Министерство науки и высшего образования РФ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рыбинский государственный авиационный технический университет

имени П.А. Соловьева»

Факультет радиоэлектроники и информатики

Кафедра МПО ЭВС

**выпускная квалификационная работа**

Разработка микрофронтенда для веб-приложений продуктов страховых компаний.

на соискание степени бакалавр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

по направлению 09.03.04 Программная инженерия\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

профиль бакалавриата: Разработка программно-информационных систем\_\_\_\_\_\_\_\_\_\_

**Пояснительная записка**

Соискатель, студент группы ИПБ-16 Рахимов К.С.

*(Код) (Подпись, дата ) (Фамилия И.О.)*

Руководитель Волков М.Л.

*(Уч. степень, звание) (Подпись, дата) (Фамилия И.О.)*

Консультант

по экономике к.т.н., доцент Клементьева Н.А.

*(Уч. степень, звание) (Подпись, дата) (Фамилия И.О.)*

Нормоконтролер ст.преп. Задорина Н.А.

*(Уч. степень, звание) (Подпись, дата) (Фамилия И.О.)*

К защите допустить

Зав. кафедрой к.ф.-м.н., профессор Паламарь И.Н.

*(Уч. степень, звание) (Подпись, дата) (Фамилия И.О.)*

Рыбинск 2020

Оглавление

[1. Анализ предметной области 4](#_Toc41517827)

[1.1. Постановка задачи 4](#_Toc41517828)

[1.2. Описание проблемы 4](#_Toc41517829)

[1.3. Обзор аналогов 6](#_Toc41517833)

[2. Проектная документация 8](#_Toc41517834)

[2.1. Техническое задание 8](#_Toc41517835)

[2.1.1. Назначение разработки 8](#_Toc41517836)

[2.1.1. Требования к программе или программному изделию 8](#_Toc41517837)

[2.1.2. Требования к документации на дипломный проект (работу) 20](#_Toc41517838)

[2.2. Пояснительная записка к программному обеспечению 21](#_Toc41517839)

[2.2.1. Анализ предметной области 21](#_Toc41517840)

[2.2.2. Обзор аналогов 21](#_Toc41517841)

[2.2.3. Цели создания системы 21](#_Toc41517842)

[2.2.4. Назначение системы 21](#_Toc41517843)

[2.2.5. Технические характеристики 22](#_Toc41517844)

[2.3. Программа и методика испытаний 28](#_Toc41517845)

[2.3.1. Объект испытаний 28](#_Toc41517846)

[2.3.2. Цель испытаний 28](#_Toc41517847)

[2.3.3. Требования к программе 28](#_Toc41517848)

[2.3.4. Методы испытаний 29](#_Toc41517849)

[3. Акт испытаний программного продукта 31](#_Toc41517850)

[3.1. Объект испытаний 31](#_Toc41517851)

[3.2. Цель испытаний 31](#_Toc41517852)

[3.3. Результат испытаний 31](#_Toc41517853)

[3.4. Выводы 32](#_Toc41517854)

[4. Экономическая часть 33](#_Toc41517855)

[4.1. Экономическое обоснование разрабатываемого программного обеспечения 33](#_Toc41517856)

[4.2. Описание функций, назначения и потенциальных пользователей программного обеспечения 33](#_Toc41517857)

[4.3. Расчет затрат на разработку программного обеспечения 33](#_Toc41517858)

[4.4. Оценка результата от продажи программного обеспечения 33](#_Toc41517859)

[4.5. Расчет показателей эффективности инвестиций в разработку программного обеспечения. 33](#_Toc41517860)

[Акт о внедрении 34](#_Toc41517861)

[Заключение 35](#_Toc41517862)

[Литература 36](#_Toc41517863)

[Приложение 37](#_Toc41517864)

[Приложение А 37](#_Toc41517865)

[Приложение Б 40](#_Toc41517866)

1. Анализ предметной области
   1. Постановка задачи

Ни для кого не секрет, что страховые компании имеют большой ряд различных страховых продуктов. Страховые продукты могут сильно различаться между собой. Получается, для разработки каждого продукта, необходима определённая команда разработчиков, которая разрабатывает компоненты, при помощи различных инструментов. В данный момент страховые компании, имеют одно веб-приложение, в котором большое количество различных продуктов. Для того чтобы веб-приложение не было перегружено, и разработчики продуктов не имели много зависимостей от общего веб-приложения, было принято решение упростить веб-приложение, разделив каждый продукт страховой компании на отдельное веб-приложение. Веб-приложение (фронтенд) необходимо разделить на несколько веб-приложений (микрофронтендов), тем самым организовать правильную микрофронтендную структуру всего проекта.

