

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий  
(наименование института полностью)

Кафедра « Прикладная математика и информатика »  
(наименование кафедры)

02.03.03 Математическое обеспечение и администрирование информационных систем

(код и наименование направления подготовки, специальности)

Технология программирование

(направленность (профиль)/специализация)

## БАКАЛАВРСКАЯ РАБОТА

на тему «Разработка веб-приложения для идентификации посредников при продаже электронных полисов ОСАГО»

Студент	<u>С.С. Шахбазян</u> (И.О. Фамилия)	_____	(личная подпись)
Руководитель	<u>С.В. Мкртычев</u> (И.О. Фамилия)	_____	(личная подпись)
Консультанты	<u>А.В. Прошина</u> (И.О. Фамилия)	_____	(личная подпись)

**Допустить к защите**

Заведующий кафедрой к.т.н., доцент кафедры ПМИ, А.В. Очеповский  
(ученая степень, звание, И.О. Фамилия)

\_\_\_\_\_ (личная подпись)

« \_\_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_\_ г.

Тольятти 2018

## Аннотация

Тема выпускной квалификационной работы «Разработка веб-приложения для идентификации посредников при продаже электронных полисов ОСАГО».

Ключевые слова: РАЗРАБОТКА, ВЕБ-ПРИЛОЖЕНИЕ, ИДЕНТИФИКАЦИЯ ПОСРЕДНИКОВ, ОСАГО, СТРАХОВАНИЕ.

Объектом исследования идентификация посредников при продаже электронных полисов ОСАГО.

Предметом исследования является приложение для идентификации посредников при продаже электронных полисов ОСАГО.

Методы исследования: web-технологии, методы построения информационных систем, объектно-ориентированное программирование.

В теоретической части рассматривается вопрос о разработке веб-приложений. Проводится сравнительный анализ языков программирования: PHP, Ruby и Python. Изучив некоторые критерии было принято решение, что для разработки данного веб-приложения оптимальным будет использование языка программирования PHP. Кроме того, были выбраны языки разметки, HTML и CSS. Для выбора СУБД был проведен сравнительный анализ MongoDB и MySQL, принимая во внимание, тот факт, что MySQL и MongoDB по многим параметрам схожи, но MySQL доступнее и удобнее в использовании было принято решение использовать именно эту систему управления базами данных.

Практическая часть посвящена концептуальному и логическому моделированию веб-приложения для идентификации посредников при продаже электронных полисов ОСАГО. С помощью данного приложения можно проверять на наличие посредников в базе данных компаний, предоставляющих услуги страхования ОСАГО.

Данная работа состоит из 49 страниц, включая введение, 3 главы, заключение, список используемой литературы из 25 источников.

## **ABSTRACT**

The title of the graduation work is Development of web applications for the identification of intermediaries.

Key words: web application, OSAGO, identification.

The object of research the electronic policies of OSAGO.

The subject of this is the identification of intermediaries in the sale of OSAGO electronic policies.

The theoretical part deals with the development of web applications. Analysis of the development tools was carried out, the result of which was: the choice of PHP programming language, and markup languages HTML and CSS.

In this work it is shown that people trying to buy an electronic OSAGO policy fall on intermediaries who can deceive and commit fraudulent actions.

The work is of interest for a wide range of customers. The results show that the existing system is inefficient and requires improvements to optimize the process.

The practical part is devoted to the conceptual and logical modeling of a web application for the identification of intermediaries in the sale of OSAGO electronic policies. With the help of this application, you can check for the presence of intermediaries in the database of companies that provide insurance for OSAGO.

This work consists of 49 pages, including an introduction, three chapters, conclusion, a list of used literature from 25 sources.

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ .....	5
Глава 1. АНАЛИЗ ИНФОРМАЦИИ И ПОСТАНОВКА ЗАДАЧИ .....	7
1.1 Основные этапы разработки веб-приложения .....	7
1.2 Выбор и описание архитектуры приложения .....	9
1.3 Оформление электронного полиса ОСАГО.....	11
1.4 Достоинства электронного ОСАГО .....	12
1.5 Недостатки электронного полиса .....	13
1.6 Посредники в сфере электронных полисов ОСАГО.....	14
1.7 Существующие информационные системы.....	15
1.8 Мошенничество в сфере автострахования.....	17
1.9 Данные для идентификации посредников .....	17
Глава 2. РАЗРАБОТКА КОНЦЕПТУАЛЬНОЙ МОДЕЛИ.....	20
2.1 Формирование требований к веб-приложению .....	20
2.2 Разработка алгоритма работы веб-приложения .....	22
2.3 Диаграмма вариантов использования .....	23
2.4 Диаграмма последовательности.....	24
2.5 Алгоритм работы проверяющей функции, проводящей проверку .....	25
Глава 3. РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ ИДЕНТИФИКАЦИИ ПОСРЕДНИКОВ ПРИ ПРОДАЖЕ ЭЛЕКТРОННЫХ ПОЛИСОВ ОСАГО ..	27
3.1 Выбор средств реализации.....	27
3.2 Выбор языков разметки .....	33
3.3 Выбор СУБД.....	34
3.4 Реализация интерфейса веб приложения .....	38
ЗАКЛЮЧЕНИЕ .....	41
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ .....	42
Приложение А. Фрагмент исходного кода проверяющей функции .....	44
Приложение Б. Фрагмент кода реализации формы для проверки.....	48

## ВВЕДЕНИЕ

В 2015 году законодатели внесли поправки в текущую редакцию закона об обязательном автостраховании. С того времени каждый автомобилист имеет право оформлять электронные полисы ОСАГО. В течение примерно 1,5 лет после вступления в силу законопроекта многие страховщики отказывались в предоставлении новой услуги гражданам. Они заявляли, что существующие технические возможности не позволяют удовлетворить в полной мере запросы автомобилистов, поэтому компании не могут продавать полисы ОСАГО в электронном виде.

В итоге до начала 2017 года, несмотря на введение нового типа документов, большинство страховщиков своими действиями вынуждали граждан оформлять автострахование на бумажных бланках. Такой подход позволял компаниям навязывать автомобилистам дополнительные услуги.

Но с начала 2017 года в силу вступило новое требование Центробанка, обязывающее всех страховщиков предоставлять гражданам доступ к оформлению электронных полисов ОСАГО. Покупка страховки осуществляется непосредственно через сайты компаний.

С появлением электронного полиса в сферу автострахование пришли и посредники, а приложений и программа для работы с ними еще не разработаны. Поэтому тема моей выпускной квалификационной работы актуальна.

**Объектом исследования** являются идентификация посредников в сфере электронных полисов ОСАГО.

**Предметом исследования** является веб-приложение для идентификации посредников при продаже электронных полисов ОСАГО.

**Целью** данной бакалаврской работы является разработка веб-приложения для идентификации посредников при продаже электронных полисов ОСАГО.

Для достижения поставленной цели, необходимо решить следующие задачи:

- провести анализ сферы электронного страхования автомобилей;

- определить структуру;
- спроектировать модель будущего веб-приложения;
- выбрать средства разработки данного веб-приложения.

В ходе данной выпускной квалификационной работы (ВКР) будет разработана модель веб-приложение для идентификации посредников при продаже электронных полисов ОСАГО.

Выпускная квалификационная работа (ВКР) состоит из введения, трех глав и заключения.

Во введении описывается актуальность рассматриваемой темы, определяются объект и предмет ВКР работы, ставится цель и выявляются задачи.

В первой главе проведен анализ информации по проблеме исследования и выполнена постановка задачи на разработку веб-приложения.

Во второй главе описывается разработка и реализация проектных решений веб-приложения для работы с посредниками в сфере электронного страхования автовладельцев.

В третьей главе описываются средства реализации.

В заключении подводятся итоги разработки веб-приложения, формируются окончательные выводы по рассматриваемой теме.

# Глава 1. АНАЛИЗ ИНФОРМАЦИИ И ПОСТАНОВКА ЗАДАЧИ

## 1.1 Основные этапы разработки веб-приложения

Общее количество этапов разработки обычно варьируется от пяти до восьми, но картина остается практически такой же. Разберем семь этапов разработки веб-приложения.

Первый этап - сбор информации: цель, основные задачи и целевая аудитория.

Этот этап, этап анализа и исследования, определяет, как будут выглядеть последующие шаги. Самая важная задача на этом этапе - получить четкое представление об основных задачах, которые будет выполнять будущее веб-приложение. Определение целевой аудитории, которая будет являться пользователями вашего веб-приложения.

Различные типы сайтов предоставляют посетителям различную функциональность, что означает, что в соответствии с целями следует использовать разные технологии. Хорошо описанный и подробный план, составленный на основе данных предварительной разработки, может защитить клиента от использования дополнительных ресурсов при решении неожиданных проблем, таких как изменение дизайна или добавление функциональности, которая изначально не планировалась.

