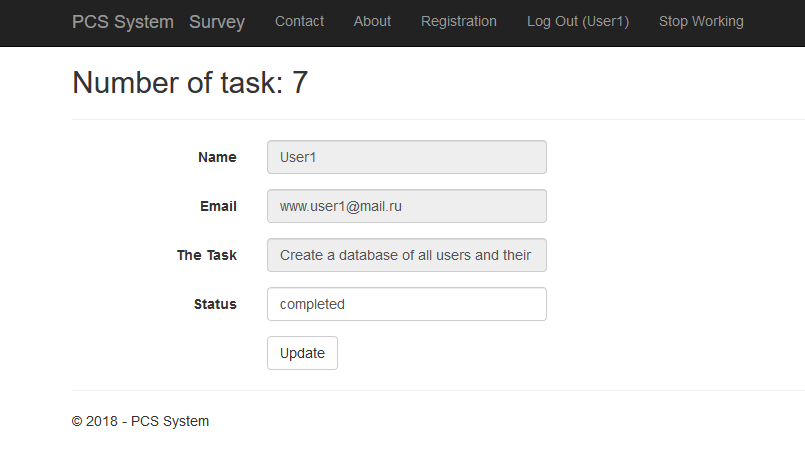


Рисунок 4.5 – Редактирования задания пользователем

Другие поля, кроме статуса, заблокированы, чтобы избежать редактирования информации разными пользователями.

### Работа с опросами

Страница опросов содержит список доступных вопросов, которые создаются администратором, эти опросы собирают общую информацию о сотрудниках (рисунок 4.6).

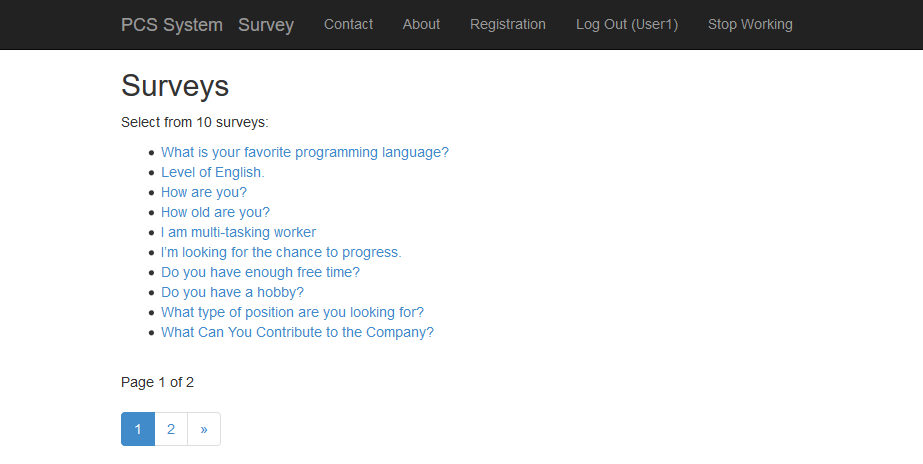


Рисунок 4.6 – Страница опросов пользователя

Создание нового вопроса проходит по определенной схеме, как показано на рисунке 4.7.

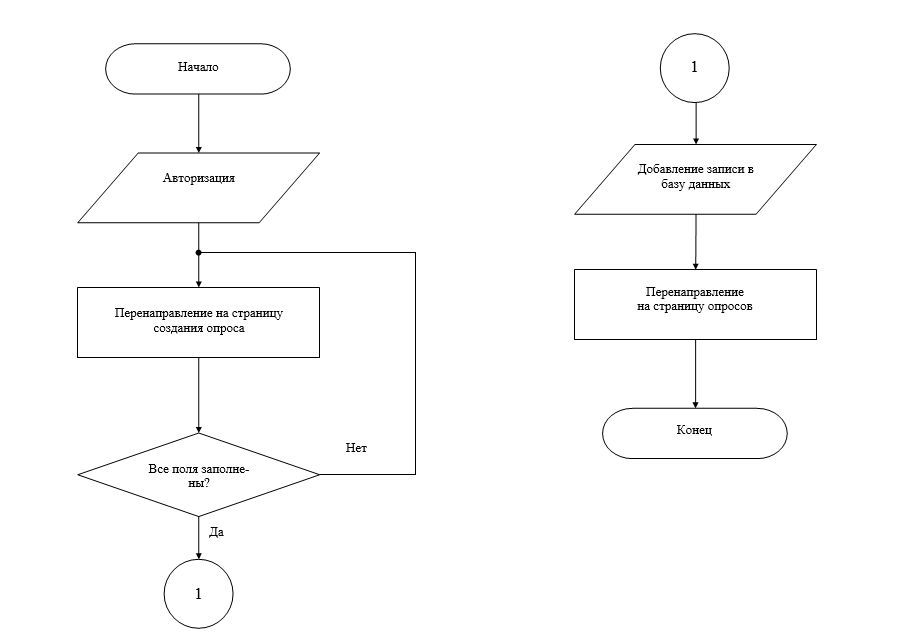


Рисунок 4.7 – Блок-схема создания нового опроса

Зайдя в один из опросов, мы получим либо варианты для возможных ответов, либо информацию о уже выбранном варианте (рисунок 4.8).

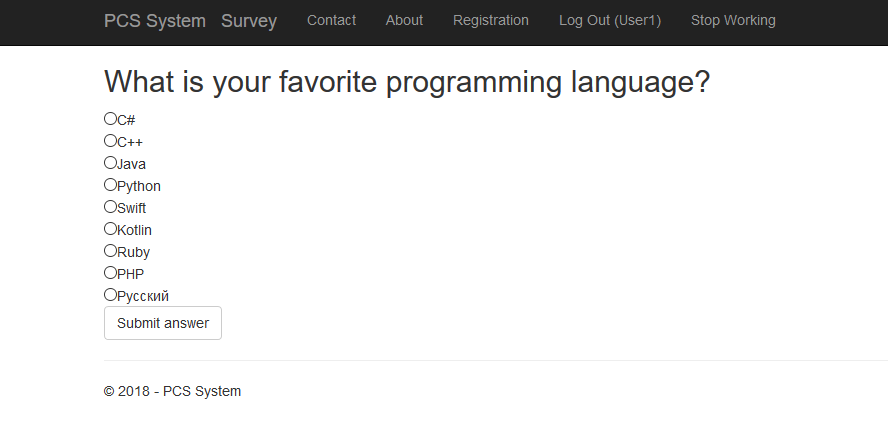


Рисунок 4.8 – Варианты ответа на вопрос

После ответа на вопрос, на этой страница появится информация о выбранном ответе.

