|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Формат | Обозначение | Наименование | Количество листов | Примечание |
|  |  |  |  |  |
| А4 | - | Задание по дипломному проектированию | 1 |  |
|  |  |  |  |  |
| А4 | МВН.50990.062.ПЗ | Пояснительная записка | 53 |  |
|  |  |  |  |  |
| А4 | МВН.50990.062 | Приложения | 12 |  |
|  |  |  |  |  |
| А4 | МВН.50990.062.01 | Экономическая часть | 1 |  |
|  |  |  |  |  |
| А4 | МВН.50990.062.02 | Диаграмма вариантов использования | 1 |  |
|  |  |  |  |  |
| А4 | МВН.50990.062.03 | Диаграмма развертывания | 1 |  |
|  |  |  |  |  |
| А4 | МВН.50990.062.04 | DFD-диаграмма | 1 |  |
|  |  |  |  |  |
| А4 | МВН.50990.062.05 | Вид приложения | 1 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Изм.

Лист

N докум.

Мороз В.Н.

Магеров В.В.

Бровко В.В

Дровосекова Т.Н

Голубева О.В

Подпись

Дата

Разраб.

Пров.

Реценз.

Н. контр.

Утв.

ДП.1210417.ДО-2017

З

Ведомость объема

дипломного проекта

Лит.

Лист

4

Листов

1

### 1-40 01 01

### Полоцкий государственный университет, группа 12-ИТ-3

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 6

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧИ 8

1.1 Описание предметной области 8

1.2 Постановка задачи 9

1.3 Обзор существующих аналогов 10

2 РАЗРАБОТКА АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧ 14

2.1 Функциональная структура 14

2.2 Алгоритмическое представление решаемых задач 16

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 17

3.1 Выбор и обоснование инструментальных средств 17

3.2 Структура объекта разработки 20

3.3 Проектирование 22

4 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ 27

4.1 Результаты реализации 27

4.2 Результаты тестирования 32

5 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 40

5.1 Обоснование необходимости выведения программного

обеспечения на рынок 40

5.2 Структура работ по созданию программного обеспечения 41

5.3 Составление сметы затрат на разработку программного

обеспечения 42

5.4 Расчет отпускной цены программного обеспечения в случае

реализации на рынок 49

5.5 Сравнительный технико-экономический анализ эффективности

производства нового продукта 50

ЗАКЛЮЧЕНИЕ 51

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 52

ПРИЛОЖЕНИЯ 53

Изм.

Лист

N докум.

Мороз В.Н.

Магеров В.В.

Бровко Н.В.

Дровосекова Т.Н

а

Голубева О.В

Подпись

Дата

Разраб.

Пров.

Реценз.

Н. контр.

Утв.

МВН.50990.040.ПЗ

Мобильное приложение для ГАИ (Пояснительная записка)

Лит.

Лист

5

Листов

53

### 1-40 01 01

### Полоцкий государственный университет, группа 12-ИТ-2

**ВВЕДЕНИЕ**

Ни один рабочий процесс не будет налаженным и продуктивным, если не уделить достаточно внимания организации рабочего времени сотрудников. Тем более, что такая организация необходима как работодателю, так и самим работникам, которые при грамотной системе учёта своего рабочего времени в целом становятся более собранными и дисциплинированными.

Многие организации сегодня готовы предложить потенциальным сотрудникам гибкий рабочий график (форма организации рабочего времени, при которой в определённых пределах работник может самостоятельно определять часы работы в смену; устанавливается период полной отработки установленного суммарного количества рабочих часов (рабочего дня, недели, месяца)). Такой подход в большинстве случаев делает предложение более привлекательным, в отличие от фиксированного рабочего графика (в фиксированное время работник должен находиться на рабочем месте). Однако организовывать рабочий процесс на основе гибкого графика – сложная задача, так как к каждому из сотрудников нужен индивидуальный подход с учетом его индивидуального времени работы. Таким образом, при возрастании количества сотрудников возрастает и сложность управления рабочим процессом организации, и, соответственно, крупным организациям со значительным количеством работников намного проще использовать подход с фиксированным рабочим графиком. Гибкий рабочий график, при этом, имеет следующие преимущества:

* учитываются индивидуальные особенности сотрудника (благоприятное время суток для работы и т.п.);
* возможность совмещение графика работы сразу в нескольких организациях;
* возможность совмещения графика работы с учебой и другими видами деятельности;
* подсознательное влияние на расположение к вакансии путем повышения уровня индивидуального подхода к сотруднику.

Из основных недостатков следует отметить повышенную сложность ведения отчетности в связи с необходимостью индивидуального подхода к каждому сотруднику. Для учета рабочего времени требуется работать с большим количеством документов, что влечет за собой огромные затраты. Данный процесс является трудоёмким, требующим больших затрат не только материальных, но и человеческих ресурсов. Кроме того, при работе с большим количеством документов, учитывая человеческий фактор, легко допустить ошибку, что может привести к нежелательным последствиям.

Как возможность совершенствования процесса учета рабочего времени можно рассматривать его автоматизацию.

Одним из основных методов автоматизации процессов в рамках конкретной предметной области является использование автоматизированных информационных систем (АИС), которые позволяют:

* + сокращать трудозатраты при выполнении традиционных информационных процессов и операций;
  + устранять рутинные операции;
  + модернизировать, а иногда и полностью заменять элементы традиционных технологий;
  + значительно ускорять процессы обработки и преобразования информации;
  + повышать оперативность и качественный уровень обслуживания пользователей и т. п.

В целом АИС можно рассматривать как человеко-машинную систему с автоматизированной технологией получения результатной информации, необходимой для информационного обеспечения персонала и оптимизации процесса управления в предметной деятельности.

Целью дипломного проектирования является автоматизация процесса планирования и учета рабочего времени. Для этого необходимо создать автоматизированную информационную систему, которая позволит сотруднику планировать, а руководству организаций вести учет рабочего времени, что позволит значительно сократить затраты временных и человеческих ресурсов.

**1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧИ**

**1.1 Описание предметной области**

Государственная автоинспекция Республики Беларусь – это подразделение в структуре МВД РБ, которое осуществляет контрольные, надзорные и разрешительные функции в области обеспечения безопасности дорожного движения. Рассмотрим некоторые функции, возлагаемые на Госавтоинспекцию:

* ведение учета транспортных средств;
* выдача регистрационных знаков;
* проведение периодических технических осмотров;
* ведение учета дорожно-транспортных пришествий и их анализ;
* информирования населения об обстановке в сфере дорожного движения;
* выявление нарушения правил дорожного движения и их пресечение;

Перечисленные функции ежедневно выполняются сотрудниками Госавтоинспекции, которых можно разделить на две категории:

* сотрудники несущий дорожно-патрульную службу;
* сотрудник-оператор ведущий прием гражданского населения в отделении;

**1.2 Постановка задачи**

Для решения проблемы планирования и учета рабочего времени в рамках гибкого рабочего графика необходимо разработать систему, которая позволит работникам устанавливать и планировать своё рабочее время, а представителям организаций, работодателям, утверждать либо отклонять предложенный график, а также предоставлять средства для контроля, просмотра отчетной информации.

Для облегчения процесса взаимодействия между работодателем и работником приложение должно предоставить возможность прямого общения путем обмена текстовыми сообщениями в интерактивном режиме. Кроме того, с учетом предоставляемых свобод для работника, необходимо предусмотреть возможность работы у нескольких работодателей одновременно, что должно отразиться на функционале разрабатываемого решения.

Также требуется предоставить возможность фиксирования фактически отработанного работником времени с предоставлением отчетности как работодателю, так и самому работнику.

Учитывая различные сферы деятельности, а также иные факторы, которые могут стать причиной необходимости связи с работодателем в неопределенный момент времени и неопределенном местоположении работодателя, наиболее подходящим типом приложения для работника является мобильное приложение. Представитель работодателя будет взаимодействовать с системой посредством веб-интерфейса.

Таким образом, наиболее оптимальным решением будет разработка рассматриваемой системы в виде комплекса мобильного и веб-приложения, обрабатывающего данные из общего источника. Так как среди мобильных устройств наиболее распространены устройства, работающие под управлением ОС Android, разрабатываемое приложение будет ориентировано под данную операционную систему.

Основной функционал приложения должен предоставлять возможность:

* синхронизации данных о рабочем процессе;
* передачи и приёма текстовых сообщений;
* фиксирования планируемого и фактически отработанного рабочего времени;
* просмотра детальной отчетной информации;
* поддержки и удобства навигации по списку работодателей для работников и работников для работодателей с интерактивным обновлением связанной отчетной информации.

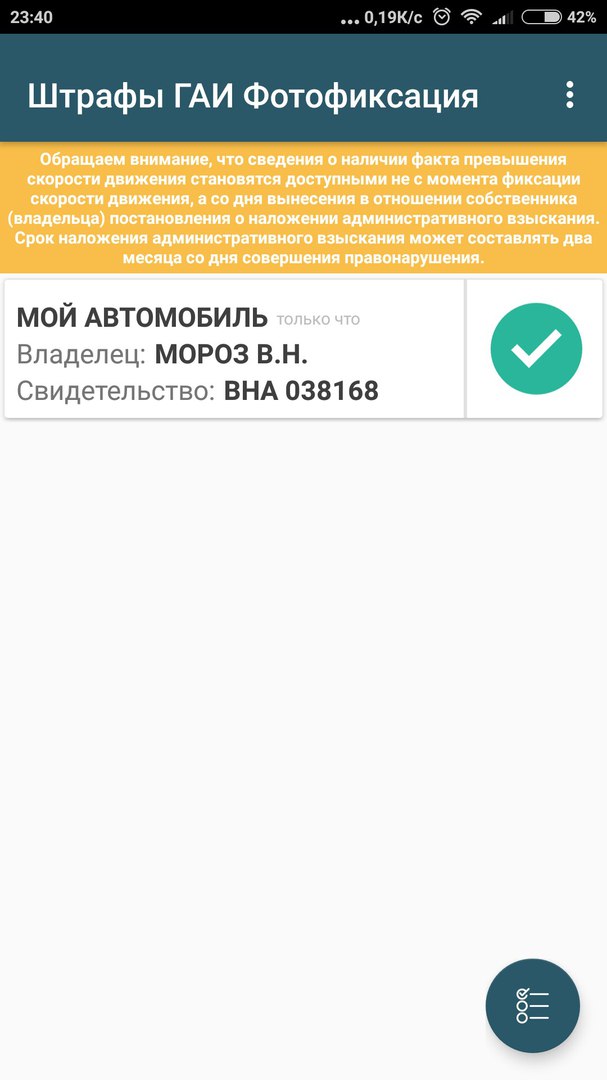
Техническое задание для разрабатываемого продукта содержится в приложении А.