Второй этап - планирование: создание карты веб-приложения и каркаса.

На этом этапе цикла разработки сайта разработчик создает данные, которые могут дать клиенту возможность судить о том, как будет выглядеть весь сайт. На основе информации, собранной на предыдущем этапе, создается карта сайта. Карта сайта позволяет понять, как выглядит внутренняя структура веб-приложения, но не описывает интерфейс пользователя. Каркас представляет собой визуальное представление пользовательского интерфейса, который планируется создать. Но он не содержит никаких элементов дизайна,

таких как цвета, логотипы и т. Д. Он описывает только элементы, которые будут добавлены на страницу и их местоположение.

Третий этап – разработка дизайна: макеты страниц, обзор базового функционала веб-приложения.

На этапе проектирования формируется само веб-приложение. На этом этапе создается весь визуальный контент. Еще раз, вся информация, которая была собрана на первом этапе, имеет решающее значение. Клиент и целевая аудитория должны учитываться при работе над дизайном.

Макет приложения является результатом работы дизайнера. Это может быть графический эскиз или фактический графический дизайн. Основная функция макета - представлять структуру информации, визуализировать контент и демонстрировать базовый функционал. Макеты содержат цвета, логотипы, изображения и могут дать общее представление о будущем продукте.

Четвертый этап - написание кода.

На этом этапе идёт разработка самого приложения. Графические элементы, которые были разработаны на предыдущих этапах, должны использоваться для создания реального веб-приложения. Обычно сначала создается основная страница, а затем добавляются все подстраницы в соответствии с иерархией веб-сайта, которая была ранее создана.

Пятый этап - тестирование, обзор и запуск.

Тестирование, вероятно, является самой обычной частью процесса. Каждая отдельная ссылка должна быть проверена, чтобы убедиться, что среди них нет нерабочих. Нужно проверить каждую форму, каждый скрипт, запустить программное обеспечение проверки орфографии, чтобы найти возможные опечатки. Используйте проверки кода, чтобы проверить, соответствует ли ваш код текущим веб-стандартам. Действительный код необходим, например, если для вас важна совместимость с несколькими браузерами.



После проверки и повторной проверки вашего веб-приложения пришло время загрузить его на сервер. Для этой цели используется программное обеспечение FTP (File Transfer Protocol). После того, как вы развернули файлы, вы должны запустить еще один окончательный тест, чтобы убедиться, что все ваши файлы установлены правильно.

Шестой этап - техническое обслуживание: Мониторинг мнения и регулярное обновление.

Важно помнить, что веб-приложение — это скорее услуга, чем продукт. Недостаточно «доставить» веб-приложение пользователю. Нужно убедиться, что все работает правильно.

Система обратной связи, добавленная на сайт, позволит вам обнаружить возможные проблемы, с которыми сталкиваются конечные пользователи. Задача с наивысшим приоритетом в этом случае - исправить проблему так быстро, как только сможете. Если этого не будет сделано пользователи предпочтут использовать другой сайт вместо того, чтобы смириться с неудобствами.

Другая важная вещь – поддержка веб-приложения в актуальном состоянии. При использовании CMS, регулярные обновления позволят избежать ошибок и снизить риски нарушения безопасности.

## **1.2 Выбор и описание архитектуры приложения**

При разработке веб-приложений используются две наиболее распространенные архитектуры.

Первая архитектура - классическую архитектуру “клиент-сервер”;

Система может легко использовать распределенную обработку, используя архитектуру клиент-сервер. В этой архитектуре система баз данных состоит из двух частей: интерфейсного или клиентского, а также внешнего или серверного.

Клиент - это приложение базы данных, которое инициирует запрос на операцию, выполняемую на сервере базы данных. Он запрашивает, обрабатывает и представляет данные, управляемые сервером. Рабочую станцию клиента можно оптимизировать для своей работы. Например, клиенту может не потребоваться большая емкость диска, или он может воспользоваться графическими возможностями. Часто клиент работает на другом компьютере, чем сервер базы данных. Многие клиенты могут одновременно работать с одним сервером.

Сервер запускает программное обеспечение и обрабатывает функции, необходимые для совместного доступа к данным общего доступа. Сервер получает и обрабатывает запросы, исходящие из клиентских приложений. Компьютер, управляющий сервером, может быть оптимизирован для выполнения своих обязанностей. Например, серверный компьютер может иметь большую емкость диска и быстрые процессоры.

Вторая архитектура – многоуровневая архитектура

Традиционная многоуровневая архитектура имеет следующие компоненты:

- процесс клиента или инициатора, который запускает операцию;
- один или несколько серверов приложений, которые выполняют части операции. Сервер приложений содержит большую часть логики приложения, обеспечивает доступ к данным для клиента и выполняет некоторую обработку запросов, тем самым удаляя часть загрузки с сервера базы данных. Сервер приложений может служить интерфейсом между клиентами и несколькими серверами баз данных и может обеспечить дополнительный уровень безопасности;
- конечный сервер или сервер базы данных, который хранит большую часть данных, используемых в операции.

Эта архитектура позволяет использовать сервер приложений для выполнения следующих действий:

- Проверка учетных данных клиента, например веб-браузера
- Подключение к серверу Oracle Database
- Выполнение запрошенной операции от имени клиента
- Если используется прокси-аутентификация, то идентификатор клиента поддерживается на всех уровнях соединения.

Сервисно-ориентированная архитектура (SOA) - это многоуровневая архитектура, в которой функциональность приложения инкапсулируется в сервисах. Службы SOA обычно реализуются как веб-службы. Доступ к веб-службам возможен с помощью протокола HTTP и основаны на наборе открытых стандартов на основе XML, таких как WSDL и SOAP.

Так как, большинство возможностей многоуровневой архитектуры в веб-приложении разрабатываемым в данной бакалаврской работе не будут использоваться было принято решение выбрать классическую архитектуру «клиент-сервер».

На рисунке 1 представлена архитектура веб-приложения разработка, которого проводится в данной выпускной квалификационной работе.

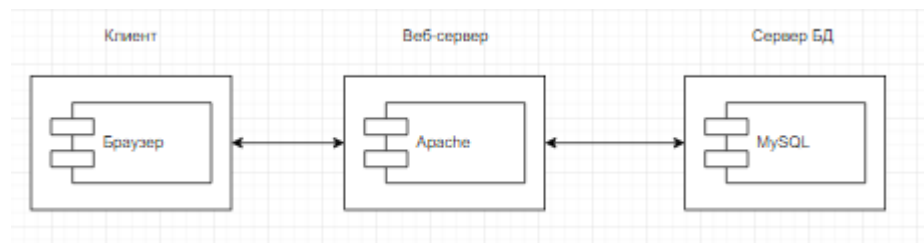


Рисунок 1 - Архитектура веб-приложения «клиент-сервер»

Клиентом будет выступать браузер.

Apache является сервером приложения и синхронизирует работу компонентов системы и создает между ними связь.

Базы данных хранятся на сервере доступ, к которому ограничен и напрямую не взаимодействует с клиентом, и поэтому, повышается безопасность системы.

### 1.3 Оформление электронного полиса ОСАГО

Электронный ОСАГО является с точки зрения действующего законодательства аналогом бумажного страхового договора. Оба документа имеют одинаковую юридическую силу. Электронный полис продается через официальные сайты страховых компаний. Документ оформляется удаленно. В отличие от бумажных полисов электронный ОСАГО не печатается на бланке Гознака. Соответственно, документ не имеет никаких защитных механизмов типа водяных знаков. Разница в оформлении является единственным отличием электронной страховки ОСАГО от стандартных полисов. При ее покупке автомобилисту необходимо указывать те же требования, что он приводил в заявлении на приобретении автостраховки на бумажном носителе: характеристики автомобиля, ФИО, водительский стаж и другое.

Правила и порядок оформления электронного полиса регулируется законом об ОСАГО. В течение первых нескольких месяцев после введения соответствующих поправок данная норма работала в тестовом режиме. Начиная с 1-го октября 2015 года все граждане имеют возможность приобрести электронный ОСАГО на официальных интернет-ресурсах страховых компаний, которые предоставляют услугу онлайн-продажи таких полисов.

#### **1.4 Достоинства электронного ОСАГО**

Основное достоинство электронного полиса заключается в том, что его можно оформить удаленно без посещения офиса страховой компании. Этот факт приобретает особую актуальность для жителей удаленных регионов и занятых граждан.