### Работа с дополнительной информацией

При нажатии на вкладку Contact пользователя перенаправит на страницу, на которой находится информация о разработчике (рисунок 4.9).

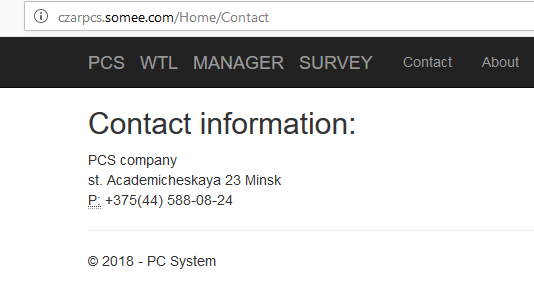


Рисунок 4.9 – Контактная информация

Тем самым пользователь с определенным доступом сможет получить информацию о разработчике и месте разработки.

### Регистрация проработанного времени

Во время авторизации пользователя в системе запускается таймер, который следит за тем, сколько проработал пользователь за рабочий день, когда пришел на рабочее место, во сколько окончил рабочий день. Это позволяет проконтролировать и собрать определенные данные. Таймер останавливается после выхода из системы, либо после нажатия специальной кнопки, которая позволит закончить работу пользователю (рисунок 4.10).

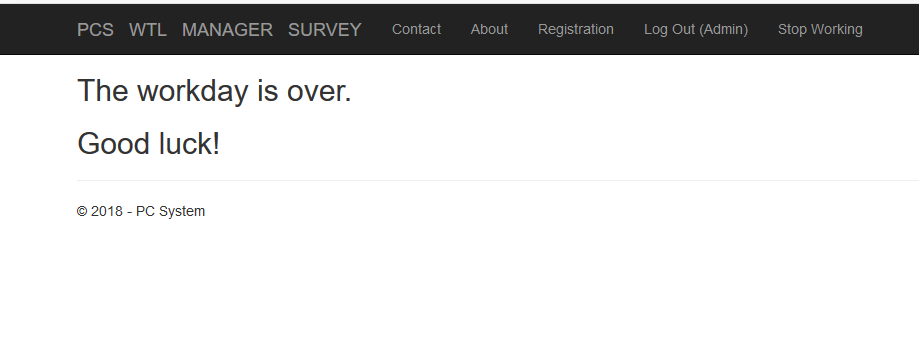


Рисунок 4.10 – Окончание работы пользователя

Все данные о времени сохраняются в таблицу базы данных. Эти данные используются для построения графиков о посещаемости сотрудников.

## Руководство для пользователя с правами администратора

Пользователь с правами администратора имеет дополнительные привилегии, к которым можно отнести:

* управление аккаунтами;
* создание и редактирование заданий для пользователей;
* создание и редактирование опросов;
* просмотр построенных графиков на основе полученной информации;
* просмотр информации о посещаемости.

### Посещаемость сотрудников

После входа в приложение с правами администратора, появляются дополнительные ссылки на другой функционал. Первая ссылка содержит информацию о посещаемости, которая содержит таблицу с данными о пользователе и его посещаемости на сегодняшний день (рисунок 4.11).

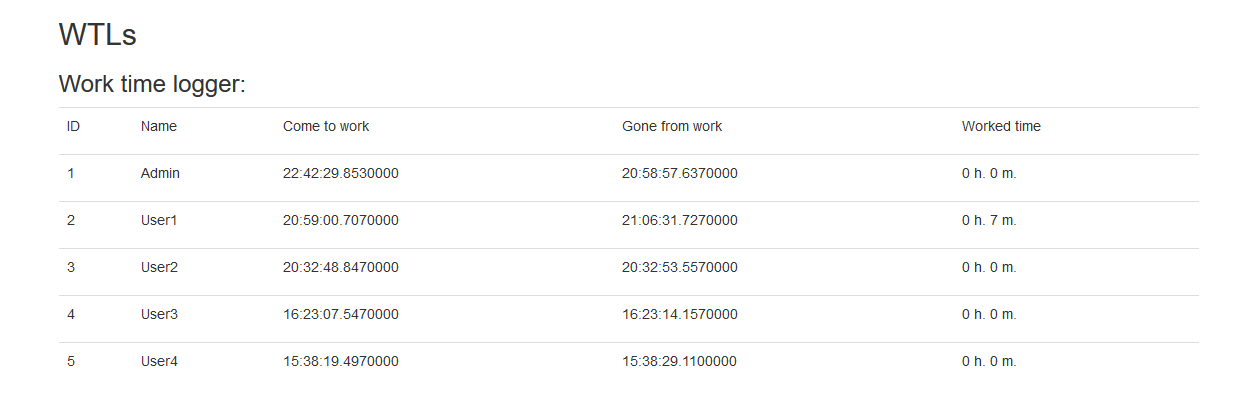


Рисунок 4.11 – Посещаемость сотрудников

Чуть ниже на основе такой таблицы автоматически строится график, который содержит длительность рабочего дня сотрудников (рисунок 4.12).

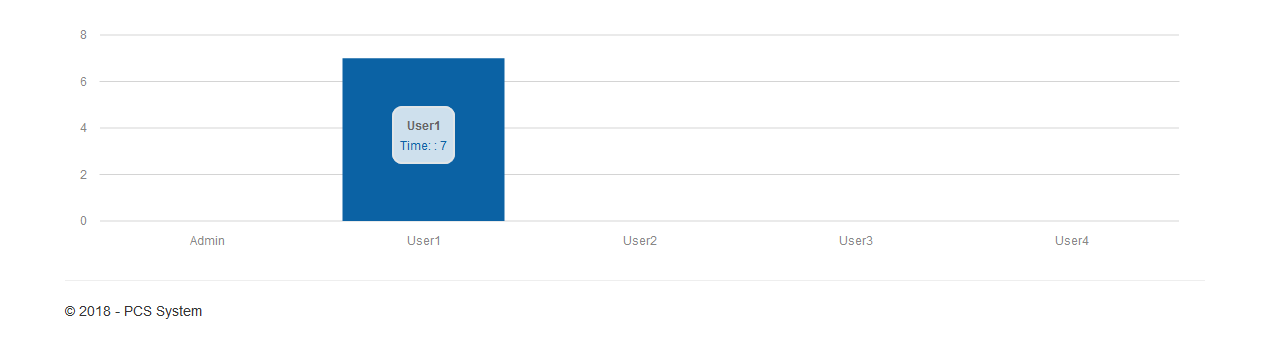


Рисунок 4.12 – График посещаемости сотрудников

Этот график строится на основе полученного времени входа на аккаунт и выхода из него.