* 1. Описание проблемы

В ходе анализа поставленной задачи был выделен ряд проблем, описанных ниже:

* Одна команда разработчиков в рамках одного проекта (одного веб-приложения)
* Зависимость к определенному списку технологий
* Большая вероятность, что всё веб-приложение сломается из-за одной незначительной ошибки.
* Тяжело отслеживать и решать ошибки.

Для решения обозначенных проблем, были приняты следующие цели:

1. Разработка микрофронтендной архитектуры для страховой компании;
2. Синхронизация с уже существующей системой договоров;
3. Минимизация зависимостей микрофронтендов.
4. Связка веб-приложений (реализованных на различных инструментальных основах) в одно единое веб-приложение;
5. Децентрализованная центральная система, которая должна распределять каждое веб-приложение;
6. Упрощение работы для сотрудников компании;
7. Взаимодействие между веб-приложениями;
8. Маршрутизация, как и от центральной системы, так и от внутренних веб-приложений.

Для достижения поставленных целей можно выделить следующие задачи:

1. Проектирование системы в целом и отдельных её частей;
2. Выбрать основные инструменты и форматы представленных данных;
3. Спроектировать микрофронтендную архитектуру;
4. Спроектировать структуру представленных данных;
5. Разработать и реализовать микрофронтенд, который будет являться главным менеджером для всех микрофронтендов;
6. Разработать и реализовать хранилище данных о клиенте страховой компании продукта;
7. Разработать и реализовать микрофронтенд, в котором будет механизм авторизации пользователя в систему;
8. Разработать и реализовать хранилище договоров страхового продукта;
9. Разработать и реализовать микрофронтенд и серверное приложение, для манипуляции над хранилищем договоров;
10. Разработать и реализовать микрофронтенд, в котором будет выведен по поиску определённый договор из хранилища договоров;
11. Разработать и реализовать хранилище договоров страхового продукта.
12. 3. Обзор аналогов

Для объяснения целесообразности разработки данного ПО, проведем сравнение с уже существующими программными продуктами, присутствующими на рынке, имеющими схожий функционал.

Еще до того, как в 2016 году появился термин микрофронтенд, многие крупные компании пытались решать схожие проблемы:

* Facebook BigPipe;
* YouTube Iframes;
* Project Mosaic.

В закрытом доступе представлена только система BigPipe от компании Facebook.

Сравнительный анализ будет произведен на примере системы Iframes от компании YouTube:

Схожая микрофронтендная архитектура на основе Iframes используется в компании Youtube. Главными отличиями разрабатываемой архитектуры от данного аналога являются:

* Отсутствие главного менеджера для всех компонентов;
* Отсутствие так каковой маршрутизации;
* Неудобная загрузка одновременно всех компонентов;
* Отсутствие разграничения по правам пользователя.

Существенным отличием от разрабатываемой системы является то, что отсутствует возможность расположения целого микрофронтенда, как отдельное приложение.

Таблица 1. Сравнение аналогов по критериям.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Facebook BigPipe | YouTube Iframes | Project Mosaic | Разрабатываемая система |
| Децентрализованная маршрутизация | + | - | - | + |
| Ленивая загрузка компонентов | - | + | - | + |
| Менеджер компонентов | - | - | + | + |
| Загрузчик всех компонентов | + | - | + | + |
| Взаимодействие между компонентами | - | + | + | + |

По результатам данного сравнения стало известно, представленный аналог не реализует в себе все необходимые функции. Использование уже готовых программных решений приведет к тому, что многие части придется рефакторить.

В связи с этим появляется необходимость создания более функциональной системы, которая позволит страховой компании сократить свои трудозатраты и уменьшить количество выполняемых ошибок, возникающих из-за человеческого фактора.