Онлайн-страхование также обладает следующими преимуществами в сравнении со стандартной процедурой:

- страховой договор нельзя потерять, так как он хранится на электронных носителях;

- электронный полис изготавливается в трех экземплярах, один из которых хранится в базе данных РСА, второй – у страховой компании, третий – у страхователя (автомобилиста);
- страховые компании при оформлении страховки на автомобиль не могут «навязать» дополнительные услуги и опции, нередко существенно увеличивающие расходы автомобилиста;
- низкая вероятность обмана при условии соблюдения мер предосторожности;
- люди, проживающие в отдаленных регионах, теперь могут оформить страховой полис в различных компаниях, когда как ранее у них имелся доступ только к 1-2 организациям.

### **1.5 Недостатки электронного полиса**

Несмотря на то, что у электронного ОСАГО имеется множество достоинств в сравнении с бумажными полисами, этот тип документа обладает несколькими отрицательными чертами, среди которых можно выделить:

- онлайн заказ полиса остается недоступным для тех граждан, которые ранее не оформляли страховку на автомобиль (электронный ОСАГО оформляется при условии, если в базе РСА содержится информация об автомобилисте);
- страховые компании не возвращают внесенные средства за полис, если владелец транспортного средства при оформлении договора указал неверные данные (документ в таком случае признается недействительным, а страховщики трактуют подобные ситуации как предоставление ложной информации);
- длительная проверка подлинности электронного полиса инспекторами ГИБДД (в особенности, когда возникают сбои в работе технических устройств);

- наличие множества ошибок в работе электронной системы оформления страховых договоров.

## **1.6 Посредники в сфере электронных полисов ОСАГО**

Рост e-ОСАГО и в целом электронных продаж в страховании заставил регулятора снова вернуться к вопросу о допуске посредников к электронным продажам. Так, заместитель председателя Банка России Владимир Чистюхин на международной конференции по страхованию в июле заявил, что запреты, которые существуют сейчас в плане допуска посредников в электронные продажи, необоснованны. ЦБ планирует изменить эту ситуацию, но необходимо создавать стандарты и для посредников, отметил представитель регулятора. Как вариант контроля над посредниками Чистюхин назвал создание реестра брокеров и агентов, которые будут заниматься электронными продажами. Законопроект о допуске посредников и упрощении для страховщиков идентификации клиентов внесен в Госдуму еще в апреле, но пока не дошел даже до первого чтения. «Законопроектом предлагается установить возможность заключения и сопровождения договоров страхования страховыми агентами — юридическими лицами и страховыми брокерами в виде электронных документов, — говорится в пояснительной записке к документу. — При этом права потребителей страховых услуг защищены нормами о полной ответственности страховой организации за деятельность страховых посредников, действующих от ее имени во взаимоотношениях с потребителями». Внесение предложенных изменений позволит сделать страхование более доступной услугой для самого широкого круга страхователей, а взаимодействие потребителей и страховых организаций — максимально оперативным и существенно менее затратным, уверены авторы законодательной инициативы. Всероссийский союз страховщиков в целом идею поддерживает, подчеркивая, что развитие электронных продаж — одна из стратегических задач отрасли. Однако союз страховщиков не хочет пока

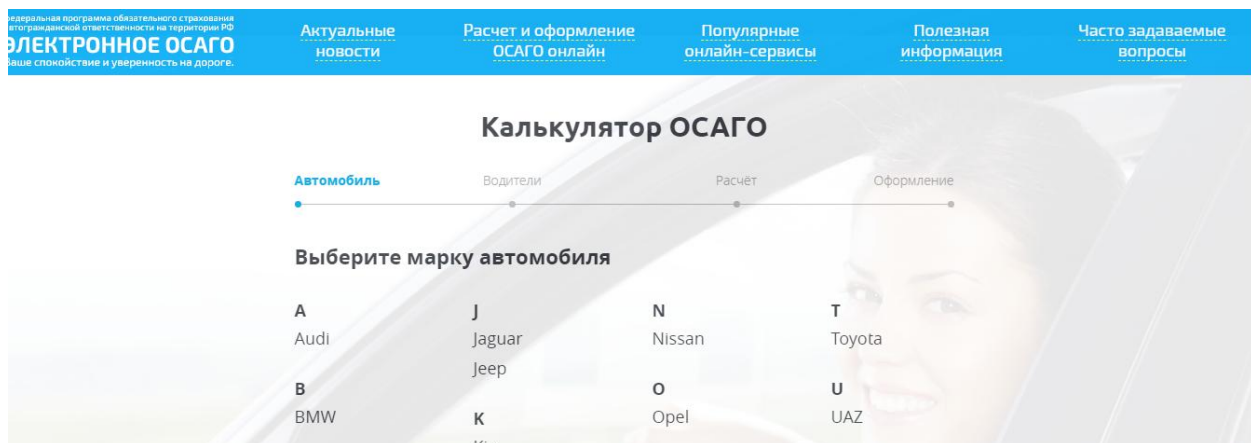
форсировать претворение идеи в жизнь применительно к e-ОСАГО. Отношение участников рынка к этой идее радикально отличаются. Кто-то уверен, что посредников нужно разрешить немедленно: это повысит доступность ОСАГО и конкуренцию, а также поможет бороться с мошенниками. Другие, напротив, полагают, что допуск посредников даст злоумышленникам простор для творчества и только усугубит проблемы.

### **1.7 Существующие информационные системы.**

На сегодняшний день существует ряд информационных систем, которые способствуют поддержке и развитию электронных полисов ОСАГО. Сред всего ряда информационных систем, стоит выделить несколько более популярных.

Первым стоит выделить системы по оформлению электронных полисов ОСАГО несмотря на то, что, у каждой компании, которая предоставляет услуги страхования, есть свои системы по оформлению электронных полисов, существует еще много систем через которые можно оформить электронный полис. Это увеличивает выбор клиентов и благодаря этому, каждый клиент сможет оформить электронный полис выбрать, компанию по тем критериям, которые ему важны.

Далее хочется выделить калькулятор ОСАГО. Калькулятор ОСАГО это информационная система, благодаря которой можно вычислить стоимость покупки ОСАГО для всех желающих. Такая система есть практически у каждой крупной компании, но, если по каким-то причинам, не хочется пользоваться калькулятором крупной компании, есть еще много сайтов, предоставляющих данную услугу. На рисунке 2 показан пример сайта предоставляющую услугу калькулятора ОСАГО



Рисунке 2 - Пример сайта предоставляющую услугу калькулятора ОСАГО

Кроме калькулятора, найти себе приемлемые условия страхования может помочь сервис для сравнения услуг, которые предоставляют страховые компании. Кроме этого, такие сервисы на своих сайтах предоставляют описание практически всех видов страхования. Также они собрали наиболее полную информацию о всех страховых компаниях, об их акциях, нововведениях, а также новейшие новости из данной сферы. На рисунке 3 показан пример сравнения цен, на продажу электронных полисов ОСАГО





	<b>ОСАГО «ЭРГО»</b> 14 232 ₽ Стоимость	Без скидки Скидка ?
	<b>ОСАГО «ВСК»</b> 14 232 ₽ Стоимость	Без скидки Скидка ?
	<b>ОСАГО «РЕСО-Гарантия»</b> 14 232 ₽ Стоимость	Без скидки Скидка ?
	<b>ОСАГО «Согласие»</b> 14 232 ₽ Стоимость	Без скидки Скидка ?

Рисунок 3 - Сравнения цен, на продажу электронных полисов ОСАГО

Но наиболее важным является сервис проверки подлинности электронных полисов. Такую информацию предоставляет, только один сервис – это сервис



российского союза автостраховщиков. Кроме проверки подлинности на данном ресурсе, можно прочитать о всех нововведениях в законе об автостраховании.

### **1.8 Мошенничество в сфере автострахования**

На данный момент, по данным страховых компаний, через посредников совершается более 50% покупок электронных полисов ОСАГО, что по закону посредничество в электронном ОСАГО прямо запрещено. Конечный покупатель расплачивается с посредником наличными, а посредник оплачивает полис на сайте компании со своей карты, покупатель берет распечатанный полис и уходит. Однако потом, когда клиент уже ушел, этот посредник начинает просто отменять транзакцию по оплате, из-за чего полис становится недействительным. Самое неприятное — автовладелец в такой ситуации не знает, что его полис уже недействителен. Страховые компаниям сложно найти контакты клиента, а иногда и вовсе не могут связаться с клиентом. Согласно закону компания, предоставляющая услуги электронного автострахования, должна отменить транзакцию, после чего полис отменяется в системах РСА.

Ведущие аналитики в сфере электронных полисов считают, что для того, чтобы сократить число мошенничеств в данной сфере, нужно сократить количество продажи электронных полисов ОСАГО через посредников, до 3 экземпляров в месяц.

### **1.9 Данные для идентификации посредников**

При регистрации на сайтах автостраховщиков нужно вводить номер телефона, как указано на рисунке 4.