**1.3 Обзор существующих аналогов**

В результате анализа аналогичных программных продуктов было определено, что наиболее схожей по функциональной составляющей с разрабатываемым продуктом является мобильное приложение для операционной системы Android «Штрафы ГАИ».

«Штрафы ГАИ» - приложение предназначено для проверки и оплаты штрафов ГАИ, выписанных на одно или несколько транспортных средств. С этой целью приложение синхронизируется с базой данных Министерства внутренних дел Республики Беларусь и отражает актуальные данные о ваших штрафах. Штрафы можно оплатить через данное приложение.

Вид главной страницы приложения отображен на рисунке 1.1.



**Рисунок 1.1** –Главное окно приложения «Штрафы ГАИ»

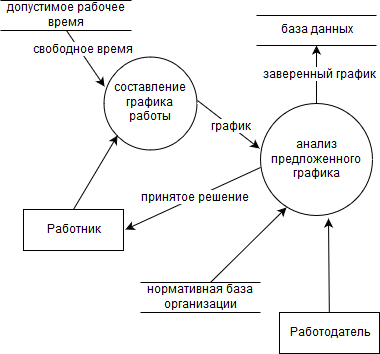
**Таблица 1.1** – Результаты сравнительной характеристики приведенного аналога и разрабатываемого приложения

|  |  |  |
| --- | --- | --- |
| Характеристика | Приложение | |
| «Штрафы ГАИ» | Разрабатываемый продукт |
| Учет всех сотрудников ГАИ | - | + |
| Учет всех водителей | - | + |
| Учет транспортных средств | + | + |
| Возможность выполнять административные операции с работниками | - | + |
| Возможность сотруднику выписывать штрафы водителям | + | + |
| Оплата штрафов, налогов и сборов водителем с помощь банковских карт | - | + |
| Регистрация на прохождение технического осмотра | - | + |
| Переоформление и снятие с учета автомобиля | - | + |

**2 РАЗРАБОТКА АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧ**

**2.1 Функциональная структура**

Процесс взаимодействия работника и работодателя при определении рабочего графика в общем виде можно представить в виде DFD-диаграммы (DFD (диаграммы потоков данных) – диаграммы, предназначенные для демонстрации движения потоков информации в системе [10]), представленной на рисунке 2.1.



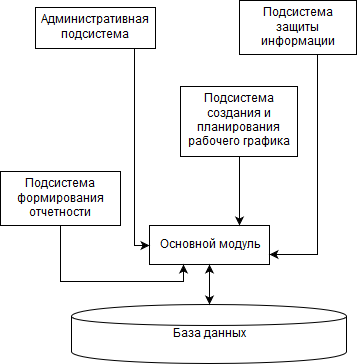
**Рисунок 2.1** – DFD-диаграмма процесса взаимодействия работника и работодателя при определении рабочего графика

Представленная на рисунке 2.1 диаграмма показывает, что процесс определения рабочего графика зависит от ряда факторов, связанных с наличием свободного времени сотрудника, нормативных документов работодателя, а также его личного мнения по поводу предложенного варианта. Приняв решение, работодатель доносит его до работника, а также, в случае одобрения, вносит необходимую информацию в базу данных.

Функционально разрабатываемая система может быть представлена несколькими подсистемами:

* подсистема формирования отчетности (предназначена для предоставления пользователю необходимой отчетной информации);
* подсистема защиты информации (предназначена для обеспечения хранимой информации от несанкционированного доступа; состоит из модулей авторизации и разграничения прав пользователей);
* административная подсистема (предназначена для управления данными о работниках);
* подсистема создания и планирования рабочего графика (предназначена для предоставления пользователю возможности управлять собственным рабочим временем).

Графически функциональная структура представлена на рисунке 2.2.



**Рисунок 2.2** – Функциональная структура

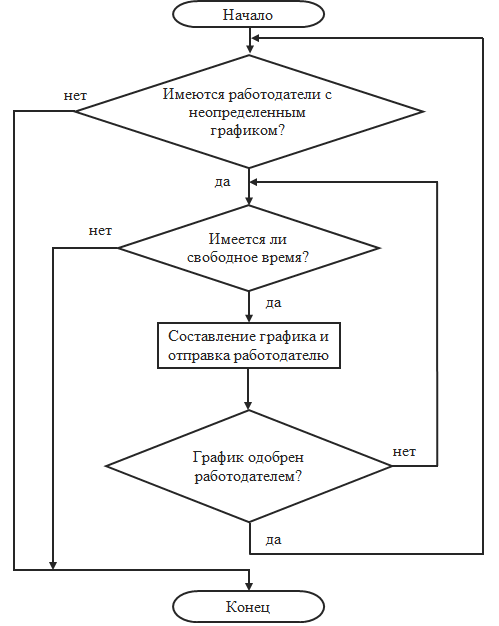
Диаграммы вариантов использования и развертывания отображены в приложениях Д и Е соответственно.

**2.2 Алгоритмическое представление решаемых задач**

Задача управления рабочим графиком может быть решена путем выполнения следующей последовательности действий:

1. Выбрать работодателя. Если весь рабочий график определен, завершить операцию.
2. Выбрать подходящее время. Если всё свободное время распределено, завершить операцию.
3. Отправить данные работодателю. Если график не утвержден, вернуться к шагу 2.
4. Перейти к шагу 1.

Графически алгоритм отображен на рисунке 2.3.



**Рисунок 2.3** – Блок-схема алгоритма создания рабочего графика

**3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**3.1 Выбор и обоснование инструментальных средств**

Так как необходимо реализовать и клиентскую и серверную часть было принято решение разрабатывать приложение с помощью языка программирования JavaScript и его вспомогательных библиотек т.к. разработка обоих частей приложения на одном и том же языке программирования позволит сэкономить время.

Серверная часть будет разрабатываться с помощью Node.js и Express JS. Для хранения данных выбрана MongoDB т.к. эта СУБД отлично работает в связке с Node.js.

Для разработки клиентской части приложения будут использованы следующие средства:

* React.js;
* React-native;
* HTML;
* SASS;
* JavaScript.

В качестве среды разработки будет использован WebStorm.

JetBrains WebStorm – интегрированная среда разработки на JavaScript, CSS & HTML от компании JetBrains, разработанная на основе платформы IntelliJ IDEA. WebStorm обеспечивает автодополнение, анализ кода на лету, навигацию по коду, рефакторинг, отладку, и интеграцию с системами управления версиями. Важным преимуществом интегрированной среды разработки WebStorm является работа с проектами (в том числе, рефакторинг кода JavaScript, находящегося в разных файлах и папках проекта, а также вложенного в HTML). Поддерживается множественная вложенность (когда в документ на HTML вложен скрипт на Javascript, в который вложен другой код HTML, внутри которого вложен Javascript) – то есть в таких конструкциях поддерживается корректный рефакторинг [11].

JavaScript - мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262[8]). JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера.

Express.js - каркас web-приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API. Де-факто является стандартным каркасом для Node.js.

React.js - это библиотека JavaScript, которая используется для создания пользовательского интерфейса. React был создан компанией Facebook, а первый релиз библиотеки увидел свет в марте 2013 года.

React Native – гибридная система от facebook на основе React.js для взаимодействия с родными элементами iOS / Android систем, то есть позволяющая использовать встроенные компоненты, которые предоставляют вышеуказанные платформы. Такой подход позволяет разработчикам, уже знакомым с React.js, в схожей манере разрабатывать нативные приложения. React Native не использует ни браузер, ни WebView — только JavaScript API поверх нативных компонентов. Это работает следующим образом: вы пишете JavaScript код, и он работает с нативными компонентами операционной системы, под которую вы разрабатываете, тем самым перенося достоинства и удобства использования React.js из браузера в мобильные приложения. В отличие от того же PhoneGap, который при возникновении нативного события блокирует поток и передает управление на JS-код, ожидая его инструкций, React Native выполняет JS в отдельном фоновом потоке, взаимодействуя с главным потоком асинхронно, т.е. в потоке JS собирается ряд команд к главному потоку и в определенный момент времени отправляется сгруппированный запрос (batch-запрос), тем самым никак не блокируя главный поток выполнения программы [5].

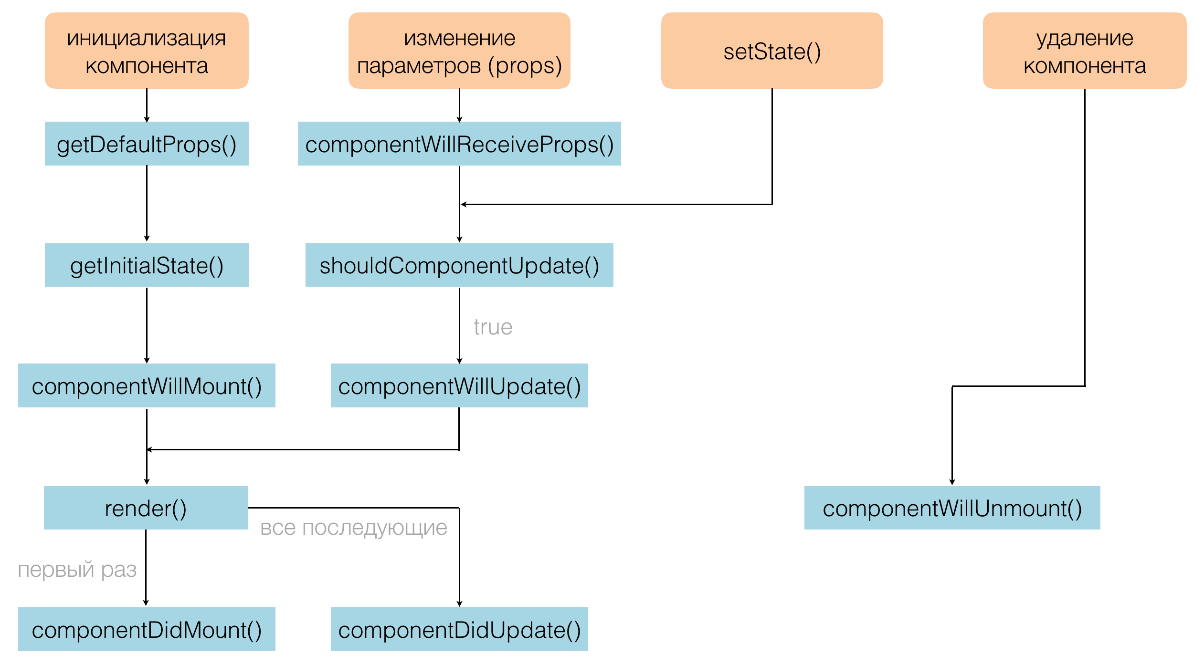
HTML – стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

SASS – это метаязык на основе CSS, предназначенный для увеличения уровня абстракции CSS кода и упрощения файлов каскадных таблиц стилей.