### Управление системой опросов

Страница Manager содержит список созданных вопросов для пользователей. Администратор может удалять, создавать и редактировать вопросы и просматривать подробную информацию о том, сколько людей выбрало определенный вариант ответа на вопрос (рисунок 4.13).

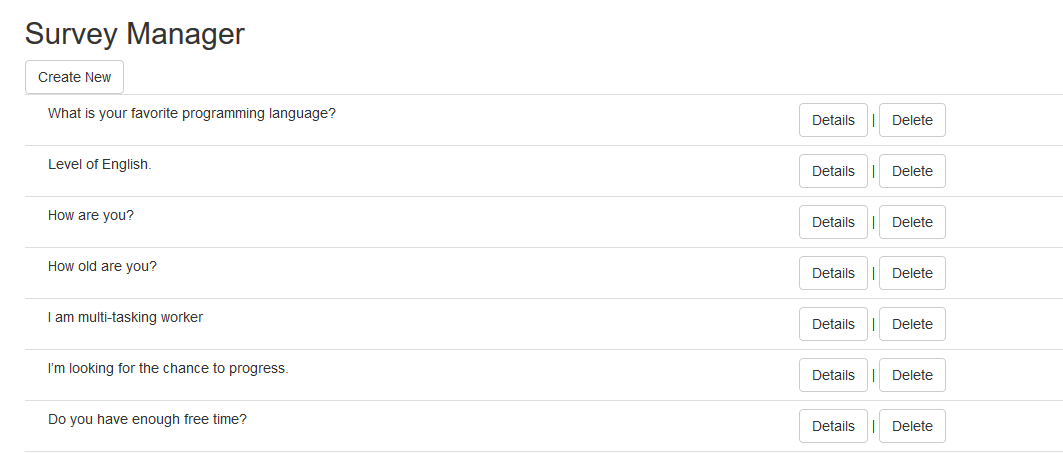


Рисунок 4.13 – Страница с создаваемыми вопросами

При создании нового вопроса, мы получаем страницу, на которой обязательно должны заполнить поле вопроса и как минимум 2 поля с вариантами ответа, иначе пользователь получит предупреждение о незаполненных полях, благодаря чему пользователь сразу поймет, в чем проблема и как ее исправить (рисунок 4.14).

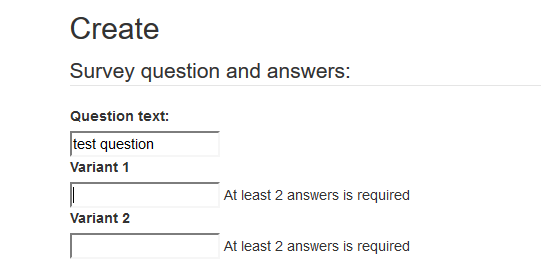


Рисунок 4.14 – Создание нового вопроса

При просмотре подробной информации о вопросе, мы можем увидеть варианты ответов и количество пользователей, ответивших за определенный вариант (рисунок 4.15).

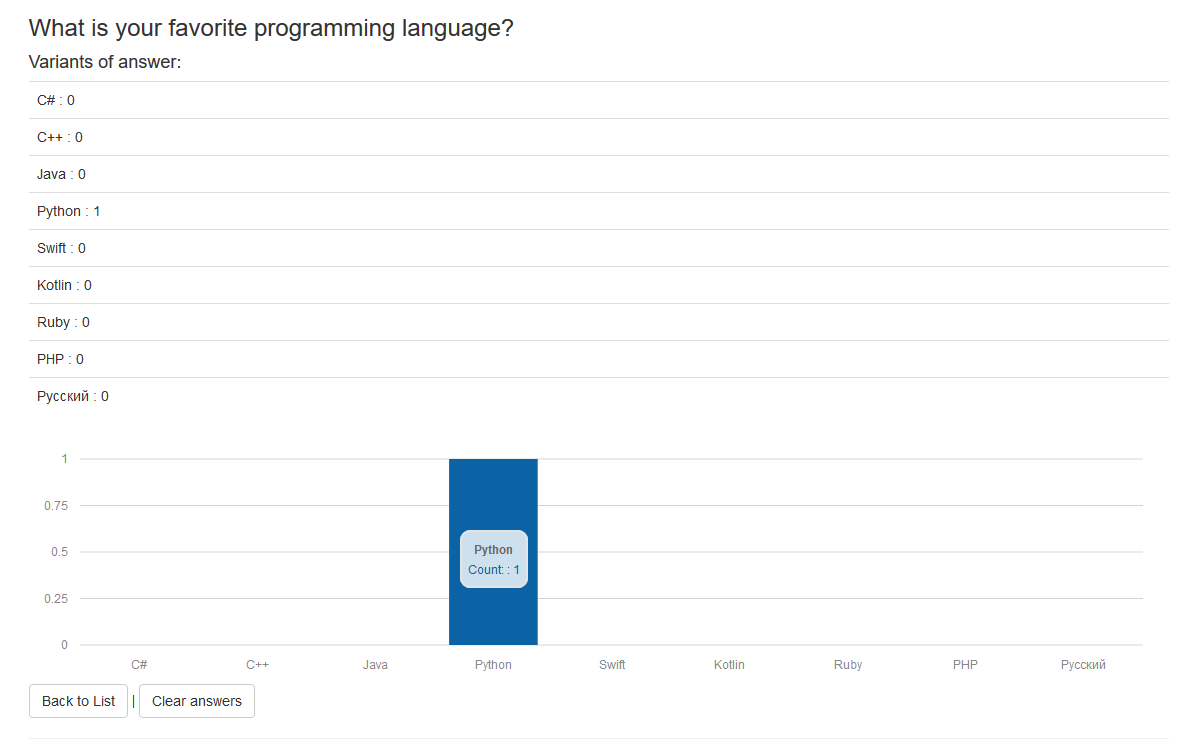


Рисунок 4.15 – Подробная информация созданного вопроса

Все данные постоянно хранятся на каждой странице для всех опросов.

## Тестирование программного средства

Тестирование программного обеспечения – процесс исследования, испытания программного обеспечения (ПО) с целью получения информации о качестве продукта.

Существующие на сегодня методы тестирования ПО не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого ПО.

Такой процесс формальной проверки, или верификации, может доказать, что дефекты отсутствуют с точки зрения используемого метода. То есть нет никакой возможности точно установить или гарантировать отсутствие дефектов в программном продукте с учётом человеческого фактора, присутствующего на всех этапах жизненного цикла ПО.