Выберите нужный вам вариант

У меня уже есть пароль    У меня ещё нет пароля

Ваш email  
 Заполните

Ваш номер мобильного телефона

Достоверное согласие на передачу персональных данных

Получить временный пароль

Пароль  
 Заполните

Рисунок 4 – Форма регистрации личного кабинета на сайте автострахования на примере “Росгосстрах”

Идентификация будет проводиться по номеру телефона и по адресу электронной почты. Были выделены именно эти данные так как при регистрации на сайтах страховщиков, посредники регистрируют полисы пользуясь услугами сервисов sms-активаторов и пользуются временной почтой. На рисунке 5 и 6 продемонстрирован примеры сайтов SMS-активатора и сайта регистрации временной почты.

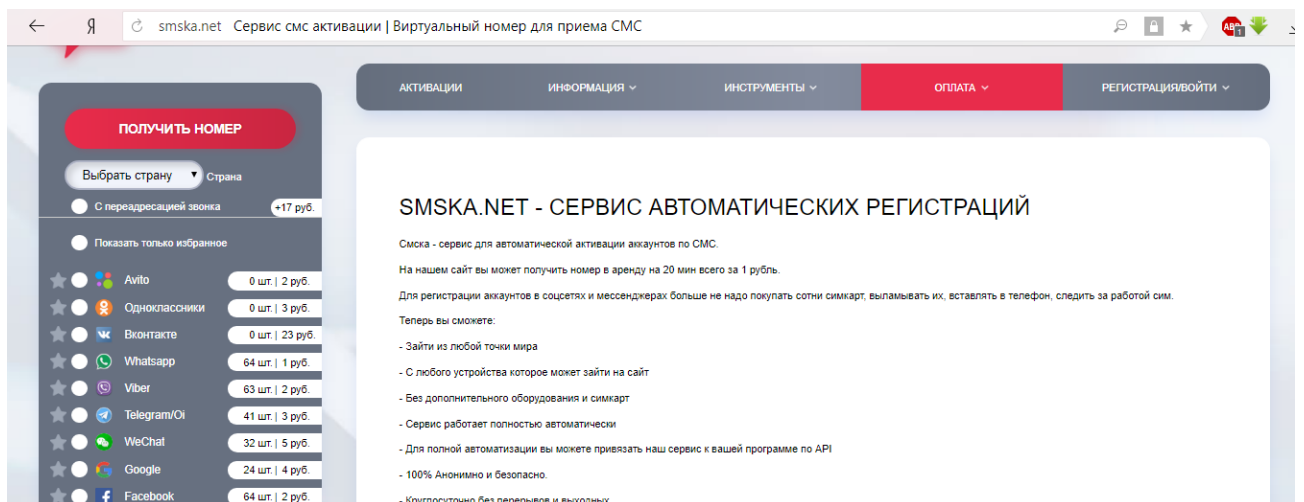


Рисунок 5 – Пример сайта SMS-активатора

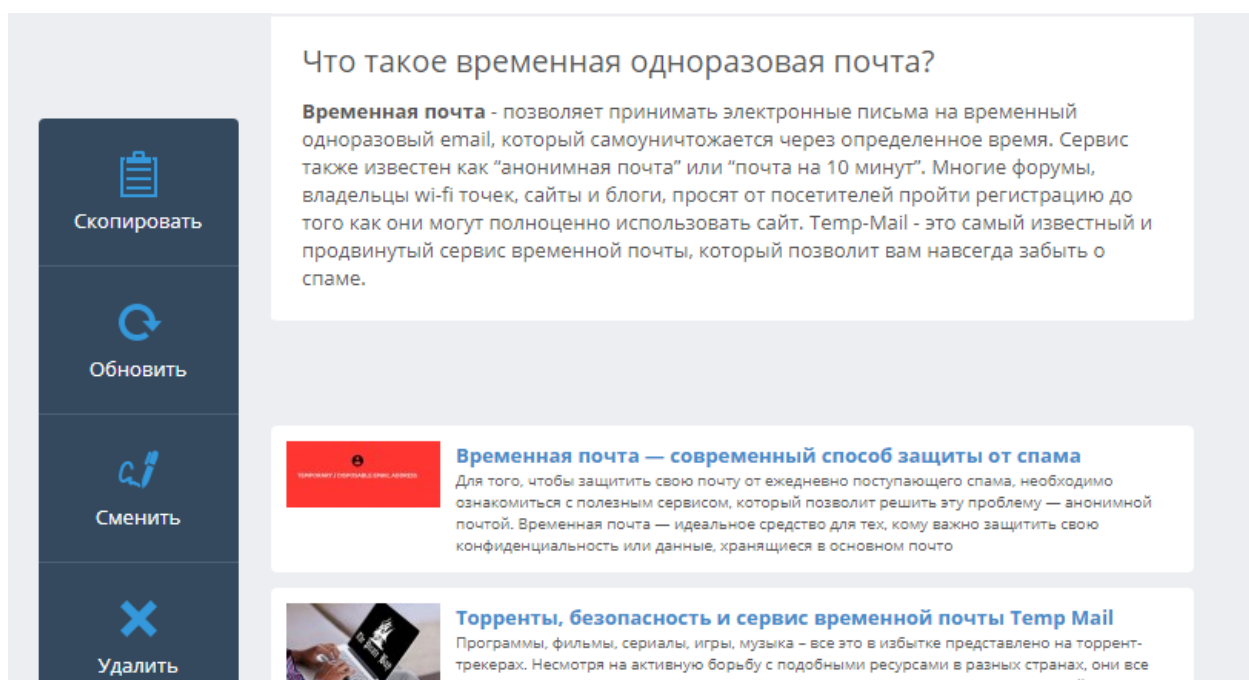


Рисунок 6 – Пример сайта регистрации временной почты

### Выводы по главе

В первой главе выпускной квалификационной работы были изучены: достоинства и недостатки электронных полисов ОСАГО. Изучены свойства.

Проанализирована проблема посредников в ОСАГО.

Выведен оптимальный метод борьбы с посредников в данной сфере.

Анализ показал, что существует большое количество сервисов, которые направлены на развитие электронных полисов ОСАГО, но ИТ-решений для работы с посредниками на рынке не существует, поэтому принято решение о разработке нового веб-приложения для идентификации посредников при продаже электронных полисов ОСАГО.

Так же были выделены данные, по которым нужно проводить идентификацию.

## Глава 2. РАЗРАБОТКА КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

### 2.1 Формирование требований к веб-приложению

Требование — это "условие или возможность, которым должна соответствовать система".

Существуют много разновидностей требований. Одна из классификаций требований называется моделью FURPS+ [GRA92], где FURPS - первые буквы названий категорий требований на английском языке.

- functionality (функциональные требования);
- usability (требования к удобству работы);
- reliability (требования к надежности);
- performance (требования к производительности);
- supportability (требования к простоте поддержки).

Знак плюса "+" в аббревиатуре FURPS+ охватывает дополнительные категории требований:

- ограничения на структуру проекта;
- требования к реализации;
- требования к интерфейсу;
- физические требования.

Функциональные требования относятся к следующим областям:

- набор функций;
- возможности;
- защита.

Требования к удобству работы охватывают следующие вопросы:

- человеческий фактор;
- эстетика;
- последовательность пользовательского интерфейса;
- электронная и контекстная справочные системы;

- мастера и агенты;
- пользовательская документация;
- учебные материалы.

Требования к надежности охватывают следующие вопросы:

- частота и серьезность сбоев;
- возможность восстановления;
- предсказуемость;
- точность;
- средняя продолжительность бесперебойной работы (MTBF).

Требования к производительности накладывают определенные условия на функциональные требования. Например, для определенной операции можно задать следующие требования к производительности:

- скорость выполнения;
- эффективность;
- коэффициент готовности;
- точность;
- пропускная способность;
- время отклика;
- время восстановления;
- используемые ресурсы.

Требования к удобству поддержки охватывают следующие вопросы:

- простота тестирования;
- простота расширения;
- простота адаптации;
- простота обслуживания;
- совместимость;
- простота настройки;
- простота обслуживания;

- простота локализации.

Требования к структуре проекта охватывают вопросы, относящиеся к структуре системы, и их часто называют ограничениями на структуру модели.

Требования к реализации относятся к особенностям программирования и конструирования системы. Например:

- соответствие стандартам;
- языки реализации;
- правила в отношении целостности баз данных;
- ограничения на ресурсы;
- рабочие среды.

Требования к интерфейсу охватывают следующие вопросы:

- внешние объекты, с которыми должна взаимодействовать система
- ограничения на форматы, время ожидания и другие обстоятельства

взаимодействия

## **2.2 Разработка алгоритма работы веб-приложения**

Логика данного веб-приложения очень проста, пользователь вводит данные в соответствующую форму и отправляет запрос. Программа проводит проверку в БД на определенные критерии и выдаёт результат. Результатом проверки приложения, в зависимости от результата проверки каждого критерия, может быть 2. Один положительный другой отрицательный.