1. Проектная документация
   1. Техническое задание
      1. Назначение разработки

Данная разработка предназначена для повышения уровня удобства работы внутри страховых компаний, упрощение работы разработчиков и уменьшение зависимостей. При помощи данной разработки появится возможность распределение страховых продуктов в раздельные хосты, тем самым каждый микрофронтенд будет расположен на определённой машине и по определённому пути.

* + 1. Требования к программе или программному изделию

Программное изделие должно быть написано с учетом особенностей текущей архитектуры модулей организации «БиАйВи».

* + - 1. Требования к модульной структуре

Web-приложение «Страхование B2B Согласие» должен содержать в себе следующее:

1. Микрофронтендную архитектуру;
2. Менеджер микрофронтендов;
3. Микрофронтенд авторизация;
4. Микрофронтенд меню;
5. Микрофронтенд создание контракта;
6. Микрофронтенд поиска контракта.
   * + 1. Требования к функциональным характеристикам

Разделение целого веб-приложения страховой компании «B2B Согласие» на микро веб-приложения, то есть микрофронтенды. Организовать связь между микрофронтендами. Разработка как на прикладном JavaScript, так и на современных JavaScript фреймворках (а именно Angular веб-фреймворк).

**Микрофронтенд авторизация.**

При первом запуске веб-приложения пользователь должен увидеть окно авторизации для входа в систему. Это окно должно содержать область ввода конфиденциальной информации (логин и пароль), для того чтобы можно было получить доступ к системе «B2B Согласие».

В качестве логина и пароля используются данные, полученные в системе администрирования систем. Микрофронтенд поддерживает только авторизацию, без регистрации новых пользователей. Прототип микрофронтенда авторизации изображен на рисунке 1.

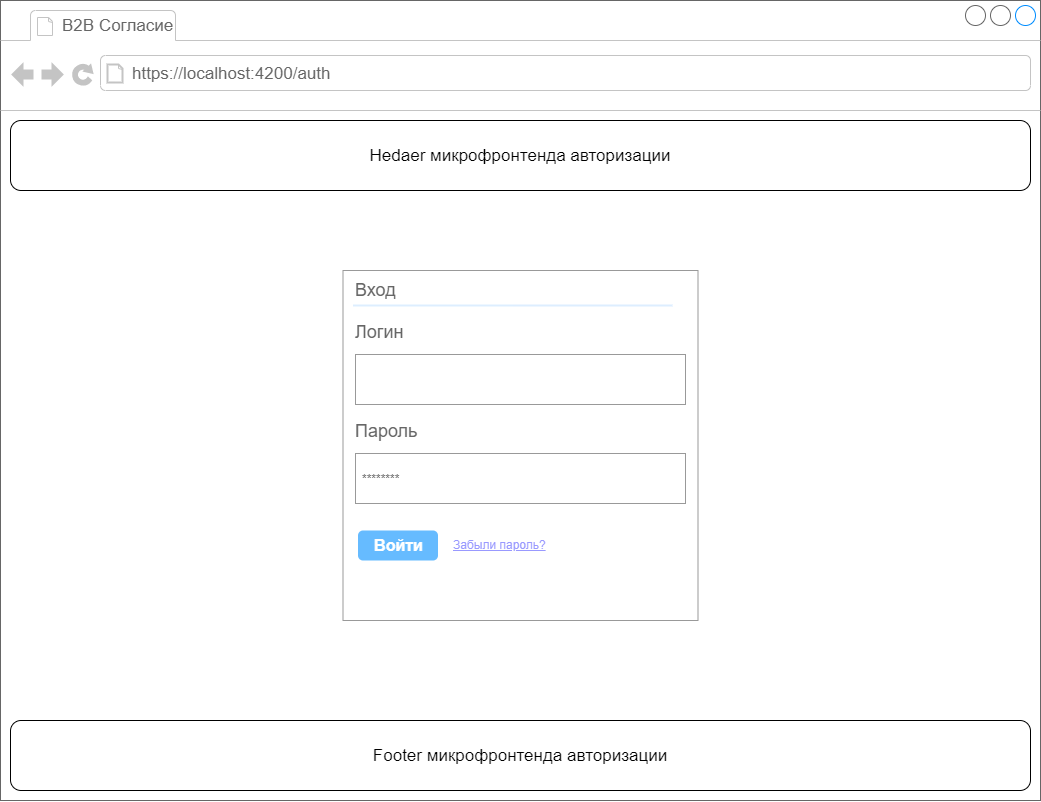


Рисунок 1. Прототип микрофронтенда авторизации.