### Тестирование полей авторизации

В приложении присутствует некоторое количество заполняемых полей, которые в свою очередь необходимо протестировать, чтобы пользователь был всегда в курсе того, что он делает неверно (рисунок 4.16)

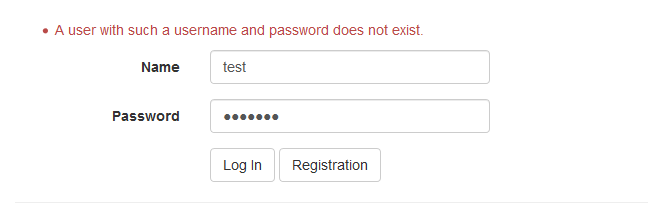


Рисунок 4.16 – Проверка на корректность данных

Если же эти поля попробовать оставить незаполненными, то пользователь получит немного другое сообщение о том, что поля нужно заполнить (рисунок 4.17).

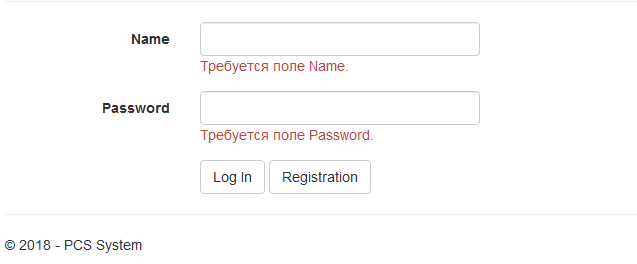


Рисунок 4.17 – Проверка на пустоту полей для ввода

Такие сообщения необходимы для пользователей, чтобы они знали, что и как необходимо заполнить, чтобы авторизация прошла успешно.

### Тестирование полей регистрации

Если в поля регистрации мы попытаемся ввести email адрес, который не соответствует стандартному виду, то пользователь получит сообщение о некорректности почтового адреса (рисунок 4.18).

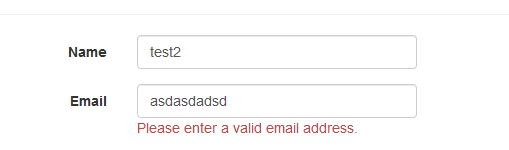


Рисунок 4.18 – Корректность вводимоего email адреса

При вводе пароля в соответствующее поле, мы получим оповещение о том, что пароль не соответствует требованиям: длина, наличие символов, наличие цифры (рисунок 4.19).

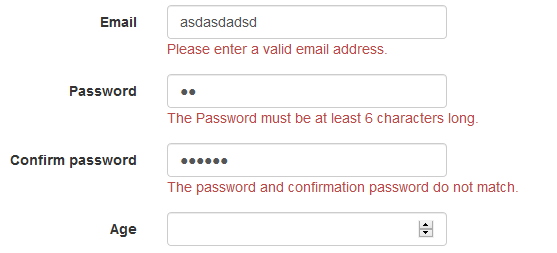


Рисунок 4.19 – Корректность вводимоего email адреса

Отображение такой информации облегчит пользователю исправление своих ошибок, при авторизации или регистрации.

### Создание новых опросов для пользователей

В программном средстве присутствует функционал, который позволяет создавать новые вопросы для опроса пользователей, которые будут собирать информацию о пользователях.

При создании нового опроса у нас появляются поля для заполнения: сам вопрос, варианты ответа. Вариантов ответа необходимо как минимум два. Если пользователь не заполнит эти 3 поля, то получит сообщения о необходимости заполнения (рисунок 4.20).

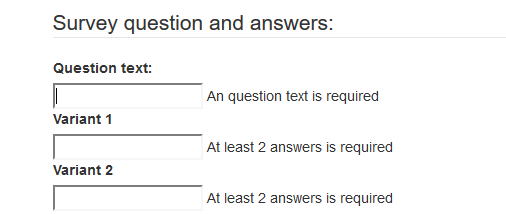


Рисунок 4.20 – Оповещение об обязательном заполнении полей

Как только пользователь заполнит эти три поля, то будет создан новый вопрос, ответы на который может оставить каждый зарегистрированный пользователь.

### Выводы по разделу

В данном разделе было приведено руководство обычного пользователя (сотрудника) и пользователя с правами администратора. Подробно были расписаны возможности все пользователей.

Само веб-приложение было проверено на правильность вводимых значений, на пустоту заполняемых полей. Ошибок не обнаружено.

Полученные результаты ручного и функционального тестирования показали, что приложения работает полностью правильно и без ошибок.

# Экономическое обоснование цены программного средства

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*45*

*ДП 05.00.ПЗ*

*Разраб.*

*Рауба А.А.*

*Провер.*

*.*

*Буснюк Н.Н.*

Консульт.

*Россоха Е.В.*

*Н. Контр.*

*Жигаровская С.А.*

*Утверд.*

*Смелов В.В.*

Экономическое обоснование цены программного средства

*Лит.*

*Листов*

*7*

БГТУ 74419017, 2018

У

## Характеристика программного средства

При выполнении данного дипломного проекта используя технологию MVC 5, была разработана система, представленная веб-приложением. Это приложение состоит из трех основных частей: сайт-визитка, каталог предложений и конструктор страниц для этого каталога.

В рамках данного раздела необходимо определить затраты, произведенные на всех стадиях разработки описанного программного средства. Также необходимо провести расчет экономии основных видов ресурсов в связи с использованием данного программного средства.

## Исходные данные и маркетинговый анализ

Исходные данные для расчета приведены в таблице 5.1.

Таблица 5.1 – Исходные данные для расчета

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателя | Единица измерения | Условные обозначения | Норматив |
| 1 | 2 | 3 | 4 |
| Численность разработчиков | чел. | Чр | 1 |
| Норматив дополнительной заработной платы | % | Ндз | 10 |
| Ставка отчислений в Фонд социальной защиты населения | % | Нфсзн | 34+0,6 |
| Цена одного машино-часа | руб. | Смч | 0,05 |
| Норматив прочих затрат | % | Нпз | 12 |
| Норматив накладных расходов | % | Нобп*,* обх | 103 |
| Норматив расходов на сопровождение и адаптацию | % | Нрса | 10 |
| Уровень рентабельности | % | Урн | 23 |
| Ставка НДС | % | Нндс | 20 |

В ходе маркетингового анализа было установлено, что средняя цена разработки сайта-визитки составляет 1320 рублей; средняя цена разработки сайта-опросника составляет 1200 рублей; в случае с конструктором страниц, прямых заказов найти не удалось, поэтому поиск велся по формулировке «Индивидуальный сайт». В итоге средняя стоимость разработки составила 4750 рублей.