Список критериев проверки:

- наличие в БД. Приложение будет проверять существует ли точно такой же элемент в базе данных страховой компании;
- наличие в БД похожих элементов. Похожими элементами могут считаться элементы, которые отличаются от исходных данных последним (для номера телефона) или последними (для email);

- наличие существующих полисов, зарегистрированные на эти или на похожие данные за последний месяц;

На рисунке 7 показано наглядное представление алгоритма работы веб-приложения

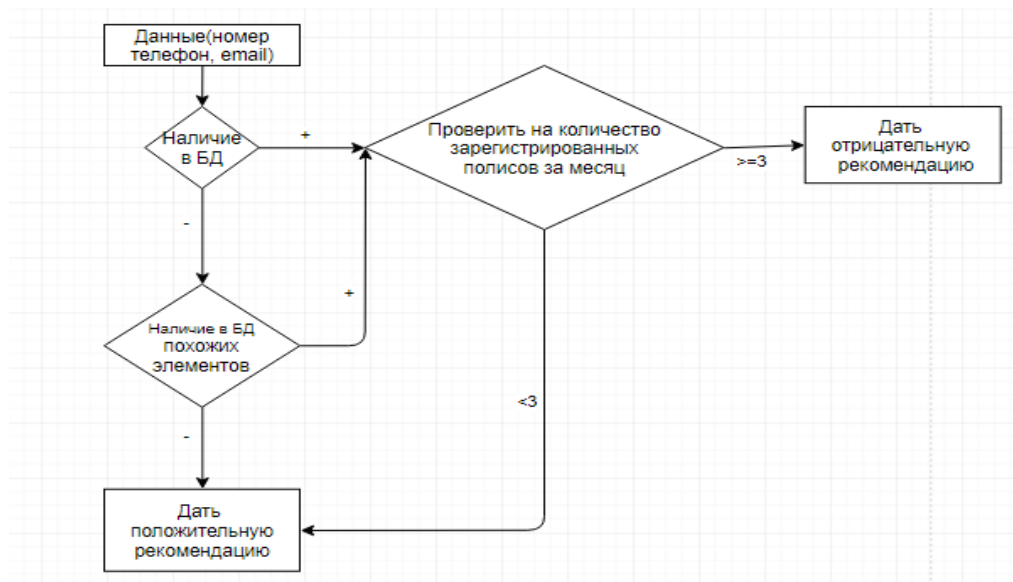


Рисунок 7 – Блок-схема алгоритма работы приложения

### 2.3 Диаграмма вариантов использования

Диаграммы вариантов использования описывают взаимоотношения между вариантами использования и действующими лицами.

Диаграммы вариантов использования для упрощения взаимодействия с пользователями системы и клиентами системы, а также для определения характеристик системы.

Вариант использования (use case) описывает группу действий в системе. Варианты использования – это описание взаимодействий между системой и ее пользователем.

Варианты использования отображают интерфейс системы и указывают форму системы, что та должна сделать.

Варианты использования должны выполнять следующее:

- вариант использования принадлежит, хотя бы к одному действующему лицу;
- вариант использования приводит к результату;
- у каждого варианта есть инициатор.

Варианты использования идентифицируются исходя из следующих соображений: каждый вариант использования представляет собой некоторую функцию, выполняемую системой в ответ на воздействие действующего лица (актера), и характеризует конкретный способ применения системы, диалог между актером и системой.

Действующее лицо (actor) - источник, взаимодействующий с системой при помощи варианта использования. Действующие лица могут являться пользователями системы или компьютерной системой.

На рисунке 8 продемонстрирована диаграмма вариантов использования приложения.

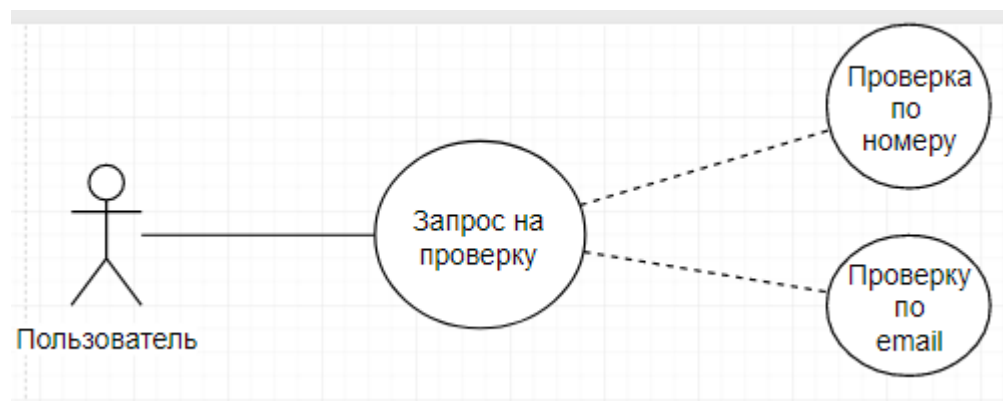


Рисунок 8 – Диаграмма вариантов использования идентификации посредника

## 2.4 Диаграмма последовательности

Диаграмма последовательности отражает поток событий.



В верхней части диаграммы показаны объекты. Вертикальные линии отображают течение времени, прямоугольники, показывают, то, что выполняет объект и стрелки, которые показывают взаимосвязь объектов.

На рисунке 9 показана диаграмма последовательности выполнения.

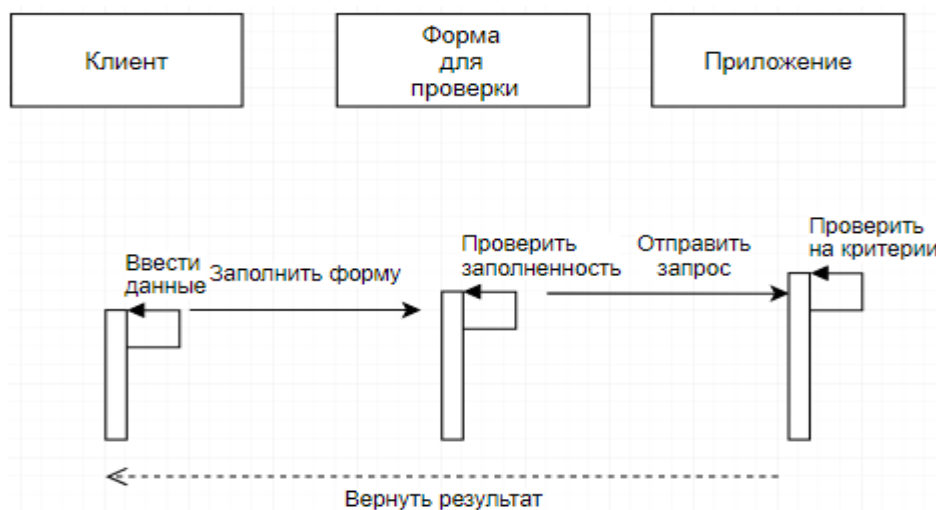


Рисунок 9 - Диаграмма последовательности выполнения сценария идентификации посредника

## 2.5 Алгоритм работы проверяющей функции, проводящей проверку

Посредники при регистрации используют ложные данные, которые были описаны в предыдущей главе. Веб-приложение, которое описывается в данной бакалаврской работе будет проверять именно эти данные. Первое, что будет проверять приложение — это наличие точно таких же номеров телефонов или адресов электронных почт, если такие есть, приложение будет проверять их количество. Если количество зарегистрированных электронных полисов за последний месяц больше двух, то приложение даст отрицательную рекомендацию на продажу электронного полиса ОСАГО такому пользователю, а если количество зарегистрированных полисов за последний месяц меньше 3, то приложение даст положительную рекомендацию.

Если таких данных нет, то будет проверять на наличие номеров телефонов, которые отличаются на 1 знак и почту, которая отличается от

введенной на 1-3 знака, так как при регистрации с временной почты и смс активатора, чаще всего попадают номера, идущие друг за другом, а почта отличается на несколько знаков. Если есть пользователи с такими данными, то приложение будет проверять количество зарегистрированных электронных полисов за последний месяц больше двух, то приложение даст отрицательную рекомендацию на продажу электронного полиса ОСАГО такому пользователю, а если количество зарегистрированных полисов за последний месяц меньше 3, то приложение даст положительную рекомендацию.

#### **Выводы по главе**

Во второй главе был разработан алгоритм работы веб-приложения.

Были построены диаграмма вариантов использования и диаграмма последовательности.

Так же был разработан и описан алгоритм работы приложения.