Обоснование выбора инструментальных средств разработки обусловлено тем, что выбранные средства относится к категории свободного программного обеспечения, а также имеют богатый функционал.

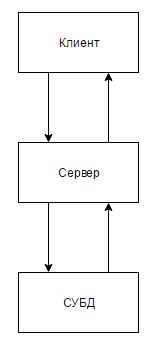
**3.2 Структура объекта разработки**

Клиентская часть объекта разработки состоит из множества компонентов связанных и взаимодействующими друг с другом. Из этих компонентов складывается программная структура приложения. Каждый компонент в процессе работы проходит через ряд этапов жизненного цикла, на каждом этапе выполняется определенная функция. Из компонентов строится модули приложения, один и тот же компонент может быть пере использован в другом модуле программы.



**Рисунок 3.1** – Схема жизненного цикла компонента

Рассмотрим серверную часть, структура которой проще так как основная функциональная нагрузка лежит на клиентской части. Серверная часть принимает запросы от клиента, обрабатывает их, обращаясь к базе данных и отправляет обратно ответ.



**Рисунок 3.2** – Структура серверной части объекта разработки

**3.3 Проектирование базы данных**

В результате анализа предметной области был выявлен ряд сущностей. Сущность – это реальный или представляемый объект, информация о котором должна сохраняться в проектируемой системе. Сущность имеет имя, уникальное в пределах системы, и соответствует некоторому классу однотипных объектов, то есть в системе существует множество экземпляров данной сущности.

База данных – набор сведений, хранящихся некоторым упорядоченным способом. Можно сравнить базу данных со шкафом, в котором хранятся документы. Сами по себе базы данных не представляли бы интереса, если бы не было систем управления базами данных (СУБД). Система управления базами данных – это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных и т.д. В общем СУБД – это система, позволяющая создавать базы данных и манипулировать сведениями из них.

Описание сущностей, выявленных в результате анализа предметной области, приведено в таблице 3.1

**Таблица 3.1** – Описание сущностей

|  |  |
| --- | --- |
| **Сущность** | **Описание** |
| Users | Сущность для личной информации о пользователе |
| Workers | Сущность для хранения данных о сотрудниках государственной автоинспекции |
| CarDrivers | Сущность для хранения данных о водителях |
| Transport | Сущность для хранения транспортных средств |
| TransportTypes | Сущность для хранения списка типов транспортных средств |
| WorkerPosition | Сущность, хранящая список всех возможных должностей сотрудников |
| Penalty | Сущность, хранящая квитанции штрафов |
| Messages | Сущность, хранящая сообщения пользователей |
| Dept | Сущность, хранящая список отделений государственной автоинспекции |
| Clause | Сущность, хранящая статьи административного кодекса регулирующие дорожное движение |

На основе представленного в таблице 3.1 набора сущностей будет спроектирована база данных для приложения.

Далее приведено описание перечня моделей, определенных с учетом характеристик выявленных сущностей

Модель «Users» служит для хранения личных данных пользователей.

Структура данной модели приведена в таблице 3.2.

**Таблица 3.2** – Описание модели «Users»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| \_id | ObjectId | Идентификатор пользователя |
| firstName | String | Фамилия пользователя |
| secondName | String | Имя пользователя |
| middleName | String | Отчество пользователя |
| birthDate | Date | Дата рождения |
| login | String | Логин |
| pass | String | Пароль |
| phoneNum | String | Номер телефона |
| email | String | Электронная почта |

Модель «Workers» служит для хранения данных о сотрудниках государственной автоинспекции.

Структура данной модели приведена в таблице 3.3.

**Таблица 3.3** – Описание модели «Workers»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| \_id | ObjectId | Идентификатор |
| level | String | Уровень доступа |
| id\_post | ObjectId | Идентификатор должности |
| id\_dept | ObjectId | Идентификатор отделения |
| id\_user | ObjectId | Идентификатор пользователя |

Модель «CarDrivers» служит для хранения данных о водителях.

Структура данной модели приведена в таблице 3.4.

**Таблица 3.4** – Описание модели «CarDrivers»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| \_id | ObjectId | Идентификатор |
| level | String | Уровень доступа |
| numDriverLicence | Number | Номер водительского удостоверения |
| dateOfIssue | Date | Дата выдачи ВУ |
| dateOfExpire | Date | Дата истечения срока действия ВУ |
| isBadVision | Boolean | Индикатор необходимости очков или линз для вождения |

Модель «Transport» служит для хранения транспортных средств.

Структура данной модели приведена в таблице 3.5.

**Таблица 3.5** – Описание модели «Transport»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| \_id | ObjectId | Идентификатор пользователя |
| id\_type | ObjectId | Идентификатор типа транспорта |
| number | Object | Автомобильный номер, содержит в себе серию, номер региона и номер |
| model | String | Модель ТС |
| color | String | Цвет |
| date | String | Год выпуска ТС |
| id\_techInspection | ObjectId | Идентификатор прохождения ТО |
| id\_carDriver | ObjectId | Идентификатор водителя |
| fuelType | String | Тип топлива |

Модель «TransportTypes» служит для хранения списка типов транспортных средств.

Структура данной модели приведена в таблице 3.6.

**Таблица 3.6** – Описание модели «TransportTypes»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| \_id | ObjectId | Идентификатор |
| type | String | Тип ТС |

Модель «WorkerPosition» служит для хранения списка должностей.

Структура данной модели приведена в таблице 3.7.

**Таблица 3.7** – Описание модели «WorkerPosition»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| \_id | ObjectId | Идентификатор |
| post | String | Должность |

Модель «Penalty» служит для хранения квитанций штрафов.

Структура данной модели приведена в таблице 3.8.

**Таблица 3.8** – Описание модели «Penalty»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| \_id | ObjectId | Идентификатор |
| id\_clause | ObjectId | Идентификатор статьи админстративного кодекса |
| id\_user | ObjectId | Идентификатор пользователя |
| dateOfContravention | Date | Дата нарушения |
| dateOfinfliction | Date | Дата выдачи квитанции |
| typePenalty | String | Тип наказания |
| summa | String | Суммма штрафа |

Модель «Messages» служит для хранения сообщений пользователей.

Структура данной модели приведена в таблице 3.9.

**Таблица 3.9** – Описание модели «Messages»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| \_id | ObjectId | Идентификатор |
| id\_mailer | ObjectId | Идентификатор отправителя |
| id\_addressee | ObjectId | Идентификатор получателя |
| text | String | Текст сообщения |
| date | Date | Дата отправки |

Модель «Dept» служит для хранения списка отделений.

Структура данной модели приведена в таблице 3.10.

**Таблица 3.10** – Описание модели «Dept»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| \_id | ObjectId | Идентификатор |
| name | String | Название отделения |
| address | String | Адрес отделения |

Модель «Clause» служит для хранения статей административного кодекса.

Структура данной модели приведена в таблице 3.11.

**Таблица 3.11** – Описание модели «Clause»

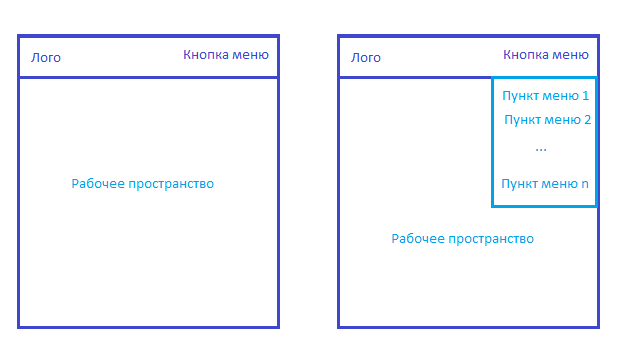
|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| \_id | ObjectId | Идентификатор |
| num\_clause | String | Номер статьи |
| clause | String | Статья |
| sum\_penalty | String | Сумма штрафа |

**3.4 Обоснование и разработка интерфейса**

Расположение элементов пользовательского интерфейса не должно предоставлять затруднений при работе с приложением. Для создания удобного интерфейса необходимо придерживаться четкой визуальной иерархии.

Так как разрабатываемой приложение является мобильным, то разработчик ограничен в экранном пространстве.

Поэтому было принято решения разместить все пункты меню в правом верхнем углу, в выезжающий список. Это позволяет сэкономить рабочее пространство. Примеры макета интерфейса изображены на рисунке 3.2



**Рисунок 3.2** – Макет интерфейса приложения

**4 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ**

**4.1 Результаты реализации**

Разработанное приложение состоит из главного файла «App.js», а также дополнительных модулей.

Файл «App.js» является точкой входа, его выполнение запускает приложение и связывает модули.

Соответствие основных результирующих активностей разработанным модулям приведено в таблице 4.1.

**Таблица 4.1** – Перечень основных форм-активностей и соответствующих им модулей

|  |  |
| --- | --- |
| Модуль | Форма-активность |
| «LoginPage» | авторизация |
| «CalendarPage» | календарь |
| «ClockingPage» | фиксирование времени |
| «MessagesPage» | обмен сообщениями |
| «PersonalDetails» | сведения о сотруднике |
| «TimeSheetPage» | список отчетов |

Перечисленные модули являются компонентами, точнее структурами, наследованными от элемента «Component» с определенными дополнительными свойствами. Каждый модуль содержит перечень главных компонентов, дополненных, в большинстве случаев, набором дополнительных компонентов, необходимых для отображения полей формы, а также для ее прорисовки.

Содержимое файла «App.js» приведено в листинге 4.1.