Итого полная стоимость разработки системы составляет 7270 рублей. Эта себестоимость является средней для такого разработки такого типа программного средства.

## Методика обоснования цены

В современных рыночных экономических условиях программное средство (ПС) выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенные и имеющие товарный вид ПС, реализуемые покупателям по рыночным отпускным ценам. Все завершенные разработки ПС являются научно-технической продукцией.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования ПС. Выбор эффективных проектов ПС связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации ПС, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

* снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ;
* сокращения расходов на оплату машинного времени и других ресурсов на отладку программ;
* снижения расходов на материалы;
* ускорение ввода в эксплуатацию новых систем;
* улучшения показателей основной деятельности в результате использования ПС.

Стоимостная оценка ПС у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* расходы на материалы и комплектующие;
* расходы на спецоборудование;
* расходы на оплату машинного времени;
* прочие прямые затраты;
* накладные расходы.

На основании затрат рассчитывается себестоимость и отпускная цена программного средства.

### Объем программного средства

Для оценки объема программного средства, все его функции классифицируются с использованием специального каталога функций, который определяет их объем. Общий объем программного средства *V*o, вычисляется как сумма объемов *Vi* каждой из *n* его функций (формула 5.1).

|  |  |  |
| --- | --- | --- |
|  | . | (5.1) |

В таблице 5.2 представлены функции, присутствующие в рассматриваемом программном средстве и соответствующий им объем в условных машино-командах.

Таблица 5.2 – Содержание и объем функций в программном средстве

|  |  |  |
| --- | --- | --- |
| № функции | Содержание функции | Объем, условных машино-команд |
| 101 | Организация ввода информации | 150 |
| 102 | Контроль, предварительная обработка | 450 |
| 105 | Обработка входного заказа и формирование таблиц | 1340 |
| 207 | Манипулирование данными | 9550 |
| 506 | Обработка ошибочных и сбойных ситуаций | 410 |
| 507 | Обеспечение интерфейса между компонентами | 970 |
| 707 | Графический вывод результатов | 480 |

Опираясь на данные таблицы 5.2, можно определить объем программного средства, разработанного в ходе дипломного проектирования:

*Vo* = 150 + 450 + 1340 + 9550 + 410 + 970 + 480 = 13350 (условных машино-команд).

Уточненный объем программного средства *V*o*/* равен произведению объема программного средства *V*o на коэффициент изменения скорости обработки информации Кск (формула 5.2).

|  |  |  |
| --- | --- | --- |
|  | *V*o*/ = V*o*·*Кск | (5.2) |

Исходя из вычисленного объема программного средства, можно определить уточненный объем программного средства:

*V*o*/* = 13350 ⋅ 0,6 = 8010 (условных машино-команд).

### Основная заработная плата

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере веб-программирования на ASP.Net MVC. В итоге было установлено, что средняя месячная заработная плата на позиции junior составляет 800 рублей.

Проект разрабатывался одним человеком на протяжении двух месяцев. Таким образом, основная заработная плата будет рассчитываться по формуле (5.3):

|  |  |
| --- | --- |
|  | (5.3) |

где Соз– основная заработная плата, руб.;

Траз – время раработки, месяцев;

Краз – количество разработчиков, человек;

Сзп – средняя месячная заработная плата.

|  |  |
| --- | --- |
| руб. |  |

### Дополнительная заработная плата

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле (5.4):

|  |  |
| --- | --- |
|  | (5.4) |

где Соз– основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

руб.

### Отчисления в фонд социальной защиты населения

Отчисления в Фонд социальной защиты населения (ФСЗН) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле (5.5):

|  |  |
| --- | --- |
|  | (5.5) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата на конкретное ПС, млн. руб.;

Нфсзн– норматив отчислений в Фонд социальной защиты населения, %.

руб.

### Расходы на материалы

Сумма расходов на материалы См определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода Нм на уточненный объем программного средства *V*o*/* (формула 5.6).

|  |  |  |
| --- | --- | --- |
|  | . | (5.6) |

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равен 0,460 руб (по данным, приведенным в приложении 2 таблице П 2.10 «Оценка значений среднего расхода материалов на разработку и отладку 100 строк кода применения ПС» методического пособия), можно определить сумму расходов на материалы:

См = 0,460 ⋅ 8010 / 100 = 36,84 руб.

### Расходы на оплату машинного времени

Сумма расходов на оплату машинного времени Смв определяется как произведение стоимости одного машино-часа Смч на уточненный объем программного средства *V*o*/* и на норматив расхода машинного времени на отладку ста строк исходного кода Нмв (формула 5.7).

|  |  |  |
| --- | --- | --- |
|  | . | (5.7) |

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен 15 (по данным, приведенным в приложении 2 таблице П 2.11 «Оценка значений среднего машинного времени на отладку 100 строк исходного кода без применения ПС» методического пособия), можно определить сумму расходов на оплату машинного времени:

Смв =0,05 ⋅ 8010 ⋅ 15 / 100 = 60,08 руб.

### Прочие прямые затраты

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз (формула 5.8).

|  |  |  |
| --- | --- | --- |
|  | . | (5.8) |

Спз = 1600 ⋅ 12 / 100 = 192 (руб.).

### Накладные расходы

Сумма накладныхрасходовСобп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх (формула 5.9).

|  |  |  |
| --- | --- | --- |
|  | . | (5.9) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов:

Собп,обх = ⋅ 103 / 100 = руб.

### Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов (формула 5.10).

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзно + См + Смв + Спз + Собп,обх | (5.10) |

Ср = + + + 36,84 + 60,08 + 192 + = 4305,88 руб.

### Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства Срса определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию Нрса (формула 5.11).

|  |  |  |
| --- | --- | --- |
|  | . | (5.11) |

Срса = 4305,88 ⋅ 10 / 100 = 430,59 (руб.).

### Полная себестоимость

Полная себестоимость Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию программного средства Срса (формула 5.12).

|  |  |  |
| --- | --- | --- |
|  | . | (5.12) |

Сп = 4305,88 + 430,59 = 4736,47 (руб.).

### Определение цены, оценка эффективности

При рассмотрении аналогов было установлено, что данные продукты являются коммерческими и найти стоимость аналогичных продуктов найти сложно. В своем представлении данный продукт представляет веб-приложение, поэтому в качестве среднее рыночной цены можно взять среднюю цену разработки веб-порталов, которая равна 7270 руб.