# Глава 3. РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ ИДЕНТИФИКАЦИИ ПОСРЕДНИКОВ ПРИ ПРОДАЖЕ ЭЛЕКТРОННЫХ ПОЛИСОВ ОСАГО

## 3.1 Выбор средств реализации

Сравнительная характеристика языков программирования: PHP, Ruby, Python.

Первый критерий - общая доля рынка.

Статистика использования и общая доля рынка любого языка программирования даёт представление о популярности этого языка среди масс. На рисунке 10 показывает наиболее широко используемые серверные языки программирования в мире по версии форума <https://w3techs.com/>. Результат исследования был опубликован 24 января 2018.

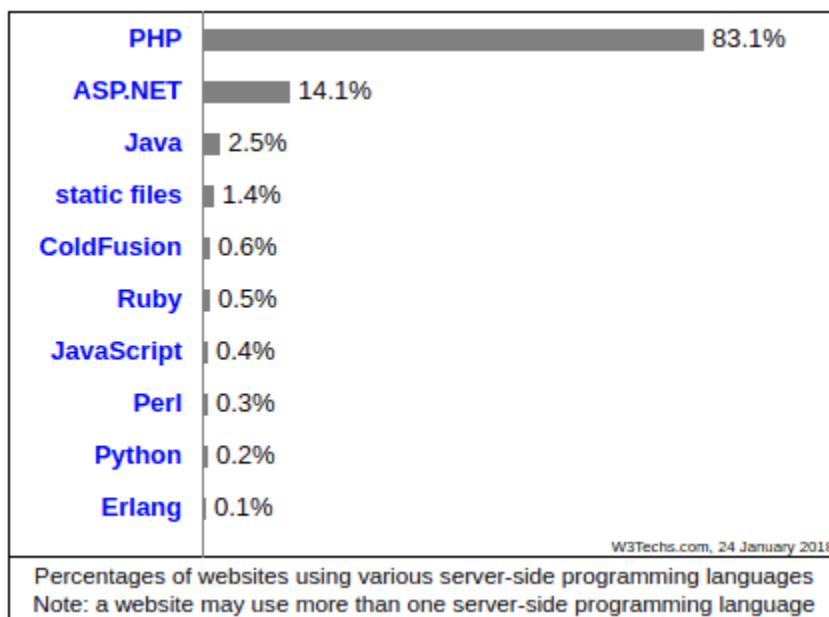


Рисунок 10 - Статистика доли рынка языков по версии форума

Рассматривая статистику можно сделать вывод, что PHP является наиболее широко используемым языком программирования в среде веб-разработки.

2. Популярные сайты, разработанные с использованием этих языков программирования.

Крупные проекты, при разработке которых использовался язык программирования PHP

- Wikipedia;
- Facebook;

Крупные проекты, при разработке которых использовался язык программирования Ruby:

- Twitter;
- Hulu;

Крупные проекты, при разработке которых использовался язык программирования Python:

- Google;
- YouTube.

По данному критерию нельзя отдать предпочтение одному из данных языков программирования.

### 3. Юзабилити

Юзабилити - один из наиболее важных факторов, которые учитываются перед тем, как выбрать язык программирования для разработки приложения. Почти все языки программирования образуют интерфейс между компьютером и пользователем. Язык с высоким юзабилити поможет разработать более мощное, масштабируемое и высокопроизводительное программное обеспечение. На рисунке 11 показана сравнительная таблица на основе критерия удобства использования языков PHP, Python и Ruby. Данные взяты с форума [habr.com](http://habr.com)

ЯЗЫК	ОЦЕНКА КРИТЕРИЯ
PHP	7/16
Ruby	13/16
Python	11/16

Рисунок 11 - Сравнительная таблица удобства использования

Изучив данную таблицу можно сделать вывод что язык программирования Ruby является лучшим по данному критерию

#### 4. Простота обучения

Простота обучения или “кривая обучаемости” также является важным параметром для выбора языка программирования для разработки проекта веб-приложений. Язык программирования с высоким значением кривой обучения прост в понимании и обладает довольно чистыми кодами. Среди этих трех языков Python имеет лучшую обучающую кривую, PHP - вторая и последняя - Ruby. На рисунке 12 показана сравнительная таблица языков по критерию простота обучения. Данные взяты с форума [habr.com](http://habr.com).

ЯЗЫК	ОЦЕНКА КРИТЕРИЯ
PHP	10/16
Ruby	6/16
Python	16/16

Рисунок 12 - Сравнительная таблица языков по критерию простота обучения по данным форума [habr.com](http://habr.com)

#### 5. Популярность

Несомненно, PHP является одним из самых старых языков. Он собрал много верных кодеров. Несмотря на то, что Python является относительно новым языком программирования, он успел набрать огромную базу программистов. На рисунке 13 изображен индекс ТЮВЕ, который измеряет

уровень популярности различных языков, созданных и поддерживаемых компанией TIOBE, которая располагается в Нидерландах.

May 2018	May 2017	Change	Programming Language	Ratings	Change
4	4		Python	5.192%	+1.64%
7	9	▲	PHP	3.321%	+0.63%
10	11	▲	Ruby	1.182%	-1.25%

Рисунок 13 - Индекс TIOBE

Чтобы получить четкое представление о веб-разработке на PHP, Python или Ruby, стоит разобраться в преимуществах и недостатках этих трех серверных языков программирования. У каждого языка есть и плюсы, и минусы. Все зависит от потребностей и ожиданий от этих языков. Можно выбрать язык, который наилучшим образом соответствует вашим конкретным бизнес-потребностям и требованиям. Вот плюсы и минусы этих трех языков программирования.

Плюсы PHP:

- бесплатное программное обеспечение по лицензии PHP;
- лёгкий в освоении (высокая скорость обучения);
- крупное сообщество пользователей и разработчиков;
- расширенная поддержка баз данных;
- предоставляет большое количество доступных расширений и исходных кодов;
- разрешает исполнение кода в ограниченных средах выполнения;
- предоставляется возможность управления нативными сессиями и расширения API;
- хорошая альтернатива конкурентам, таким как ASP (Active Server Pages) от Microsoft;
- Работает практически на любой операционной системе или платформе.

Минусы PHP:

- не подходит для разработки настольных приложений;
- традиционно скудный функционал для обработки ошибок;
- глобальные параметры конфигурации могут менять семантику языка, усложняя процессы внедрения и совместимости;
- обращение к объектам по умолчанию осуществляется методом «вызова по значению» (CallByValue), что противоречит аналогичным операциям для большинства языков и застаёт многих программистов врасплох;
- в целом считается менее защищённым по сравнению с другими языками программирования.

#### Плюсы Ruby:

- кроссплатформенность и открытый исходный код;
- может быть встроен в язык разметки гипертекста;
- язык программирования сверхвысокого уровня (VHLL) ;
- простой и понятный синтаксис, позволяющий начинающему разработчику очень быстро изучить язык;
- простое подключение к базам данных DB2, MySQL, Oracle и Sybase;
- созданные на Ruby большие масштабируемые программы просты в сопровождении;
- наличие встроенного отладчика и гибкого синтаксиса;
- возможность написания внешних библиотек на Ruby или C;
- возможность написания многопоточных приложений с простым API.

#### Минусы Ruby:

- возможны трудности в изучении;
- нехватка информационных ресурсов;
- большие затраты процессорного времени (CPU time) по сравнению с другими языками;

- сравнительно медленная разработка обновлений.

Плюсы Python:

- Лёгкий и быстрый в изучении;
- Поддерживается множеством платформ и операционных систем;
- Читабельный и организованный синтаксис;
- Обеспечение быстрого прототипирования и динамических семантических свойств;
- Огромное сообщество поддержки;
- Простое построение приложений путём тестирования и импорта необходимых функций;
- Реюзабилити (возможность повторного использования) за счёт тщательной разработки пакетов и модулей;
- Объектно-ориентированный подход к программированию.

Минусы Python:

- недостаточно эффективная работа с многоядерными и многопроцессорными вычислительными системами;
- ограниченный уровень доступа к базам данных;
- отсутствие коммерческой поддержки даже для Open Source проектов (однако, эта ситуация начинает меняться);
- небольшое количество разработчиков Python по сравнению с другими языками.

PHP – лучший язык для создания динамических веб-страниц;

Python – универсальный язык программирования, при помощи которого можно делать любые приложения в диапазоне от интернет-сайтов и десктопных приложений до роботов и системных сервисов;

Ruby – наиболее высокоуровневый язык, позволяющий уделять меньше внимания деталям интерфейса и организации хранения данных, чтобы сосредоточиться на прикладной задаче.