**Листинг 4.1** – Содержимое класса «App.js»

1. import React, { Component } from "react";
2. import { View, Keyboard, Platform as p } from "react-native";
3. import { bindActionCreators } from "redux";
4. import { connect } from "react-redux";
5. import Router from "../Router/containers/Router";
6. import { setCustomText } from 'react-native-global-props';
7. import { \_navPages, nm } from "../helpers/rootHelper";
8. import \_ from "lodash";
9. import { setRoute } from "../actions/router.actions";
10. import { setKeyboardIndent, setTimeFormat} from
11. "../actions/appSettings.actions";
12. import DeviceClockFormat from "react-native-device-clock-format";
13. const customFont = {
14. style: {
15. fontFamily: "Lato-Regular"
16. }
17. };
18. setCustomText(customFont);
19. class App extends Component {
20. componentWillMount() {
21. this.\_is12hFormat();
22. this.keyboardDidShowListener =
23. Keyboard.addListener('keyboardDidShow', this.keyboardDidShow);
24. this.keyboardDidHideListener =
25. Keyboard.addListener('keyboardDidHide', this.keyboardDidHide);
26. }
27. componentWillUnmount() {
28. this.keyboardDidShowListener.remove();
29. this.keyboardDidHideListener.remove();
30. }
31. keyboardDidShow = e => {
32. this.props.setKeyboardIndent(e.endCoordinates.height);
33. if (!this.props.route.tabBarLockMode) {
34. this.props.setRoute({...this.props.route,
35. tabBarLockMode: true});
36. }
37. };
38. keyboardDidHide = () => {
39. this.props.setKeyboardIndent(0);
40. const needTabBarLockMode = \_.find(\_navPages, {
41. title: this.props.route.title
42. }).tabBarLockMode;
43. if (needTabBarLockMode !== this.props.route.tabBarLockMode)
44. {
45. this.props.setRoute({...this.props.route,
46. tabBarLockMode: needTabBarLockMode});
47. }
48. };
49. \_is12hFormat = () => {
50. if (p.OS === "ios") {
51. DeviceClockFormat.fetch(is12 =>
52. this.props.setTimeFormat(is12));
53. } else {
54. nm.DeviceClockFormat.getTimeFormat(is12 =>
55. this.props.setTimeFormat(is12));
56. }
57. };
58. render() {
59. return (
60. <View style={{flex: 1}}>
61. <Router />
62. </View>
63. );
64. }
65. }
66. const mapStateToProps = state => ({
67. route: state.routerReducer.route
68. });
69. const mapDispatchToProps = dispatch => bindActionCreators({
70. setRoute,
71. setKeyboardIndent,
72. setTimeFormat
73. }, dispatch);
74. export default connect(mapStateToProps, mapDispatchToProps)(App);

Содержимое одного из компонентов (форма списка отчетов) приведено в листинге 4.2.

**Листинг 4.2** – Содержимое компонента «TimeSheetPage»

1. import React, { PureComponent, Component } from 'react';
2. import { Text, View, TouchableHighlight } from 'react-native';
3. import TimeSheetRenderContainer from
4. "../components/TimeSheetRenderContainer";
5. import FlatList from 'react-
6. native/Libraries/CustomComponents/Lists/FlatList';
7. import TimeSheetsInformation from "../components/TimeSheetsInformation";
8. import { connect } from "react-redux";
9. import { bindActionCreators } from "redux";
10. import Loader from "../../../components/Loader/containers/Loader";
11. import {setErrorSettings} from "../../../actions/alert.actions";
12. import {s,c,t} from "../../../helpers/rootHelper";
13. class TimeSheetPage extends Component {
14. constructor(props) {
15. super(props);
16. this.state = {
17. data: [],
18. dataSource: [],
19. openItemId: ""
20. };
21. }
22. \_keyExtractor = (item, index) => index;
23. render() {
24. if (this.props.timesheetsLoading) {
25. return <Loader/>
26. }
27. const formatTime = this.props.is12hFormat ?
28. "h:mm A" : "H:mm";
29. return (
30. <View style={s.TimeSheetContainers}>
31. <TimeSheetsInformation />
32. <FlatList
33. ref="\_flatTimesheet"
34. data={this.props.timesheets}
35. extraData={this.state}
36. keyExtractor={this.\_keyExtractor}
37. renderItem={({item}) =>
38. <TimeSheetRenderContainer
40. navigator={this.props.navigator} rowData={item}
42. formatTime={formatTime}
43. />}
44. removeClippedSubviews={false}
45. />
46. </View>
47. );
48. }
49. }
50. const mapStateToProps = ({ timesheetsReducer, routerReducer,
51. appSettingsReducer }) => ({
52. timesheets: timesheetsReducer.timesheets.length > 0 ?
53. timesheetsReducer.timesheets : [],
54. timesheetsLoading: timesheetsReducer.timesheetsLoading,
55. errorTimesheet: timesheetsReducer.error,
56. route: routerReducer.route,
57. is12hFormat: appSettingsReducer.is12hFormat
58. });
59. const mapDispatchToProps = dispatch => bindActionCreators({
60. setErrorSettings
61. }, dispatch);
62. export default connect(mapStateToProps,
63. mapDispatchToProps)(TimeSheetPage);

В листинге 4.3 приведен пример содержимого файла одного из дополнительных компонентов. Такие компоненты имеет почти каждый модуль, и они предназначены для уменьшения сложности формы путем разделения ее на логически фрагменты и вынесения функционала в подмодули.

**Листинг 4.3** – Содержимое вспомогательного компонента «WorkLog» для формы фиксирования времени

1. import React, { Component } from "react";
2. import { View, Text } from "react-native";
3. import Circle from "./Circle";
4. import {s} from "../../../helpers/rootHelper";
5. export default class WorkLog extends Component {
6. constructor(props) {
7. super(props);
8. }
9. render () {
10. return <View style = {{ flexDirection:'row',
11. marginVertical:6}}>
12. {this.\_renderWorkCircle()}
13. {this.\_renderText()}
14. {this.\_renderWorkTime()}
15. </View>
16. }
17. \_renderWorkCircle = () => {
18. //const { fillWork } = this.props;
19. return <Circle
20. containerBackgroundColor =
21. { 'rgba(242,242,242,0.5)' }
22. circleSize = { 40 }
23. circleWidth = { 8 }
24. circleTintColor =
25. { 'rgba(248,236,41,1.0)' }
26. circleBackgroundColor =
27. { 'rgba(229,229,229,1.0)' }
28. fill =
29. { this.props.fillWork }
30. period = { 10 }
31. />;
32. };
33. \_renderText = () => {
34. return <View style={{
35. alignItems:'flex-start',
36. justifyContent:'center',
37. marginHorizontal: 12,
38. flex: 0.4
39. }}>
40. <Text style={s.textStatistic}> Work duration
41. </Text>
42. </View>;
43. };
44. \_renderWorkTime = () => {
45. return (
46. <View style={{
47. alignItems:'flex-end',
48. justifyContent:'center',
49. flex: 0.5
50. }}>
51. <Text style={this.props.styles}>
52. {this.props.workedTime} </Text>
53. </View>
54. );
55. }
56. }

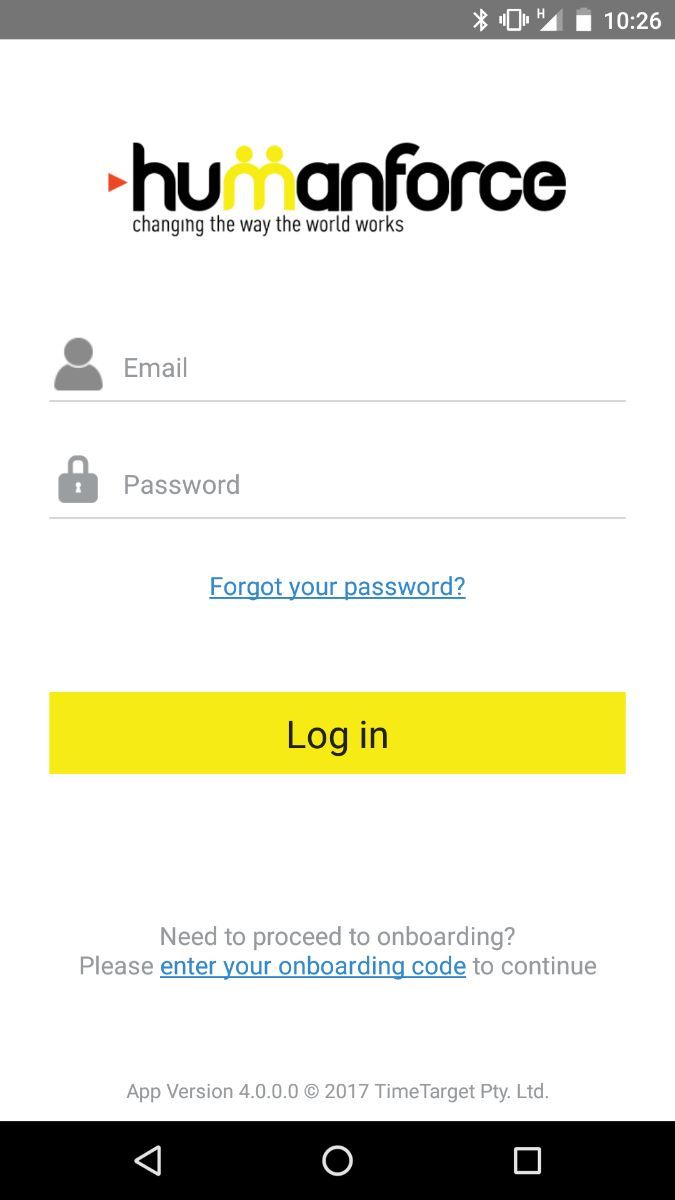
Стилизация форм основана на css-правилах, прописанных в файле «styles.js». Фрагмент содержимого данного файла приведен в листинге 4.4.