Прибыль рассчитывается по формуле (6.13):

|  |  |  |
| --- | --- | --- |
| , | (6.13) | (6.13) |

где Ппс – прибыль от реализации программного средства, руб.;

Цр – средняя рыночная цена продукта, руб.;

Сп – полная себестоимость программного средства, руб.;

Ппс = (7270 / 1,2) - 4736,47 = 1321,86 (руб.).

Уровень рентабельности разработанного программного средства определяется по формуле (6.14):

|  |  |
| --- | --- |
|  | (6.14) |

где Урент – уровень рентабельности программного средства, %

Сп – полная себестоимость программного средства, руб.;

Ппс – прибыль от реализации программного средства, руб.

Ур = 1321,86 / 4736,47 = 27,9 (%.).

## Вывод по разделу

В данном разделе были произведены расчеты различных экономических показателей. В таблице 5.3 представлены результаты расчетов для основных показателей данной главы в краткой форме. Данные показатели очень важны при анализе цены программы.

Каждый из этих показателей несет определенную роль в экономической составляющей при реализации программного средства.

Таблица 5.3 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, мес. | 2 |
| Количество программистов, чел. | 1 |
| Зарплата с отчислениями, руб. | 2208,96 |
| Расходы на материалы, оплату машинного времени, прочие, руб | 288,92 |
| Накладные расходы, руб | 1648 |
| Себестоимость разработки программного средства, руб. | 4305,88 |
| Расходы на сопровождение и адаптацию, руб. | 430,59 |
| Полная себестоимость, руб. | 4736,47 |
| Цена разработки с НДС, руб. | 7270 |
| Прибыль от реализации, руб. | 1321,86 |
| Рентабельность разработки, % | 27,9% |

Так как рентабельность программного средства является положительной величиной, проект можно считать экономически целесообразным.

ЗАКЛЮЧЕНИЕ

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*52*

*ДП 00.00.ПЗ*

*Разраб.*

*Рауба А.А.*

*Провер.*

*.*

*Буснюк Н.Н.*

Консульт.

*Буснюк Н.Н.*

*Н. Контр.*

*Жигаровская С.А.*

*Утверд.*

*Смелов В.В.*

ЗАКЛЮЧЕНИЕ

*Лит.*

*Листов*

*1*

БГТУ 74419017, 2018

У

В ходе выполнения дипломного проекта были рассмотрены основные аналоги программного средства, их преимущества и недостатки, были просмотрены и изучены их возможности. Произведен анализ и обзор теоретического материала по данной тематике дипломного проекта.

Также были проанализированы и изучены основные технологии, которые применялись в разработке данного программного средства для дипломного проекта. Само приложение было реализовано на языке C# в среде Visual Studio. В основе структуры приложения лежит паттерн MVC5. В качестве системы управления базой данных был выбран и использован Microsoft SQL Server 2014.

Данные приложения хранятся в базе данных, которая была спроектирована до разработки программного средства. В ходе подготовки к разработке ПО была разработана структура базы данных и приложения.

При написании руководства для всех пользователей системы были учтены все возможные действия пользователя и подробно описаны все возможные действия.

В рамках дипломного проекта было разработано программное средство «Веб-приложение для учета и анализа показателей работы персонала предприятия», которое соответствует предъявленным требованиям и поставленным задачам.

Данный дипломный проект успешно внедрен в ОАО «Слуцкий сахарорафинадный комбинат».

Работа промежуточного варианта веб-приложения системы учета показателей работы сотрудников демонстрировалась на 68 научно-технической конференции учащихся в Белорусском государственном технологическом университете 11 апреля 2018 года.

Список используемых источников

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

*1*

*53*

*ДП 00.00.ПЗ*

*Разраб.*

*Рауба А.А.*

*Провер.*

*.*

*Буснюк Н.Н.*

Консульт.

*Буснюк Н.Н.*

*Н. Контр.*

*Жигаровская С.А.*

*Утверд.*

*Смелов В.В.*

Список использованных источников

*Лит.*

*Листов*

*1*

БГТУ 74419017, 2018

У

1. Информационный сайт: контроль персонала [Электронный ресурс] / worldsellers.ru – Режим доступа: <http://worldsellers.ru/kontrol-personala/> – Дата доступа: 17.04.2018.
2. Сайт компании: контроль персонала [Электронный ресурс] / gd.ru – Режим доступа: <https://www.gd.ru/articles/3718-kontrol-personala> – Дата доступа: 17.04.2018.
3. Информационный сайт: ASP .NET [Электронный ресурс] / web-creator.ru – Режим доступа: <https://web-creator.ru/articles/dot_net_and_asp> – Дата доступа: 17.04.2018
4. Сайт о программировании Metanit: Введение в MS SQL Server [Электронный ресурс] / metanit.com – Режим доступа: <https://metanit.com/sql/sqlserver/1.1.php> – Дата доступа: 18.04.2018.
5. Сайт о программировании ProfessorWeb: Введение в C# [Электронный ресурс] / professorweb.ru – Режим доступа: <https://professorweb.ru/my/csharp/charp_theory/level1/1_1.php> – Дата доступа: 18.04.2018.
6. Сайт о программировании ProgoPedia: Язык программирования SQL [Электронный ресурс] / progopedia.ru – Режим доступа: <http://progopedia.ru/language/sql/> – Дата доступа: 18.04.2018.
7. Сайт о программировании Shkolo: Язык разметки HTML [Электронный ресурс] / shkolo.ru – Режим доступа: <http://shkolo.ru/html-yazyik-razmetki-giperteksta/> – Дата доступа: 19.04.2018.
8. Сайт о программировании ProfessorWeb: Библиотека jQuery [Электронный ресурс] / professorweb.ru – Режим доступа: [https://professorweb.ru/my/javascript/jquery](https://professorweb.ru/my/javascript/jquery/level1/jquery_index.php) – Дата доступа: 19.04.2018.
9. Сайт для публикаций: Morris.js [Электронный ресурс] / habrahabr.ru – Режим доступа: <https://habrahabr.ru/post/139326/> – Дата доступа: 19.04.2018.
10. Сайт о программировании Metanit: Особенности паттерна MVC 5 [Электронный ресурс] / metanit.com – Режим доступа: https://metanit.com/sharp/mvc5/1.1.php – Дата доступа: 19.04.2018.
11. Адам Фримен, Стивен Сандерсон, ASP.NET MVC 4 Framework с примерами на C# для профессионалов / Адам Фримен, Стивен Сандерсон. – 4-е издание, 2014. – 666 с.
12. Каштелян, Т. В. Экономические обоснования дипломных проектов. Методические указания для студентов специальности 1-40 01 02 «Информационные системы и технологии» / Т. В. Каштелян. – Мн.: БГТУ, 2013. – 30 с.