В блоке, где обозначено «Header микрофронтенда авторизации», должен располагаться компонент с определённой информацией и логотипом «B2B Согласие».

В блоке, где обозначено «Footer микрофронтенда авторизации», должен располагаться компонент с определённой информацией и ссылкой.

**Микрофронтенд меню.**

После авторизации пользователь попадает на страницу меню, где расположены различные ссылки на сервисы.

Сервисы:

* Главная;
* Калькулятор;
* Договоры;
* Клиенты;
* Сервисы;
* Сервисы v2;
* Платежи;
* Журнал;
* БСО;
* Задачи.

Но также в данной странице необходимо вывести договоры (работа с портфелем договоров). Должны лежать ссылки на поиск договора и на оформление договора. Компания «B2B Согласие» имеет большое количество договоров, но для данной разработки можно обойтись лишь одним договором и это будет договор «Каско ФЛ v2».

Прототип микрофронтенда меню изображен на рисунке 2.

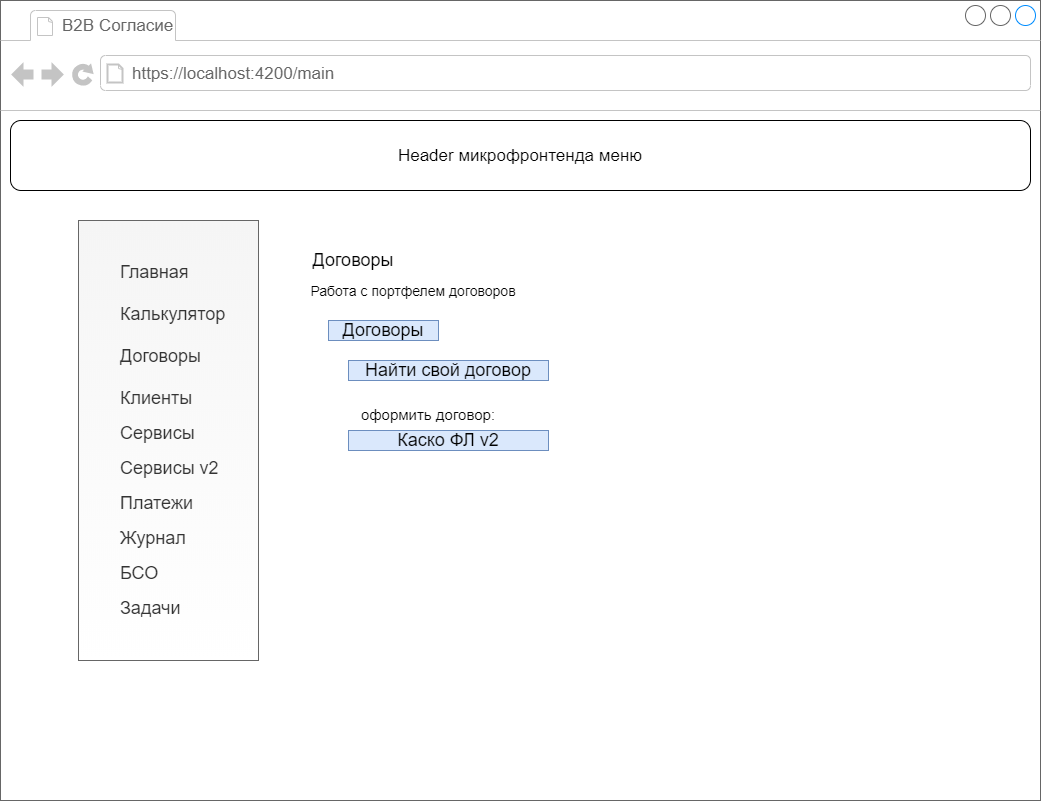


Рисунок 2. Прототип микрофронтенда меню.

В блоке, где обозначено «Header микрофронтенда авторизации», должен располагаться компонент с определённой информацией и логотипом «B2B Согласие».