**Листинг 4.4** – Фрагмент содержимого файла «styles.js»

1. import { PixelRatio, StyleSheet, Dimensions, Platform as p }
2. from "react-native";
3. import c from "./constants";
4. const {width, height} = Dimensions.get("window");
5. // {with image} width - 2 \* ((width / 25) + 40 + 5);
6. const messageWidth = width - 2 \* (width / 25) - 5;
7. const widthOfBlock = width / 4,
8. circleDiameter = width / 6;
9. export default StyleSheet.create({
10. // set of custom styles
11. flex1: {
12. flex: 1
13. },
14. flex04: {
15. flex: 0.4
16. },
17. flex075: {flex: 0.75},
18. drawerDropButtonCont: {position: "absolute", bottom: 0,
19. right: 0},
20. drawerLoaderCont: {
21. backgroundColor: "rgba(0,0,0,0.3)",
22. justifyContent: "center",
23. alignItems: "center",
24. flex: 1
25. },

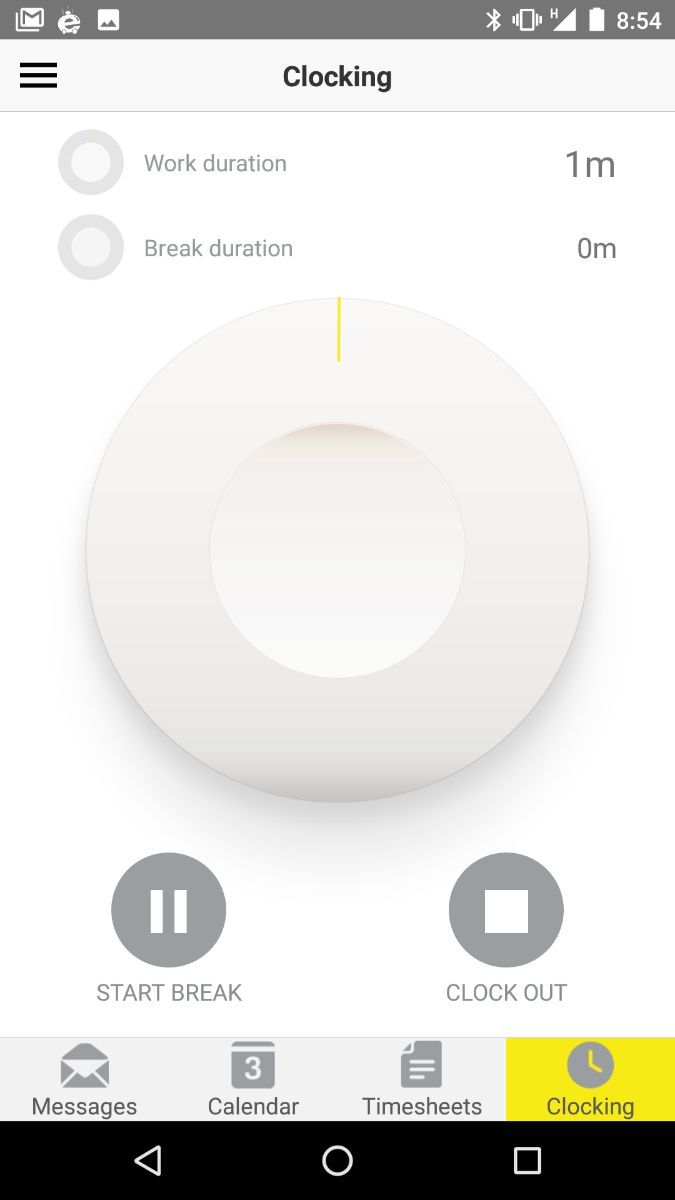
28. //DropDown
29. dropDownIcon: {
30. position: "absolute",
31. right: 10,
32. bottom: 1
33. },
34. dropDownErrText: {color: "red", fontSize: 12},
35. // Authenticate
36. authPageContainer: {
37. flex: 1,
38. padding: 30,
39. paddingBottom: 0,
40. flexDirection: "column",
41. justifyContent: "space-between",
42. backgroundColor: "#fff"
43. },
44. authPageForgotText: {
45. marginTop: 30,
46. marginBottom: 25,
47. fontSize: 15,
48. alignSelf: "center"
49. },
50. authPageUnderButtonContainer: {
51. marginTop: 10
52. },
53. authPageUnderButtonText: {
54. fontSize: 15,
55. textAlign: "center",
56. color: c.GREY,
57. marginBottom: 15
58. },
59. authPageBottomText: {
60. paddingBottom: 10,
61. fontSize: 12,
62. alignSelf: "center",
63. color: c.GREY
64. },
65. …
67. //onboardingFinal
68. onboardingFinalWrapper: {flex:1, flexDirection: "column",
69. justifyContent: "space-between"},
70. onboardingFinalMiddleContainer: {paddingLeft: 50,
71. paddingRight: 50, alignItems: 'center'},
72. onboardingFinalText1: {textAlign: "center", fontSize: 16,
73. color: c.BLACK},
74. onboardingFinalText2: {marginTop: 40,textAlign: "center",
75. fontSize: 16, color: c.GREY},
76. onboardingFinalText3: {marginTop: 40,fontSize: 16,
77. color: c.BLACK},
78. onboardingFinalBottomContainer: {height: 40, textDecorationLine:
79. "underline", fontSize: 14, color: c.BLUE\_TEXT},
81. //Components
82. localSearchWrapper: {flexDirection: "column", width: 30},
83. localSearchTouchable: {width: 30, height: 30},
84. localSearchTouchableView: {justifyContent: "center", flex: 1},
85. localSearchInputContainer: {flexDirection: "column", flex: 1},
86. localSearchTouchableOpacityContainer: {flexDirection: "column",
87. width: 30},
88. localSearchTouchableOpacity: {flexDirection: "column",
89. width: 30},
90. localSearchTouchableOpacityIconView: {justifyContent: "center",
91. flex: 1}
92. });

Вид формы авторизации приведен на рисунке 4.1.



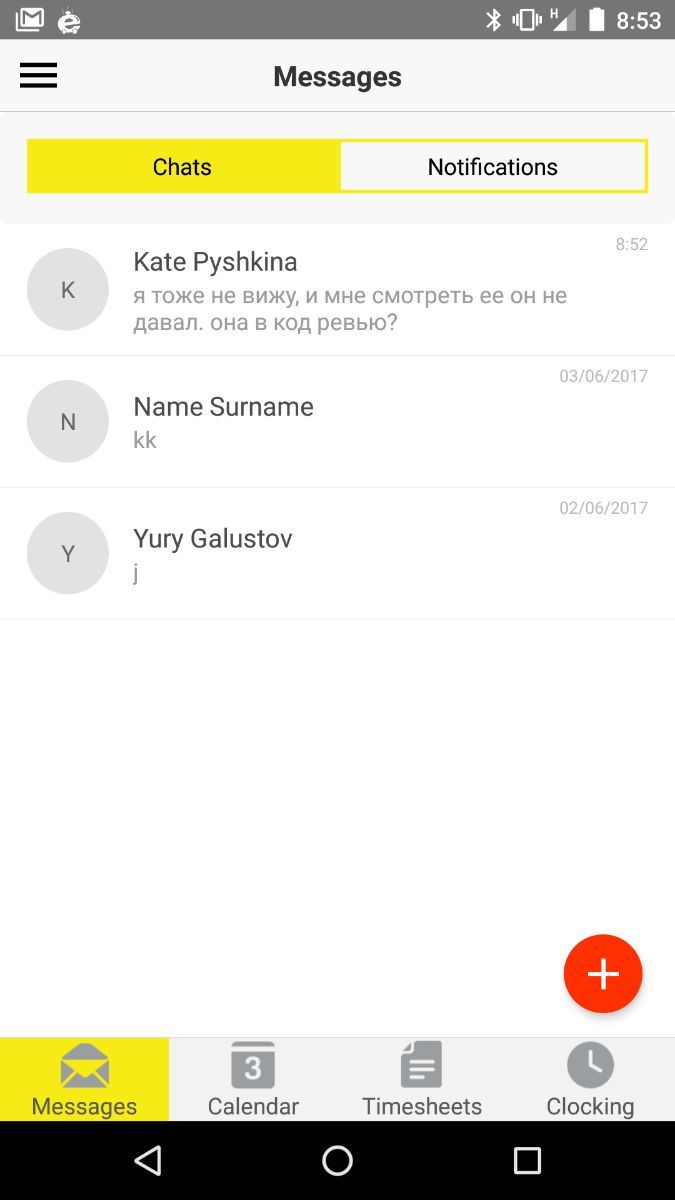
**Рисунок 4.1** – Вид формы авторизации

На рисунке 4.2 представлен вид формы, предназначенной для фиксирования отработанного времени, времени на перерывы.



**Рисунок 4.2** – Вид формы для фиксирования отработанного времени

Вид рабочей формы обмена сообщениями отображен на рисунке 4.3



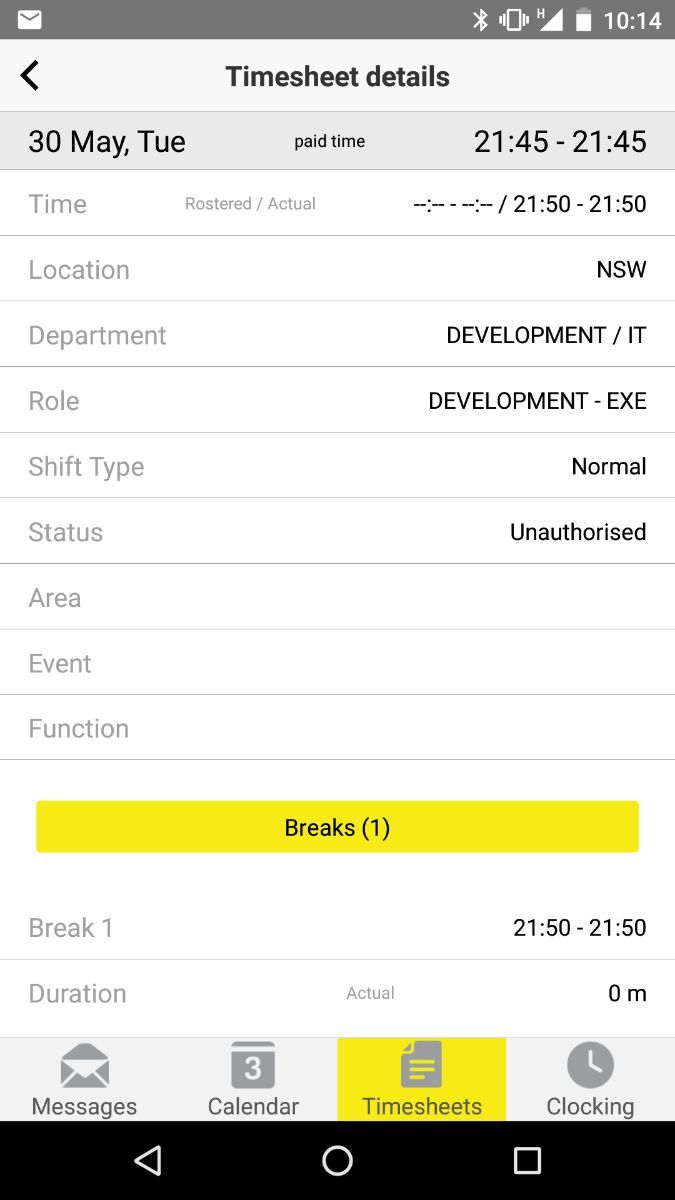
**Рисунок 4.3** – Вид рабочей формы обмена сообщениями

На рисунке 4.4 отображен вид формы со списком отчетов.