# ПРИЛОЖЕНИЕ А

Листинг 1. Код контроллера AccountController.

using Personnel\_Control\_System.Models;

using System;

using System.Linq;

using System.Net;

using System.Web.Mvc;

using System.Web.Security;

namespace Personnel\_Control\_System.Controllers

{

public class AccountController : Controller

{

public ActionResult Login()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Login(LoginModel model)

{

try {

if (ModelState.IsValid)

{

// поиск пользователя в бд

User user = null;

using (UserContext db = new UserContext())

{

user = db.User.FirstOrDefault(u => u.Name == model.Name && u.Password == model.Password);

var Come = db.WTLs.SingleOrDefault(u => u.Name == model.Name);

if(Come != null)

{

Come.Come = DateTime.Now;

db.SaveChanges();

}

db.SaveChanges();

}

if (user != null)

{

FormsAuthentication.SetAuthCookie(model.Name, true);

return RedirectToAction("Index", "Home");

}

else

{

ModelState.AddModelError("", "A user with such a username and password does not exist.");

}

}

}

catch

{

//return error status code

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

return View(model);

}

public ActionResult Register()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Register(RegisterModel model)

{

if (ModelState.IsValid)

{

User user = null;

using (UserContext db = new UserContext())

{

user = db.User.FirstOrDefault(u => u.Name == model.Name);

}

if (user == null)

{

// создаем нового пользователя

using (UserContext db = new UserContext())

{

db.User.Add(new User

{

Name = model.Name,

Email = model.Email,

Password = model.Password,

Age = model.Age,

});

db.WTLs.Add(new WTLs

{

Name = model.Name,

Email = model.Email,

Come = DateTime.Now,

Gone = DateTime.Now,

TSpanH = 0,

TSpanM = 0,

});

db.SaveChanges();

user = db.User.Where(u => u.Name == model.Name && u.Password == model.Password).FirstOrDefault();

}

// если пользователь удачно добавлен в бд

if (user != null)

{

FormsAuthentication.SetAuthCookie(model.Name, true);

return RedirectToAction("Index", "Home");

}

}

else

{

ModelState.AddModelError("", "User with such login already exists.");

}

}

return View(model);

}

public ActionResult StopWork(WTLs model)

{

if (ModelState.IsValid)

{

using (UserContext db = new UserContext())

{

var Gone = db.WTLs.SingleOrDefault(u => u.Name == User.Identity.Name);

DateTime ComeTime;

DateTime GoneTime;

if (Gone != null)

{

Gone.Gone = DateTime.Now;

ComeTime = Gone.Come;

GoneTime = Gone.Gone;

Gone.TSpanH = ((TimeSpan)(GoneTime - ComeTime)).Hours;

Gone.TSpanM = ((TimeSpan)(GoneTime - ComeTime)).Minutes % 60;

db.SaveChanges();

}

db.SaveChanges();

}

}

FormsAuthentication.SignOut();

return View(model);

}

public ActionResult Logoff(LoginModel model)

{

using (UserContext db = new UserContext())

{

var Gone = db.WTLs.SingleOrDefault(u => u.Name == User.Identity.Name);

DateTime ComeTime;

DateTime GoneTime;

if (Gone != null)

{

Gone.Gone = DateTime.Now;

ComeTime = Gone.Come;

GoneTime = Gone.Gone;

Gone.TSpanH = ((TimeSpan)(GoneTime - ComeTime)).Hours;

Gone.TSpanM = ((TimeSpan)(GoneTime - ComeTime)).Minutes % 60;

db.SaveChanges();

}

db.SaveChanges();

}

FormsAuthentication.SignOut();

return RedirectToAction("Index", "Home");

}

}

}

# ПРИЛОЖЕНИЕ Б

Листинг 2. Код контроллера HomeController.

using Personnel\_Control\_System.Models;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Web.Mvc;

namespace Personnel\_Control\_System.Controllers

{

public class HomeController : Controller

{

UserContext Taskdb = new UserContext();

[Authorize]

public ActionResult Index()

{

return View(Taskdb.Task);

}

[Authorize]

[HttpGet]

public ActionResult Add()

{

return View();

}

[Authorize]

[HttpPost]

public ActionResult Add(Task tsk)

{

try

{

Taskdb.Task.Add(tsk);

Taskdb.SaveChanges();

}

catch (RetryLimitExceededException /\* dex \*/)

{

//Log the error (uncomment dex variable name and add a line here to write a log.)

ModelState.AddModelError("", "Unable to save changes. Try again, and if the problem persists, see your system administrator.");

}

return RedirectToAction("Index");

}

[Authorize]

[HttpGet]

public ActionResult EditTask(int? id)

{

if (id == null)

{

return HttpNotFound();

}

Task tsk = Taskdb.Task.Find(id);

if (tsk != null)

{

return View(tsk);

}

return HttpNotFound();

}

[Authorize]

[HttpPost]

public ActionResult EditTask(Task tsk)

{

Taskdb.Entry(tsk).State = EntityState.Modified;

Taskdb.SaveChanges();

return RedirectToAction("Index");

}

[Authorize]

public ActionResult Delete(int id)

{

Task b = Taskdb.Task.Find(id);

if (b == null)

{

return HttpNotFound();

}

return View(b);

}

[Authorize]

[HttpPost, ActionName("Delete")]

public ActionResult DeleteConfirmed(int id)

{

Task b = Taskdb.Task.Find(id);

if (b == null)

{

return HttpNotFound();

}

Taskdb.Task.Remove(b);

Taskdb.SaveChanges();

return RedirectToAction("Index");

}

[Authorize]

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View();

}

[Authorize]

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

}

}

# ПРИЛОЖЕНИЕ В

Листинг 3. Код контроллера SurveyController.

using Personnel\_Control\_System.Models;

using Personnel\_Control\_System.ViewModels;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Web.Mvc;

using PagedList;

namespace Personnel\_Control\_System.Controllers

{

public class SurveyController : Controller

{

SurveyContext surveyDb = new SurveyContext();

//List of surveys, divided into pages

// GET: Survey

public ActionResult Index(int? page)

{

int pageSize = 10;

int pageNumber = (page ?? 1);

List<Question> questions = surveyDb.Questions.ToList();

return View(questions.ToPagedList(pageNumber, pageSize));

}

public ActionResult Details(int? id)

{

//if request is incorrect

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Question question = surveyDb.Questions.Find(id);

if (question == null)

{

return HttpNotFound();

}

//searching for answers - if user already passed this survey

List<AnswerVariant> answervariants;