Также в данном компоненте, должен располагаться логин пользователя, который вошел в систему и кнопка для выхода из системы.

Как только пользователь нажал на кнопку выхода из системы, то его должно перекинуть на страницу авторизации.

**Микрофронтенд создание контракта.**

После перехода из главного меню по нажатию на ссылку «Каско ФЛ v2», пользователь должен попасть на страницу создания контракта.

В данной странице пользователь, заполнив определённые поля ввода, может сохранить новый контракт. Соответственно новый контракт должен сохраниться в базе данных, для того чтобы в дальнейшем мы могли найти данный контракт.

Прототип микрофронтенда создание контракта изображен на рисунке 3.

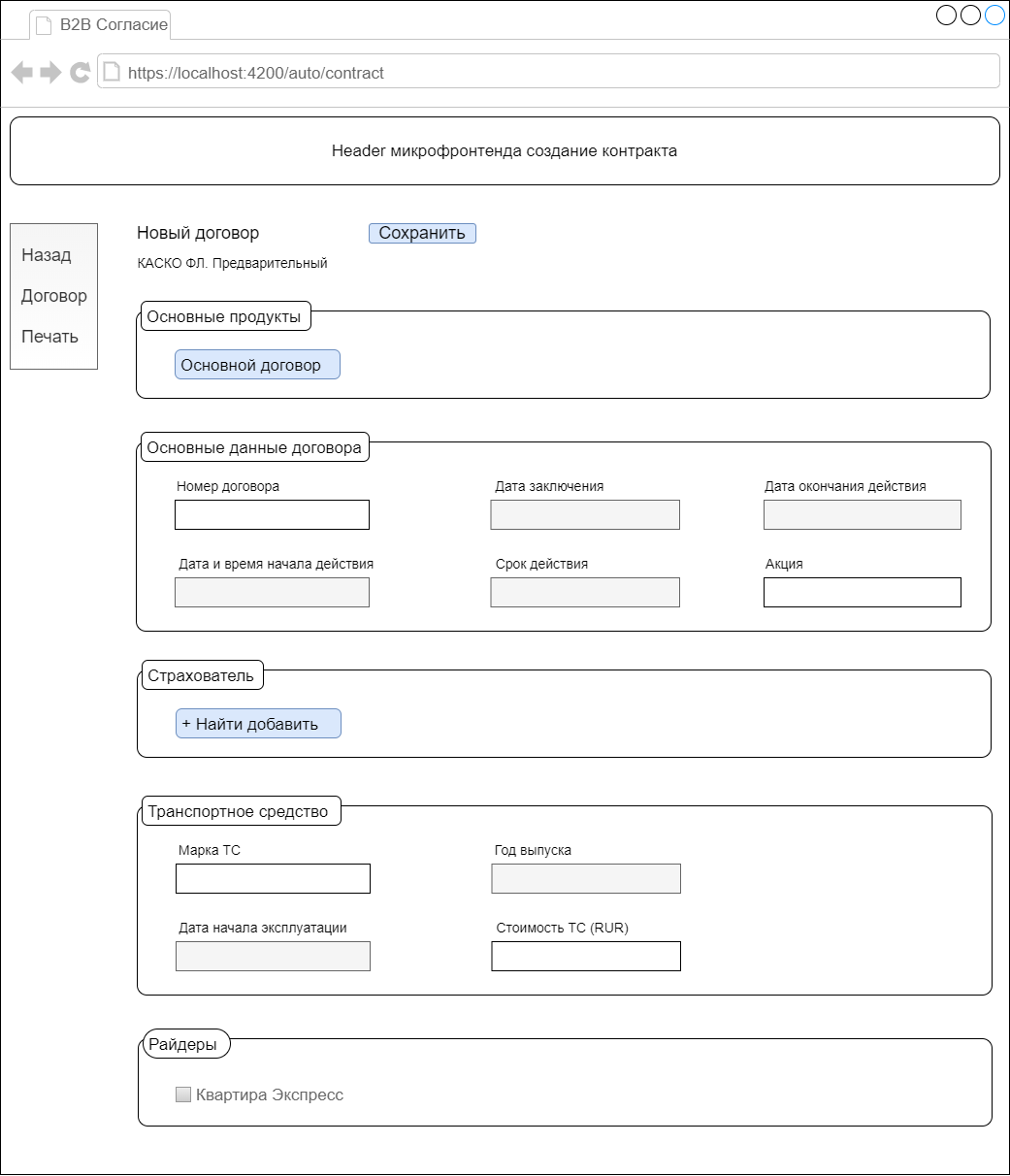


Рисунок 3. Прототип микрофронтенда создание контракта.