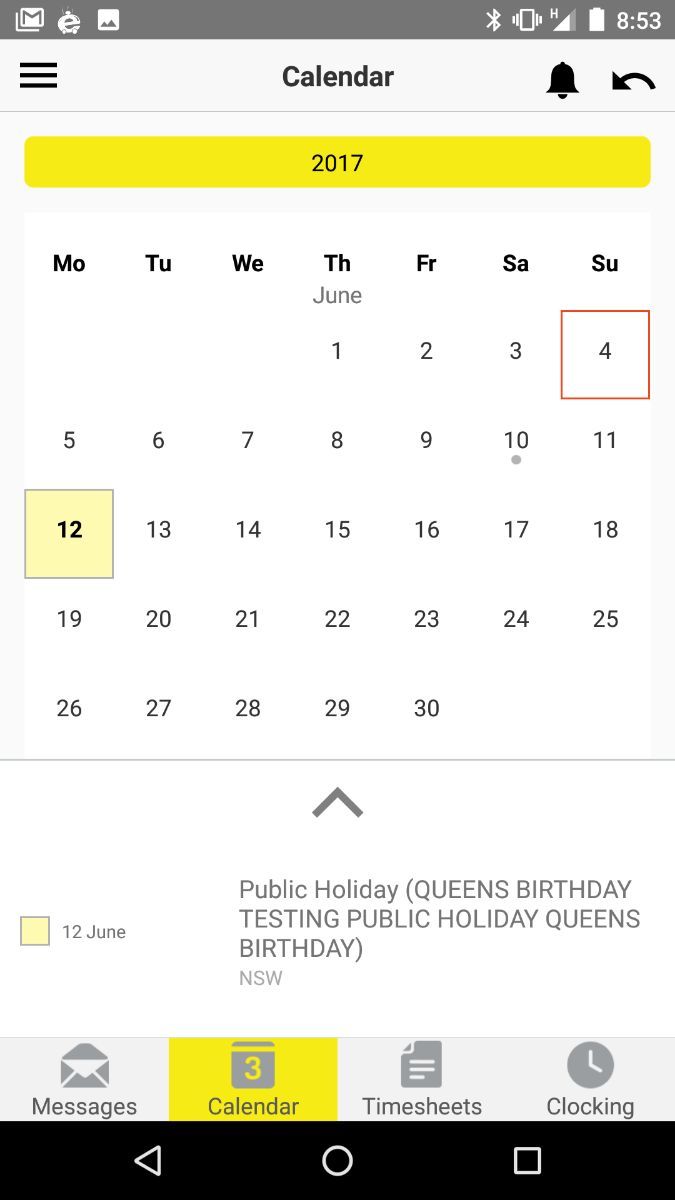
**Рисунок 4.4** – Вид формы со списком отчетов

Вид формы с детализацией отчета представлен на рисунке 4.5.



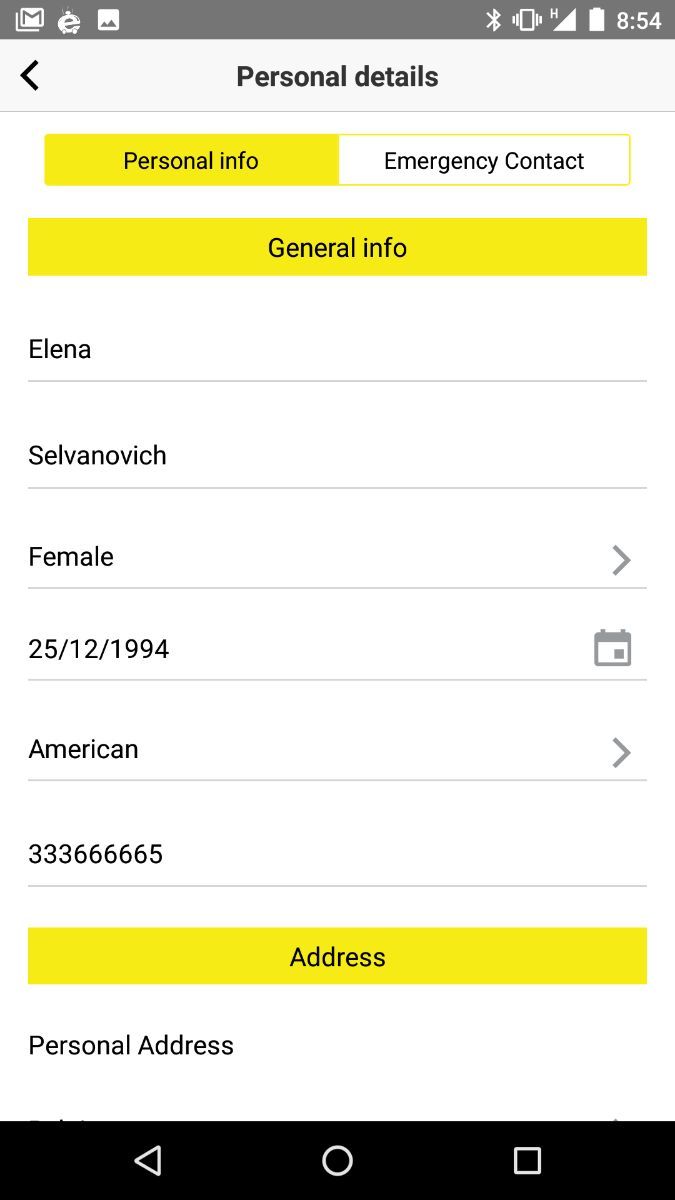
**Рисунок 4.5** – Вид формы с детализацией отчета

На рисунке 4.6 отображен вид формы с календарём.



**Рисунок 4.6** – Вид формы с календарём

Вид формы, предоставляющей информацию о сотруднике, отображен на рисунке 4.7.



**Рисунок 4.7** – Вид формы, предоставляющей информацию

о сотруднике

**4.2 Результаты тестирования**

Тестирование спроектированной системы проводилось еще на этапе разработки и было направлено на исключение следующих ошибок:

* ошибка производительности – ошибка, главным симптомом которой является неудовлетворительная или заниженная производительность (например, низкая пропускная способность или увеличение задержки);
* ошибка безопасности – ошибка, увеличивающая риск несанкционированного проникновения в систему, что позволит просматривать или изменять файлы, не обладая соответствующими правами;
* ошибка потери ресурсов – ошибка, ведущая к потере динамически размещаемых ресурсов, таких как оперативная память или место на диске;
* системная ошибка – ошибка, вероятность обнаружения которой путем тестирования модулей, компонентов или интеграции мала, проявляющаяся в поведении не отдельных компонентов, но системы в целом (например, ошибка производительности, ошибка потери ресурсов, ошибка безопасности).

В процессе разработки отдельные модули выполнялись с различным набором входных параметров, полученные результаты анализировались. Также было проведено тестирование взаимодействия между связанными модулями, проанализирован обмен параметрами. Последним этапом был процесс тестирования полностью функционирующей системы.

Таким образом, тестированию были подвержены как отдельные модули, связи между ними, так и функционирование всей системы. Тестирование в процессе разработки привело к минимуму обнаруженных на последнем этапе тестирования ошибок.

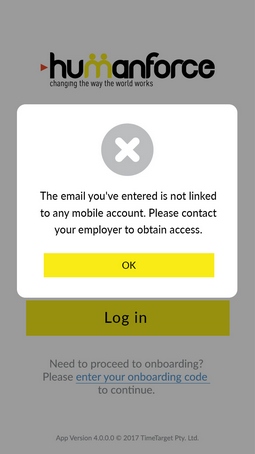
Основные ошибки, обнаруженные в процессе тестирования, были связаны с некорректным отображением некоторых элементов, причиной чему было использование неправильных стилей.

Также были обнаружены некоторые незначительные ошибки в процессе интеграции модулей.

Все обнаруженные ошибки были исправлены, после чего было произведено дополнительное тестирование.

Последняя версия приложения работает стабильно, на действия пользователя откликается корректно, правила валидации отрабатывают, пользователь получает необходимые информационные сообщения, а также подсказки стилевого оформления.

На рисунке 4.8 приведен пример результата отображения диалогового окна с сообщением об ошибке.



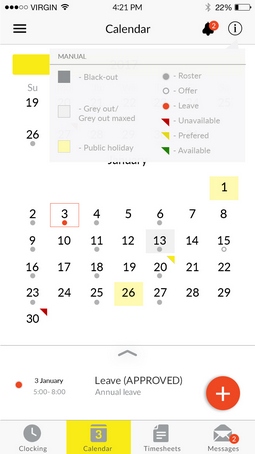
**Рисунок 4.8** – Пример результата отображения диалогового окна с сообщением об ошибке

Результат обмена сообщениями приведен на рисунке 4.9.



**Рисунок 4.9** – Результат обмена сообщениями

Результат отображения событий в календаре приведен на рисунке 4.10.



**Рисунок 4.10** – Результат отображения событий в календаре

**5 ЭКОНОМИЧЕСКАЯ ЧАСТЬ**

**5.1 Обоснование необходимости выведения программного**

**обеспечения на рынок**

Возрастание объемов информации в информационной системе организаций, потребность в ускорении и более сложных способах ее обработки вызывают необходимость автоматизации работы информационной системы, т. е. автоматизации обработки информации.

Для удовлетворения информационных потребностей в рамках конкретной предметной области используются автоматизированные информационные системы (АИС).

В целом АИС можно рассматривать как человеко-машинную систему с автоматизированной технологией получения результатной информации, необходимой для информационного обеспечения персонала и оптимизации процесса управления в предметной деятельности.

Сравним преимущества и недостатки информационных систем с неавтоматизированной (бумажной) и с автоматизированной информационной технологией.