В результате обзора технологий создания веб-приложений был выбран язык PHP

### 3.2 Выбор языков разметки

HTML (HyperText Markup Language)— язык разметки (маркировки) гипертекста. Благодаря интернету развивается гипертекст, но создавался он для других целей. HTML позволяет переходить от одной части текста к другой, так же разные части могут храниться на различных компьютерах. HTML не является языком программирования т.к, его создали только для разметки web-страниц. Данный язык разметки позволяет показывать различные элементы страницы пользователям, так как хочет это отобразить разработчик. Необходимо помнить, что не только разные браузеры, но и разные версии могут по-разному воспринимать и выводить на экране код. Следовательно, элементы, корректно выглядящие в браузере Opera могут выглядеть иначе в Internet Explorer и других браузерах. При написании кода HTML-страниц необходимо проверять, как отобразился код различных браузерах, например, Mozilla Firefox, Internet Explorer и Opera. Код разметки в HTML состоит из так называемых «тэгов». Тэги отправляют информацию браузерам о том, как должны выглядеть и располагаться элементы на странице. Все тэги заключается в угловые скобки «<» и «>». Все тэги делятся на парные и на не парные, например, тег **b**, который делает текст жирным является парным тэгом. `<b>Жирный</b>` - заключенный между `<b>` и `</b>` будет выделен жирным. Эти тэги определяют, как будет выглядеть содержимое, находящегося в нем. Данные, которые находятся между такими символами, называются содержимым тэга. Примером не парного тэга является тэг `hr` — который чертит линию. Для того, чтобы работать с HTML, его необходимо лишь знать. Из-за большого количества разработчиков было создано большое количество редакторов, упрощающих работу с кодом. Список более популярных:

- Sublime Text;

- FrontPage;
- HomeSite;
- DreamWeaver.

#### Каскадные таблицы стилей CSS

CSS-стили устроены по обычному правилу. Необходимо выбрать компонент и написать к нему те свойства, которые необходимо применить. Стили можно применить как к глобальным элементам, например, body так и к отдельным элементам. Использовать каскадные таблицы можно:

- доступ через документ (можно описать стиль в самом тэге);
- внутри документа (стили можно записать тэге <head> внутри тэга <style>);
- отдельно (создать отдельно файл CSS и при помощи команды <link> прикрепить CSS к HTML документу).

### 3.3 Выбор СУБД

#### Сравнительный анализ MongoDB и MySQL

MySQL - полнофункциональная система управления реляционными базами данных с открытым исходным кодом (RDBMS), которая была первоначально построена MySQL AB и в настоящее время принадлежит Oracle Corporation. Он хранит данные в таблицах, которые сгруппированы в базу данных, использует “язык структурированных запросов” (SQL) для доступа к данным и некоторым командам, например «SELECT», «UPDATE», «INSERT» и «DELETE» для управления. Связанная информация может храниться в разных таблицах, но использование операции JOIN позволяет вам объединить ее, выполнять запросы используя данные из разных таблиц и минимизировать вероятность дублирования данных.

MySQL совместим практически со всеми операционными системами: Windows, Linux, Unix, Apple, FreeBSD и многими другими. Он поддерживает

различные механизмы хранения, такие как InnoDB (это по умолчанию), Federated, MyISAM, Memory, CSV, Archive, Blackhole и Merge.

MongoDB - популярная база данных с открытым исходным кодом, разработанная 10gen, позже называемая MongoDB Inc. В этом случае документы создаются и хранятся в файлах BSON, формате Binary JSON (JavaScript Object Notation), поэтому все типы данных JS поддерживаются. В этом случае MongoDB часто применяется для проектов Node.js. Кроме того, JSON позволяет передавать данные между серверами и веб-приложениями с использованием удобным для чтения формата.

Одним из лучших преимуществ, предлагаемых MongoDB, является использование динамических схем, которые устраняют необходимость предварительного определения структуры, например полей или типов значений. Такая модель позволяет представление иерархических отношений, хранение массива и способность изменять структуру записей, просто добавляя или удаляя поля. Это решение NoSQL поставляется с внедрением, автоматической настройкой и встроенной репликацией для лучшей масштабируемости и высокой доступности. Для удобства представим характеристики СУБД MySQL и MongoDB в таблице 3.1.

Таблица 3.1 – Сравнение характеристик СУБД

	<b>MySQL</b>	<b>MongoDB</b>
Языки на которых написана система	C++, C	C++, C и JS
Тип	полнофункциональная система управления реляционными базами данных с открытым исходным кодом (RDBMS)	Документо-ориентированный
Основные инструменты	Таблица, Ряд, Колонка	Коллекция, Документ, Поле

Продолжение таблицы 3.1

Структура	Строгая	Динамическая
Масштабирование	Вертикальное	Горизонтальное
Гибкость	-	+
Возможность управлять структурой	+	+
Драйверы на собственном языке	-	+
Ключевая особенность	полнотекстовый поиск и индексация; интегрированная поддержка репликации; триггеры; подзапросы; кэширование запросов; поддержка SSL; поддержка Unicode.	авто-фрагментация; родная репликация; быстрая память; поддержка встроенных моделей данных; комплексные вторичные индексы; богатая поддержка языка запросов; поддержка различных систем хранения данных
Открытый исходный код	+	+
ACID требования	+	+
Горизонтальное масштабирование с контролем местоположения данных	+	+

## Плюсы MySQL

- поддержка атомарных транзакций;
- поддержка JOIN;
- хорошо развитые решения;
- быстродействие;
- безопасность;
- надежность;
- переносимость;
- масштабируемость;
- сообщество;
- привилегия и система безопасности паролей.

## Минусы MySQL

- строгое масштабирование;
- отсутствие транзакций;
- медленная разработка;
- устойчивая структура;

## Плюсы MongoDB

- документно-ориентированное хранение данных;
- интегрированные системы хранения данных;
- динамические запросы;
- поддержка индексов;
- эффективное хранение двоичных данных больших объемов, например фото и видео;
- поддержка отказоустойчивости и масштабируемости;
- Имеет распределенный доступ к данным расположенных на нескольких серверах;
- сокращенное время между первичным отказом и восстановлением.

## Минусы MongoDB

- не лучший вариант для приложений со сложными транзакциями;
- отсутствие оператора JOIN;
- отсутствие транзакции;
- отсутствие изоляции;
- требовательна к ресурсам;
- не замена для устаревших решений;
- не стабильна;
- требовательна к ресурсам памяти.

Сравнение производительности MongoDB и MySQL затруднено, поскольку обе системы управления чрезвычайно производительны, а основные различия лежат в основе их операций.

Оба устройства с открытым исходным кодом и легко доступны, а также обе системы предлагают коммерческие версии с множеством дополнительных функций.

Сравнивая скорость MongoDB и MySQL, разработчики отмечают, что у последнего нет скорости и существуют трудности при работе с большими объемами данных, поэтому он будет лучшим выбором для приложений использующих не большие объемы данных.

Так как MySQL и MongoDB по многим параметрам схожи, но MySQL доступнее и удобнее в использовании было принято решение использовать именно систему управления базами данных – MySQL.

### **3.4 Реализация интерфейса веб приложения**

На рисунке 14 представлен интерфейс формы для выполнения функции

**Добро пожаловать! Для проверки посредника введите данные**

Номер телефона Введите 11 значный номер телефона

email Введите почту

**ДАЛЕЕ**

Рисунок 14 – Интерфейс формы выполнения функции  
Результатов выполнения приложения может быть два

Первый ответ будет выдаваться в случаях, когда не обнаружено элементов с идентичными или похожими данными, или обнаружено, но количество зарегистрированных электронных полисов ОСАГО меньше трёх. На рисунке 15 продемонстрирован окно с ответом, который будет выдан в этом случае.

**Элементов с идентичными или похожими данными не обнаружено или обнаружено в допустимом количестве!**

Рисунок 15 – Окно с одним из двух вариантов ответа

Второй будет выдаваться в случаях, когда количество зарегистрированных полисов с идентичными или похожими данными превышает 2. На рисунке 16 продемонстрировано окно с ответом, который будет выдан в этом случае.

**Обнаружено подозрительное количество ранее зарегистрированных полисов**

Рисунок 16 - Окно со вторым вариантом ответа

В итоге страховые компании будут знать, посредник покупает у них электронный полис ОСАГО или всё-таки обычный клиент.

## **Выводы по главе**

В третьей главе был проведен сравнительный анализ языков программирования: PHP, Ruby и Python. Изучив некоторые критерии было принято решение, что для разработки данного веб-приложения оптимальным будет использование языка программирования PHP. Кроме того, были выбраны языки разметки, HTML и CSS. Для выбора СУБД был проведен сравнительный анализ MongoDB и MySQL, принимая во внимание, тот факт, что MySQL и MongoDB по многим параметрам схожи, но MySQL доступнее и удобнее в использовании было принято решение использовать именно эту СУБД. Реализован интерфейс веб-приложения.



## ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была разработана модель веб-приложения для идентификации посредников при продаже электронных полисов ОСАГО.