В блоке, где обозначено «Header микрофронтенда создание контракта», должен располагаться компонент с определённой информацией и логотипом «B2B Согласие».

Также в данном компоненте, должен располагаться логин пользователя, который вошел в систему и кнопка для выхода из системы.

Как только пользователь нажал на кнопку выхода из системы, то его должно перекинуть на страницу авторизации.

На странице - создание контракта, имеются 8 модулей:

* Сохранение контракта;
* Основные продукты;
* Основные данные договора;
* Поиск страхователя;
* Добавление страхователя;
* Транспортное средство;
* Райдеры;
* Райдер: Квартира Экспресс.

1. Модуль «сохранение контракта». После того как пользователь заполнил все поля, то он может нажать на кнопку «сохранить», тем самым контракт пользователя будет сохранен в базе данных. Модуль «сохранения контракта» изображен на рисунке 4.

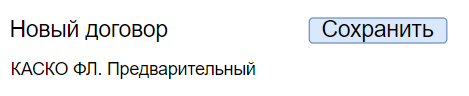


Рисунок 4. Прототип модуля сохранения контракта.

1. Модуль «основные продукты». В данном модуле пользователь может переходить из основного договора в райдеры и наоборот. Модуль «основные продукты» изображен на рисунке 5.

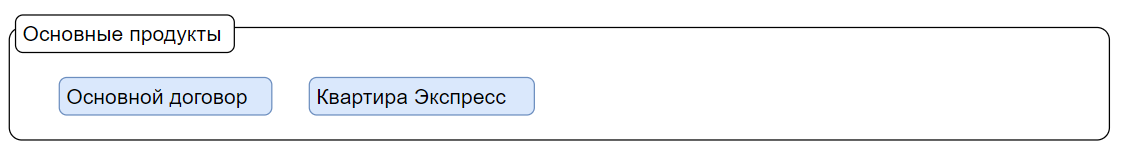


Рисунок 5. Прототип модуля основные продукты.

Отображения кнопки «Квартира Экспресс», зависит от нажатого чек-бокса в модуле «Райдеры», если чек-бокс активен, то кнопка «Квартира Экспресс», отобразиться в модуле «основные продукты».

1. Модуль «основные данные договора». В данном модуле расположены шесть полей ввода, которые должны быть заполнены пользователем, после чего эти данные будут направлены в базу данных, когда пользователь сохранит договор. Серым выделены поля, которые невозможно заполнить, так как для данной разработки нет необходимости в них. Модуль «основные данные договора» изображен на рисунке 6.

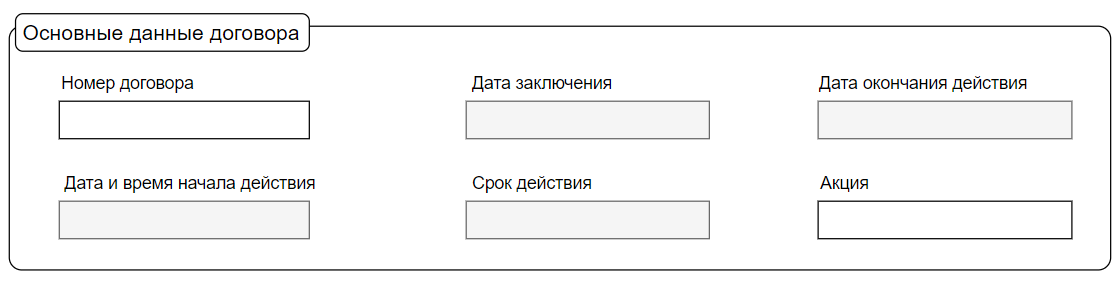


Рисунок 6. Прототип модуля основные продукты.