Преимущества неавтоматизированных (бумажных) систем:

* простота организации и (или) установки;
* простота для понимания и освоения;
* не требуются технические навыки;
* гибкость и способность к адаптации для соответствия предметной деятельности.

Однако неавтоматизированные системы обладают серьезными недостатками, такими как медленная, рутинная технология работы, сложность поиска и предоставления информации, риск получения и обработки недостоверной информации.

В АИС проявляется возможность отображения на информационную плоскость всего, что происходит с организацией. Все экономические факторы и ресурсы выступают в единой информационной форме, в виде данных. Основными преимуществами АИС являются: автоматизированная технология, обеспечивающая высокую производительность обслуживания; хранение произвольной информации на оптико-магнитных носителях; простота поиска и предоставления информации; обеспечение достоверности информации.

Так как цель дипломного проектирования – создание системы автоматизации рабочих процессов, предназначенной для сокращения временных затрат на организацию рабочего процесса как со стороны сотрудника, так и со стороны обычным пользователей, имеется возможность достижения положительного экономического эффекта путем выведения данного программного продукта на рынок.

**5.2 Структура работ по созданию программного обеспечения**

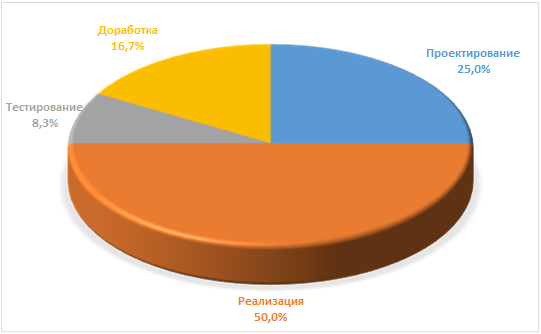
Программные средства вычислительной техники подобно другим промышленным изделиям имеют определенный жизненный цикл.

Под жизненным циклом программного обеспечения (ПО) понимают период от начала разработки нового программного средства до снятия его с эксплуатации у потребителя.

Прежде чем программа станет доступной для широко круга пользователей, должны быть завершены все стадии разработки. К основным стадиям разработки относят следующие:

* анализ требований к проекту (процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения);
* проектирование (целью проектирования является определение внутренних свойств системы и детализации её внешних (видимых) свойств на основе выданных заказчиком требований к программному обеспечению);
* реализация (разработка программного продукта на конкретном языке программирования);
* тестирование продукта (процесс исследования, испытания программного продукта);
* внедрение и поддержка (передача готового продукта пользователю и его поддержка в процессе использования).

Диаграмма распределения времени выполнения работ по разработке дипломного проекта приведена на рисунке 5.1.



**Рисунок 5.1** – Диаграмма распределения времени выполнения работ

Общая продолжительность разработки – 3 месяца.

**5.3 Составление сметы затрат на разработку программного**

**обеспечения**

Стоимостная оценка программного обеспечения и определение экономического эффекта у разработчика предполагают составление сметы затрат, которая в денежном выражении включает следующие статьи расходов:

* материалы и комплектующие (М);
* электроэнергия (Э);
* основная заработная плата исполнителей (Зо);
* дополнительная заработная плата исполнителей (Зд);
* отчисления на социальные нужды (Осн);
* амортизация (А);
* расходы на спецоборудование (Рс);
* накладные расходы (Рн);
* прочие прямые расходы (Пз).

По статье «Материалы и комплектующие» отражаются расходы на магнитные носители, бумагу, красящие ленты и другие материалы, необходимые для разработки ПО. Один из способов определения расходов – прямой подсчет необходимых материалов. Результат подсчета необходимых материалов для разработанного ПО приведен в таблице 5.1.

**Таблица 5.1** – Результат подсчета необходимых материалов

|  |  |  |  |
| --- | --- | --- | --- |
| Материал | Стоимость, руб. | Количество, шт. | Общая сумма, руб. |
| Бумага формата А4, упаковка 500 листов | 5,5 | 1 | 5,5 |
| Компакт-диск DVD-R | 2 | 1 | 2 |
| Папка-скоросшиватель | 1.5 | 1 | 1.5 |
| Всего: | 9 | | |

Таким образом, стоимость основных и вспомогательных материалов составит:

М = 5,5 + 2 + 1,5 = 9 (рублей).

Расчеты затрат на электроэнергию проводятся исходя из продолжительности периода разработки ПО, количества кВт/ч, затраченных на проектирование ПО и тарифа за 1 кВт/ч.

Базовый тариф для прочих потребителей с 01.10.2016 г. составляет 0,30858 руб. за 1 кВт/ч. Данный тариф установлен при соотношении курса белорусского рубля к доллару США 2,0461:1 (согласно Декларации об уровне тарифов на электрическую энергию, отпускаемую республиканскими унитарными предприятиями электроэнергетики ГПО «Белэнерго» для юридических лиц и индивидуальных предпринимателей (от 28.01.2016 г.)).

Тариф, уточненный на день оформления документа и день оплаты, можно рассчитать по формуле (5.1):

(5.1)

где Тн – тариф на электроэнергию, проиндексированный на изменение курса белорусского рубля к доллару США на день оформления платежного документа и день оплаты;

Тб – тариф на электроэнергию, установленный декларацией;

Кн – значение курса белорусского рубля по отношению к доллару США на день оформления платежного документа и день оплаты;

Кб – значение курса белорусского рубля по отношению к доллару США при установлении тарифов на электроэнергию.

Определим тариф на электроэнергию, учитывая значение курса белорусского рубля по отношению к доллару США, на день оплаты равному 1,91 рублей:

Тн = 0,30858 \* (0,19 + 0,81 \* 1,91 / 2,0461) = 0,291 (рублей).

Затраты на электроэнергию необходимо рассчитывать с учетом 8-ми часового рабочего дня, в течение которого активно будет использоваться ноутбук разработчика, а также дополнительное электрооборудование. Среднее потребление энергии ноутбуком – 50 Вт/ч, с учетом восьмичасового рабочего дня – 400 Вт в день, учитывая дополнительное электрооборудование – 1000 Вт в день. Таким образом, за 12 недель потребление энергии составит 84000 Вт, что в денежном эквиваленте составит:

Э = 0,291 \* 84 = 24,5 (рублей).

Общая трудоемкость, плановая численность работников и плановые сроки разработки ПО являются базой для расчета основной заработной платы разработчиков проекта. Оплата труда осуществляется на основе Единой тарифной сетки Республики Беларусь (ETC), в которой даны тарифные разряды и тарифные коэффициенты.

Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле (5.2):

(5.2)

где Зо – основная заработная плата исполнителя;

Тм – месячная тарифная ставка;

К – коэффициент премирования;

n – количество исполнителей, занятых разработкой конкретного ПО;

t – период времени, затраченный на разработку ПО.

Месячная тарифная ставка каждого исполнителя определяется по формуле (5.3):

(5.3)

где Тм – месячная тарифная ставка;

Тм1 – действующая тарифная ставка первого разряда,

Тк – тарифный коэффициент, соответствующий установленному тарифному разряду.

C 01.01.2017 г. тарифная ставка первого разряда составляет 31 руб. Предположим, разработчик ПО имеет 12 разряд. В соответствии с нормами Постановления Министерства труда Республики Беларусь от 23.03.2001 №21 работники 12 разряда имеют тарифный коэффициент 2,84. Повышающий коэффициент будет равен 3. Таким образом, месячная тарифная ставка исполнителя составляет:

Тм = 31 \* 2,84 \* 3 = 264,12 (рублей).

Премия определена в размере 35% от месячной тарифной ставки. Основная заработная плата для одного исполнителя за 3 месяца, в таком случае, составляет:

Зо = 264,12 \* 1,35 \* 3 = 1070 (рублей).

Статья «Дополнительная заработная плата» на конкретное ПО включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по нормативу в процентах к основной заработной плате. Норматив дополнительной заработной платы – 9%. Тогда дополнительная заработная плата будет равна:

Зд = Зо \* 0,09 = 1070 \* 0,09 = 96,3 (рублей).

Затраты по статье «Отчисления на социальные нужды» определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей, рассчитываются по формуле (5.4):

(5.4)

где Осн – затраты на отчисления в Фонд социальной защиты населения и на обязательное страхование;

Зо – основная заработная плата;

Зд – дополнительная заработная плата;

Нсз – норматив отчислений в Фонд социальной защиты населения (%);

Нстр – норматив отчислений на обязательное страхование (%).

На данный момент согласно законодательству норматив отчислений в Фонд социальной защиты населения составляет 34 %. Норматив отчислений на обязательное страхование от несчастных случаев – 0,6%. Следовательно, сумма отчислений составит:

Осн = (1070 + 96,3) \* (34 + 0,6) / 100 = 403.5 (рублей).

По статье «Амортизация» рассчитываются амортизационные отчисления исходя из стоимости основных средств и нематериальных активов, используемых в процессе разработки программного обеспечения, годовой нормы амортизации, сроков эксплуатации оборудования. Затраты, связанные с амортизацией, будут рассчитываться линейным методом за период срока их службы.

В таблице 5.2 приведен перечень основных средств и нематериальных активов, используемых в процессе дипломного проектирования, с указанием их стоимости и актуального срока службы. Для определения стоимости программного обеспечения использована информация с сайта www.softline.by.

**Таблица 5.2** – Информация об основных средствах и нематериальных активах

|  |  |  |
| --- | --- | --- |
| Наименование | Стоимость, рублей | Срок службы, лет |
| Ноутбук ASUS X552C | 975 | 3 |
| Microsoft Office  профессиональный 2013 | 332 | 4 |
| JetBrains WebStorm IDE | 0 | 10 |
| Microsoft Windows 7 Профессиональная | 221 | 3 |

Норма амортизации устанавливается на основе экономически целесообразного срока службы основных средств и должна обеспечить возмещение их морального и физического износа. Годовая норма амортизации определяется согласно формуле (5.5):

(5.5)

где На – годовая норма амортизации;

Сп – первоначальная стоимость основных средств;

Л – ликвидационная стоимость основных средств;

Тн – нормативный срок службы.

Результаты расчета амортизации основных средств и нематериальных активов с учетом времени разработки ПО (12 недель) приведены в таблице 5.3 (ликвидационная стоимость при расчетах не учитывалась).