Answer userAnswer = surveyDb.Answers.SingleOrDefault(a => a.QuestionId == id && a.Username == User.Identity.Name);

//if not - sending to view list of answers

if (userAnswer == null)

{

answervariants = surveyDb.AnswerVariants.Where

(a => a.QuestionId == id).ToList();

}

//if user passed this survey already - sending choosen answer to view

else

{

answervariants = surveyDb.AnswerVariants.Where(a => a.AnswerVariantId == userAnswer.AnswerVariantId).ToList();

}

var survey = new Survey

{

SurveyQuestion = surveyDb.Questions.Single(q => q.QuestionId == id),

SurveyAnswers = answervariants

};

return View(survey);

}

[HttpPost]

public ActionResult Details(Survey answeredSurvey)

{

int questId = answeredSurvey.QuestId;

//just in case if user hasn't choose answer

try

{

int id = answeredSurvey.ChoosenAnswer;

Answer choosenAnswer = new Answer

{

AnswerVariantId = id,

QuestionId = questId,

Username = User.Identity.Name

};

surveyDb.Answers.Add(choosenAnswer);

surveyDb.SaveChanges();

}

catch

{

//return error status code

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

//else - return to survey page

//should view just choosen answer

return RedirectToAction("Details", new { id = questId });

}

}

}

# ПРИЛОЖЕНИЕ Г

Листинг 4. Код контроллера SurveyManagerController.

using Personnel\_Control\_System.Models;

using Personnel\_Control\_System.ViewModels;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Web.Mvc;

namespace Personnel\_Control\_System.Controllers

{

public class SurveyManagerController : Controller

{

private SurveyContext db = new SurveyContext();

// GET: SurveyManager

public ActionResult Index()

{

return View(db.Questions.ToList());

}

// GET: SurveyManager/Details/5

public ActionResult Details(int? id)

{

//if request is incorrect

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Question question = db.Questions.Find(id);

if (question == null)

{

return HttpNotFound();

}

List<AnswerVariant> answervariants = db.AnswerVariants.Where

(a => a.QuestionId == id).ToList();

List<Answer> answers = db.Answers.Where(a => a.QuestionId == id).ToList();

var survey = new Survey

{

SurveyQuestion = db.Questions.Single(q => q.QuestionId == id),

SurveyAnswers = answervariants,

Answers = answers

};

return View(survey);

}

//clear history of survey

public ActionResult ClearAnswers(int? id)

{

//if request is incorrect

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Question question = db.Questions.Find(id);

if (question == null)

{

return HttpNotFound();

}

List<Answer> deletedAnswers = db.Answers.Where(a => a.QuestionId == id).ToList();

foreach (Answer answ in deletedAnswers)

{

db.Answers.Remove(answ);

db.SaveChanges();

}

return RedirectToAction("Index");

}

// GET: SurveyManager/Create

public ActionResult Create()

{

return View();

}

[HttpPost]

public ActionResult Create(FormCollection values)

{

var survey = new SurveyCreate();

TryUpdateModel(survey);

try

{

Question newQuestion = new Question();

newQuestion.Text = survey.QuestionText;

db.Questions.Add(newQuestion);

db.SaveChanges();

int newQestId = db.Questions.Single(q => q.Text == survey.QuestionText).QuestionId;

AnswerVariant newAW = new AnswerVariant { QuestionId = newQestId, Text = survey.AnsverVariant1 };

db.AnswerVariants.Add(newAW); db.SaveChanges();

newAW.Text = survey.AnsverVariant2;

db.AnswerVariants.Add(newAW); db.SaveChanges();

if (!String.IsNullOrEmpty(survey.AnsverVariant3)

&& !String.IsNullOrWhiteSpace(survey.AnsverVariant3))

{

newAW.Text = survey.AnsverVariant3;

db.AnswerVariants.Add(newAW); db.SaveChanges();

}

if (!String.IsNullOrEmpty(survey.AnsverVariant4)

&& !String.IsNullOrWhiteSpace(survey.AnsverVariant4))

{

newAW.Text = survey.AnsverVariant4;

db.AnswerVariants.Add(newAW); db.SaveChanges();

}

if (!String.IsNullOrEmpty(survey.AnsverVariant5)

&& !String.IsNullOrWhiteSpace(survey.AnsverVariant5))

{

newAW.Text = survey.AnsverVariant5;

db.AnswerVariants.Add(newAW); db.SaveChanges();

}

if (!String.IsNullOrEmpty(survey.AnsverVariant6)

&& !String.IsNullOrWhiteSpace(survey.AnsverVariant6))

{

newAW.Text = survey.AnsverVariant6;

db.AnswerVariants.Add(newAW); db.SaveChanges();

}

if (!String.IsNullOrEmpty(survey.AnsverVariant7)

&& !String.IsNullOrWhiteSpace(survey.AnsverVariant7))

{

newAW.Text = survey.AnsverVariant7;

db.AnswerVariants.Add(newAW); db.SaveChanges();

}

if (!String.IsNullOrEmpty(survey.AnsverVariant8)

&& !String.IsNullOrWhiteSpace(survey.AnsverVariant8))

{

newAW.Text = survey.AnsverVariant8;

db.AnswerVariants.Add(newAW); db.SaveChanges();

}

if (!String.IsNullOrEmpty(survey.AnsverVariant9)

&& !String.IsNullOrWhiteSpace(survey.AnsverVariant9))

{

newAW.Text = survey.AnsverVariant9;

db.AnswerVariants.Add(newAW); db.SaveChanges();

}

return RedirectToAction("Index");

}

catch

{

return View(survey);

}

}

// GET: SurveyManager/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Question question = db.Questions.Find(id);

if (question == null)

{

return HttpNotFound();

}

return View(question);

}

// POST: SurveyManager/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Question question = db.Questions.Find(id);

db.Questions.Remove(question);

db.SaveChanges();

//deleting answer variants for deleted question

List<AnswerVariant> answervariants = db.AnswerVariants.Where

(a => a.QuestionId == id).ToList();

foreach (AnswerVariant deleteAW in answervariants)

{

db.AnswerVariants.Remove(deleteAW);

db.SaveChanges();

}

//deleting answers

ClearAnswers(id);

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

# ПРИЛОЖЕНИЕ Д

Блок-схема авторизации пользователя

# ПРИЛОЖЕНИЕ Е

Блок-схема регистрации пользователя

# ПРИЛОЖЕНИЕ Ж

Схема создания опроса

# ПРИЛОЖЕНИЕ З

Схема работы приложения

# ПРИЛОЖЕНИЕ И

Экранная копия главной страницы

# ПРИЛОЖЕНИЕ К

Таблица экономических показателей