1. Модуль «поиск страхователя». Данный модуль имеет два подмодуля, которые имеют различные диалоговые окна – «найти страхователя» и «создание физ. лица». При нажатии появляется сначала диалоговое окно «найти страхователя». Модуль «поиск страхователя» изображен на рисунке 7.

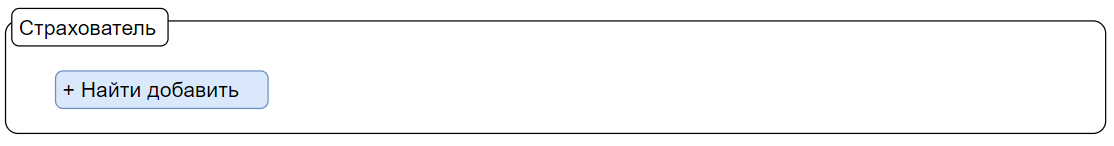


Рисунок 7 Прототип модуля страхователь.

* 1. Подмодуль «найти страхователя». В данном модуле содержится поиск страхователя по введенным данным от пользователя. Подмодуль «найти страхователя» изображен на рисунке 8.

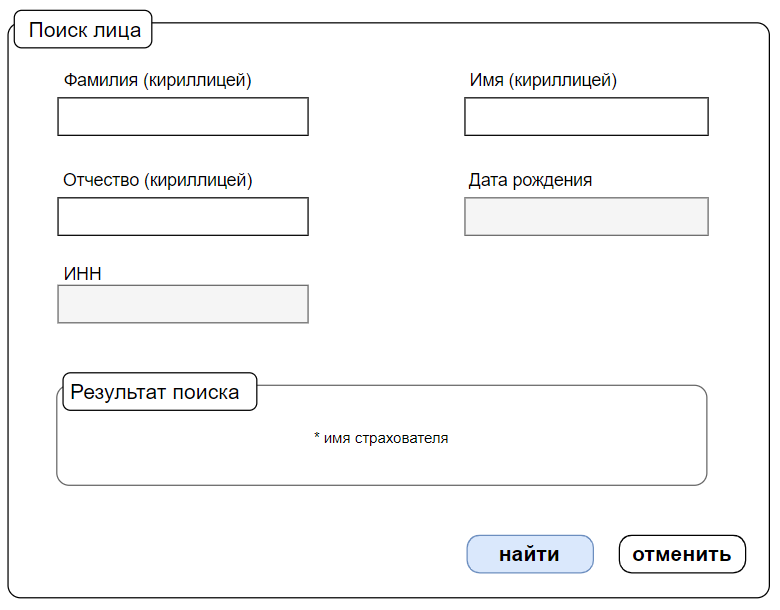


Рисунок 8. Прототип подмодуля поиск лица.

После того как пользователь ввел данные в поля ввода и нажал на кнопку «найти», то в поле «результат поиска» отобразиться кнопка с полным именем страхователя, нажав на которую, в контракте будет содержаться страхователь.

* 1. Подмодуль «создание физ. лица». В данном модуле расположены поля ввода, в которых пользователь должен ввести необходимые данные, после чего нажав на кнопку «сохранить», создаётся новый страхователь. Подмодуль «создание физ. лица» изображен на рисунке 9.

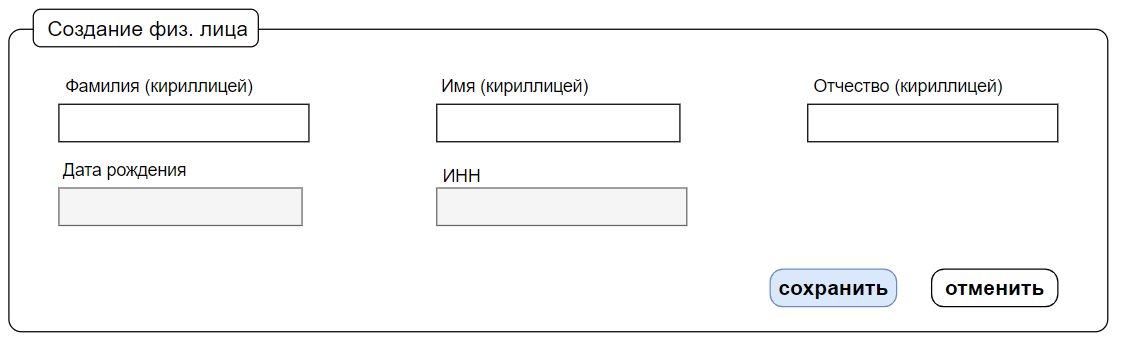


Рисунок 9. Прототип подмодуля создание физ. лица.