Были изучены: достоинства и недостатки электронных полисов ОСАГО и свойства. Проанализирован вопрос посредников в данной сфере. И выведен оптимальный метод борьбы с мошенничеством в сфере электронного страхования.

Анализ показал, что для работы с посредниками на рынке не существует ИТ-решений, поэтому принято решение о разработке нового веб-приложения для идентификации посредников при продаже электронных полисов ОСАГО.

Так же были выделены данные, по которым нужно проводить идентификацию.

Был разработан алгоритм работы веб-приложения. Были построены диаграмма вариантов использования и диаграмма последовательности. Так же был разработан и описан алгоритм работы функции.

Был проведен сравнительный анализ языков программирования: PHP, Ruby и Python. Изучив некоторые критерии было принято решение, что для разработки данного веб-приложения оптимальным будет использование языка программирования PHP. Кроме того, были выбраны языки разметки, HTML и CSS. Для выбора СУБД был проведен сравнительный анализ MongoDB и MySQL, принимая во внимание, тот факт, что MySQL и MongoDB по многим параметрам схожи, но MySQL доступнее и удобнее в использовании было принято решение использовать именно эту систему управления базами данных.

Реализовано веб-приложение.

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

### *Нормативно-правовые акты*

1. ГОСТ 34.320-96. Информационная технология. Система стандартов по базам данных. Концепции и терминология для концептуальной схемы и информационной базы.

### *Научная и методическая литература*

2. Золотов С. Ю. Проектирование информационных систем : учеб. пособие / С. Ю. Золотов ; Томский гос. ун-т систем управления и радиоэлектроники. - Томск : Эль Контент, 2013. - 86 с.

3. Информационные аналитические системы : учебник / Т. В. Алексева ; под ред. В. В. Дика. - Москва : Синергия, 2013. - 379 с.

4. Крахоткина Е.В. Методы и средства проектирования формационных систем и технологий : учебное пособие / Крахоткина Е.В. - Ставрополь: Северо-Кавказский федеральный университет, 2015. -152 с.

5. Булычева П.А. Разработка алгоритма работы веб-модуля для структурирования знаний студента по дисциплине, 2018. -130 с.

6. Бэнкер. К. MongoDB в действии, 2016. - 287 с.

7. Доусон М. Програмируем на Python. – СПб.: Питер, 2014. – 416 с.

8. Скляр. Д. Изучаем PHP 7. Руководство по созданию интерактивных веб-сайтов, 2017, -464 с.

9. Зандстра. М. PHP. Объекты, шаблоны и методики программирования, 2016, -528с

### *Электронные ресурсы*

10. Средства автоматизации моделирования UML. - URL: <http://citforum.ru/database/case/index.shtml>

11. Страховая компания «Росгосстрах». - URL: <https://www.rgs.ru>

12. Калькулятор электронного полиса ОСАГО: -URL: <https://osago.online/#services>

13. Сервис сравнения услуг поставщиков в различных сферах деятельности: -URL: <https://www.sravni.ru/about/>

14. Официальный сайт языка программирования Ruby: – URL: <https://www.ruby-lang.org/ru/>

15. Официальный сайт индекса TIOBE. – URL: <https://tiobe.com/tiobe-index/>

16. Официальный сайт языка программирования Python -URL: <https://www.python.org/>

17. Официальный сайт языка программирования PHP -URL: <http://php.net/>

18. Официальный сайт языка программирования MongoDB -URL: <https://www.mongodb.com/>

19. Официальный сайт языка программирования MongoDB -URL: <https://www.mysql.com/>

20. Ресурс для IT-специалистов, издаваемый компанией «ТМ» URL: <https://habr.com/>

*Литература на иностранном языке*

21. Nixon R. Learning PHP, MySQL, JavaScript, CSS & HTML5 - 3rd Edition, 2014.

22. Grinberg. M. Flask Web Development: Developing Web Applications with Python 2nd Edition, 2018, -316p

23. S. Haddad, F. Kordon, L. Pautet, L. Petrucci Models and Analysis in Distributed Systems, 2014. -368 p.

24. A. Zed Shaw. Learn Python the Hard Way 3rd Edition, 2014. -320p.

25. J. Valade, S. Suehring. PHP, MySQL, JavaScript & HTML5 All-in-One For Dummies 2013, -720p

## Приложение А. Фрагмент исходного кода проверяющей функции

```
<?php
if (isset($_POST['number'])) { $number = $_POST['number']; if ($number == "")
{ unset($number);} }
    if (isset($_POST['email'])) { $email=$_POST['email']; if ($email == "")
{ unset($email);} }
    $number = stripslashes($number);
    $number = htmlspecialchars($number);
    $email = stripslashes($email);
    $email = htmlspecialchars($email);
    include ("bd.php");
    $result = mysql_query("SELECT * FROM users WHERE
number='$number'", $db);
    $myrow = mysql_fetch_array($result);
    $result1 = mysql_query("SELECT * FROM users WHERE email='$email'", $db);
    $myrow1 = mysql_fetch_array($result1);
    if ((isset($myrow['number'])) and (isset($myrow1['email'])))
    {
        if ($myrow['epolicy']==1)
            {
                echo ("Обнаружен number and email элемент");
                echo "<BR>";
                echo '<td>' . $myrow['number'] . '</td>';
                echo "<BR>";
                echo '<td>' . $myrow['email'] . '</td>';
                echo "<BR>";
                echo '<td>' . $myrow['name'] . '</td>';
                echo "<BR>";
                echo '<td>' . $myrow['epolicy'] . '</td>';
```

```

echo "<BR>";
echo '</tr>';
    }
}
elseif (((isset($myrow1['email'])) and ($myrow1['epolicy']==1)))
{
echo ("Обнаружен email элемент");
echo "<BR>";
echo '<td>' . $myrow1['number'] . '</td>';
echo "<BR>";
echo '<td>' . $myrow1['email'] . '</td>';
echo "<BR>";
echo '<td>' . $myrow1['name'] . '</td>';
echo "<BR>";
echo '<td>' . $myrow1['epolicy'] . '</td>';
echo "<BR>";
echo '</tr>';
}
elseif ((isset($myrow['number'])) and ($myrow['epolicy']==1))
{

echo ("Обнаружен number элемент");
echo "<BR>";
echo '<td>' . $myrow['number'] . '</td>';
echo "<BR>";
echo '<td>' . $myrow['email'] . '</td>';
echo "<BR>";
echo '<td>' . $myrow['name'] . '</td>';
echo "<BR>";
}

```

```

echo '<td>' . $myrow['epolicy'] . '</td>';
echo "<BR>";
echo '</tr>';
}
elseif ((empty($myrow['number'])) and (empty($myrow1['email'])))
{
    echo "Нехватает данных";
}
else {
for ($i=0; $i++<9;){
    $newnumber3[] = $newnumber1.$newnumber2;
    $newresult = mysql_query ("SELECT * FROM users WHERE
number='$newnumber3'", $db);
    $newmyrow = mysql_fetch_array($newresult);
    echo "Обнаружен для ".$newnumber3." элемент";
    echo "<BR>";
    echo '<td>' . $newmyrow2['number'] . '</td>';
    echo "<BR>";
    echo '<td>' . $newmyrow2['email'] . '</td>';
    echo "<BR>";
    echo '<td>' . $newmyrow2['name'] . '</td>';
    echo "<BR>";
    echo '<td>' . $newmyrow['epolicy'] . '</td>';
    echo "<BR>";2
    echo '</tr>';
        $i++;";
    }
    for ($newemail2=0; $newemail2++<9;){
    $newemail3[] = $newemail.$newemail2;

```

```
$newresult1 = mysql_query ("SELECT * FROM users WHERE  
email='$newemail3'", $db); //  
  
$newmyrow3 = mysql_fetch_array($newresult3);  
echo "Обнаружен для ".$newemail3." элемент";  
echo "<BR>";  
echo '<td>'. $newmyrow3['number'] . '</td>';  
echo "<BR>";  
echo '<td>'. $newmyrow3['email'] . '</td>';  
echo "<BR>";  
echo '<td>'. $newmyrow3['name'] . '</td>';  
echo "<BR>";  
echo '<td>'. $newmyrow3['epolicy'] . '</td>';  
echo "<BR>";  
echo '</tr>';
```

## Приложение Б. Фрагмент кода реализации формы для проверки

```
<html>
<head>
<title>Главная страница</title>
</head>
<body>
<h2>Главная страница</h2>
<form action="Check.php" method="post">
<p>
<label>Номер телефона<br></label>
<input name="number" type="text" size="15" maxlength="11" pattern="[0-9]{11}">
</p>
<p>
<label>email:<br></label>
<input name="email" type="text" size="15" >
</p>
<p>
<input type="submit" name="submit" value="Проверить">
</p>
<br>
</form>
</body>
</html>
```