**Таблица 5.3** – Информация об основных средствах и нематериальных активах

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Годовая норма амортизации, % | Годовые отчисления на амортизацию, рублей | Отчисления за период разработки ПО, рублей |
| Ноутбук ASUS X552C | 33,3 | 324,6 | 81,15 |
| Microsoft Office  профессиональный 2013 | 25 | 83,15 | 20,78 |
| JetBrains WebStorm | - | - | - |
| Microsoft Windows 7 Профессиональная | 33,3 | 73,79 | 18,45 |

Суммарные отчисления на амортизацию за период разработки в итоге составляют:

А = 81,15 + 20,78 + 18,45 = 120,38 (рублей).

Статья «Расходы на спецоборудование» включает затраты средств на приобретение вспомогательных специального назначения технических и программных средств, необходимых для разработки конкретного ПО, включая расходы на их проектирование, изготовление, отладку, установку и эксплуатацию. При разработке рассматриваемого продукта технические и программные средства специального назначения не использовались, поэтому затраты по статье «Расходы на спецоборудование» отсутствуют.

Статья «Накладные расходы» включает затраты, связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды, относятся на конкретное ПО по нормативу в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по организации. Для расчета накладных расходов можно использовать формулу (5.6):

(5.6)

где Рн – накладные расходы;

Зо – основная заработная плата исполнителей;

Нр – норматив накладных расходов.

Примем норматив накладных расходов равным 10%, размер накладных расходов будет равен:

Рн = 1070 \* 0,1 = 107 (рублей).

Статья «Прочие прямые расходы» на конкретное программное обеспечение включает затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по организации, в процентах к основной заработной плате. Для расчета накладных расходов можно использовать формулу (5.7):

(5.7)

где Пз – прочие расходы;

Зо – основная заработная плата исполнителей;

Нп – норматив прочих расходов.

Примем норматив прочих расходов равным 5%, стоимость прочих расходов будет равна:

Пз = 1070 \* 0,05 = 53,5 (рублей).

Общая сумма расходов по смете (плановая себестоимость) рассчитывается по формуле (5.8):

(5.8)

где С – плановая себестоимость;

М – затраты на материалы и комплектующие;

Э – затраты на электроэнергию;

Зо – основная заработная плата исполнителей;

Зд – дополнительная заработная плата исполнителей;

Осн – отчисления на социальные нужды;

А – амортизационные отчисления;

Рс – расхода на спецоборудование;

Рн – накладные расходы;

ПЗ – прочие прямые расходы.

Результаты расчета плановой себестоимости приведены в таблице 5.4.

**Таблица 5.4** – Результаты расчета плановой себестоимости

|  |  |
| --- | --- |
| Статья затрат | Затраты, рублей |
| Материалы и комплектующие (М) | 9 |
| Электроэнергия (Э) | 24,5 |
| Основная заработная плата исполнителей (Зо) | 1070 |
| Дополнительная заработная плата исполнителей (Зд) | 96,3 |
| Отчисления на социальные нужды (Осн) | 403,5 |
| Амортизация основных средств и нематериальных активов (А) | 120,38 |
| Накладные расходы (Рн) | 107 |
| Прочие прямые расходы (Пз) | 53,5 |
| Общая сумма расходов по смете (С) | 1884,2 |

В результате расчета общая сумма расходов по смете равна 1884,2 рублей, т. е. плановая себестоимость разработанного программного продукта составляет 1884,2 рублей. Диаграмма, отображающая влияние статей затрат на себестоимость программного обеспечения, отображена на рисунке 5.2.

Как показывает диаграмма, представленная на рисунке 5.2, основная доля затрат приходится на оплату основной заработной платы исполнителей. Затраты по статьям «Материалы и комплектующие» и «Электроэнергия» составляют наименьшую долю.



**Рисунок 5.2** – Диаграмма, отображающая влияние статей

затрат на себестоимость программного обеспечения

**5.4 Расчет отпускной цены программного обеспечения**

**в случае реализации на рынок**

Отпускная цена продукции формируется исходя из плановой себестоимости производства продукции, всех видов установленных налогов и прибыли, а также качества, потребительских свойств продукции и конъюнктуры рынка.

С учетом действующих в республике нормативных документов отпускная цена на продукцию предприятия рассчитывается по формуле (5.9):

(5.9)

где Оц – отпускная цена изготовителя;

С – плановая себестоимость;

П – прибыль.

Прибыль закладывается в цену по нормативу рентабельности, установленному на предприятии. Расчет производится по формуле (5.10):

(5.10)

где R – норматив рентабельности;

С – плановая себестоимость.

Примем норматив рентабельности равным 20%. Тогда прибыль будет равна:

П = 1884,2 \* 0,2 = 376,84 (рублей).

Стоимость проекта с учетом налога на добавленную стоимость (НДС) представляет собой сумму отпускной цены и НДС. НДС рассчитывается по формуле (5.11):

(5.11)

где НДС – величина налога на добавленную стоимость;

С – плановая себестоимость;

П – прибыль;

Стндс – ставка налога на добавленную стоимость, выраженная в процентах.

Ставка налога на добавленную стоимость с 2010 г. установлена в размере 20 %. НДС, в таком случае, будет равен:

НДС = (1884,2 + 376,84) \* 0,2 = 452,2 (рублей).

Таким образом, прогнозируемая отпускная цена на программное обеспечение с НДС равна:

Оц = 1884,2 + 376,84 + 452,2 = 2713,24 (рублей).

Таким образом, разработчик программного обеспечения может продать заказчику программное обеспечение по рассчитанной цене, что покроет затраты и обеспечит прибыль за разработку проекта.

**5.5 Сравнительный технико-экономический анализ**

**эффективности производства нового продукта**

Для оценки конкурентоспособности разработанного программного продукта на рынке можно сравнить его с ближайшим аналогом – приложение «Штрафы ГАИ». Данная система позволяет оплачивать штрафы ГАИ через систему ЕРИП.

Результаты сравнительной характеристики рассматриваемого аналога и разработанного приложения приведены в таблице 5.5 (знаки «+» и «-» означают наличие и отсутствие рассматриваемой характеристики соответственно).

**Таблица 5.5** – Результаты сравнительной характеристики приложения «Штрафы ГАИ» и разработанного продукта

|  |  |  |
| --- | --- | --- |
| Характеристика | Приложение | |
| «Штрафы ГАИ» | Разрабатываемый продукт |
| Учет всех сотрудников ГАИ | - | + |
| Учет всех водителей | - | + |
| Учет транспортных средств | + | + |
| Возможность выполнять административные операции с работниками | - | + |
| Возможность сотруднику выписывать штрафы водителям | + | + |
| Оплата штрафов, налогов и сборов водителем с помощь банковских карт | - | + |
| Регистрация на прохождение технического осмотра | - | + |
| Переоформление и снятие с учета автомобиля | - | + |

Таким образом, пользователь, отдавший предпочтение разработанному программному продукту, может добиться положительного экономического эффекта за счет более мощного функционала, по сравнению с конкурентом, что даст понижению трудоемкости операций, повышение производительности, повышения уровня сервиса и улучшения показателей основной детальности предприятия.

**ЗАКЛЮЧЕНИЕ**

Результатом дипломного проектирования является мобильное приложение, предназначенное для автоматизации планирования и учета рабочего времени. Разработанный программный продукт в достаточной мере решает выявленную в процессе анализа предметной области проблему и позволяет работнику продуктивно планировать своё рабочее время, работодателю вести учет и взаимодействовать с работником. Учитывая специфику мобильного приложения, пользователи могут взаимодействовать друг с другом независимо от местоположения.

В процессе дипломного проектирования были проделаны следующие работы:

* анализ предметной области и постановка задачи создания программного обеспечения, способного решить выявленные в результате анализа проблемы;
* разработка приложения;
* тестирование разработанного приложения с целью обнаружения и исправления ошибок.

При разработке приложения была использована технология React Native, что обеспечило хорошую расширяемость системы, а также возможность кроссплатформенной реализации.

Использование разработанного продукта приведет к значительному сокращению временных и человеческих затрат в процессе учета рабочего времени, повышению производительности труда и обеспечению необходимой отчетной информацией как работника, так и работодателя.

По результатам расчета сметной стоимости и анализа существующих аналогов можно сделать вывод о том, что разработанный продукт, в случае реализации на рынок, окажется вполне конкурентоспособным.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Бейзер, Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер. – СПб.: Питер, 2004. – 318 с.
2. Методические указания по курсовому проектированию по дисциплине «Системы управления базами данных» для студентов специальности 1-40 01 01 «Программное обеспечение информационных технологий» специализации 1-40 01 01 01 «Компьютерные системы и ИНТЕРНЕТ технологии» / сост. И. Б. Бураченок. – Н.: ПГУ, 2009. – 117с.
3. «High Performance» JavaScript / Николас Закас. – Символ-Плюс, 2012. – 256 с.
4. «Секреты JavaScript ниндзя» / Джон Резиг, Беэр Бибо. – Вильямс, 2016. – 416 с.
5. «Learning React Native»/ Bonnie Eisenman. – Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, 2016. – 432 с.
6. «Программирование мобильных устройств на платформе Google Android» / Голощапов А.Л. – БХВ-Петербург, 2011. – 448.
7. Википедия – свободная энциклопедия [Электронный ресурс] / Универсальная интернет-энциклопедия. – Режим доступа: http://ru.wikipedia.org/wiki/Трёхуровневая\_архитектура. – Дата доступа: 07.06.2017.
8. Википедия – свободная энциклопедия [Электронный ресурс] / Универсальная интернет-энциклопедия. – Режим доступа: http://ru.wikipedia.org/wiki/X-code. – Дата доступа: 07.06.2017.
9. Википедия – свободная энциклопедия [Электронный ресурс] / Универсальная интернет-энциклопедия. – Режим доступа: http://ru.wikipedia.org/wiki/android. – Дата доступа: 09.06.2017.
10. Википедия – свободная энциклопедия [Электронный ресурс] / Универсальная интернет-энциклопедия. – Режим доступа: http://ru.wikipedia.org/wiki/android\_Studio. – Дата доступа: 09.06.2017.
11. Википедия – свободная энциклопедия [Электронный ресурс] / Универсальная интернет-энциклопедия. – Режим доступа: http://ru.wikipedia.org/wiki/web-storm. – Дата доступа: 09.06.2017.
12. Википедия – свободная энциклопедия [Электронный ресурс] / Универсальная интернет-энциклопедия. – Режим доступа: http://ru.wikipedia.org/wiki/Кроссплатформенность. – Дата доступа: 08.06.2017.