1. Модуль «транспортное средство». В данном поле расположены поля ввода, а именно поля с вводом транспортного средства. Модуль «транспортное средство» изображен на рисунке 10.

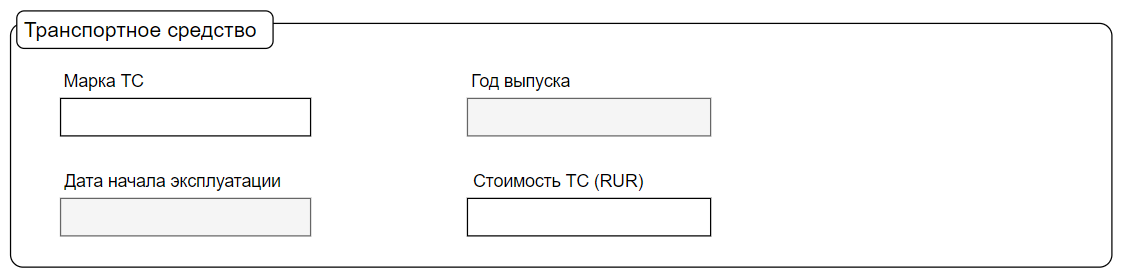


Рисунок 10. Прототип модуля транспортное средство.

1. Модуль «райдеры». В данном модуле лежит лишь один чек-бокс, по нажатию на который, отобразиться кнопка в модуле «основные продукты». Нажав на кнопку «Квартира Экспресс» отобразятся модули «Райдеры: Квартира Экспресс» и «объекты страхования». Модуль «райдеры» изображен на рисунке 11.

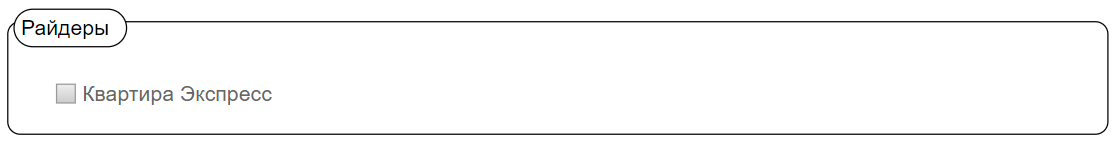


Рисунок 11. Прототип модуля райдеры.

1. Модуль «райдеры: квартира экспресс». В данном модуле расположены поля ввода, в которых необходимо ввести номер договора и количество комнат. Модуль «райдеры: квартира экспресс» изображен на рисунке 12.

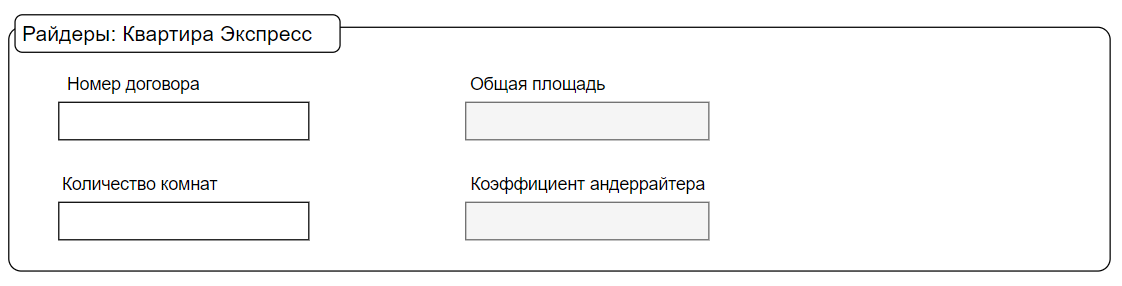


Рисунок 12. Прототип модуля райдеры: квартира экспресс.

**Микрофронтенд поиска контракта.**

После перехода из главного меню по нажатию на ссылку «Найти свой договор», пользователь должен попасть на страницу поиска контракта. Прототип микрофронтенда поиск контракта изображен на рисунке 13.

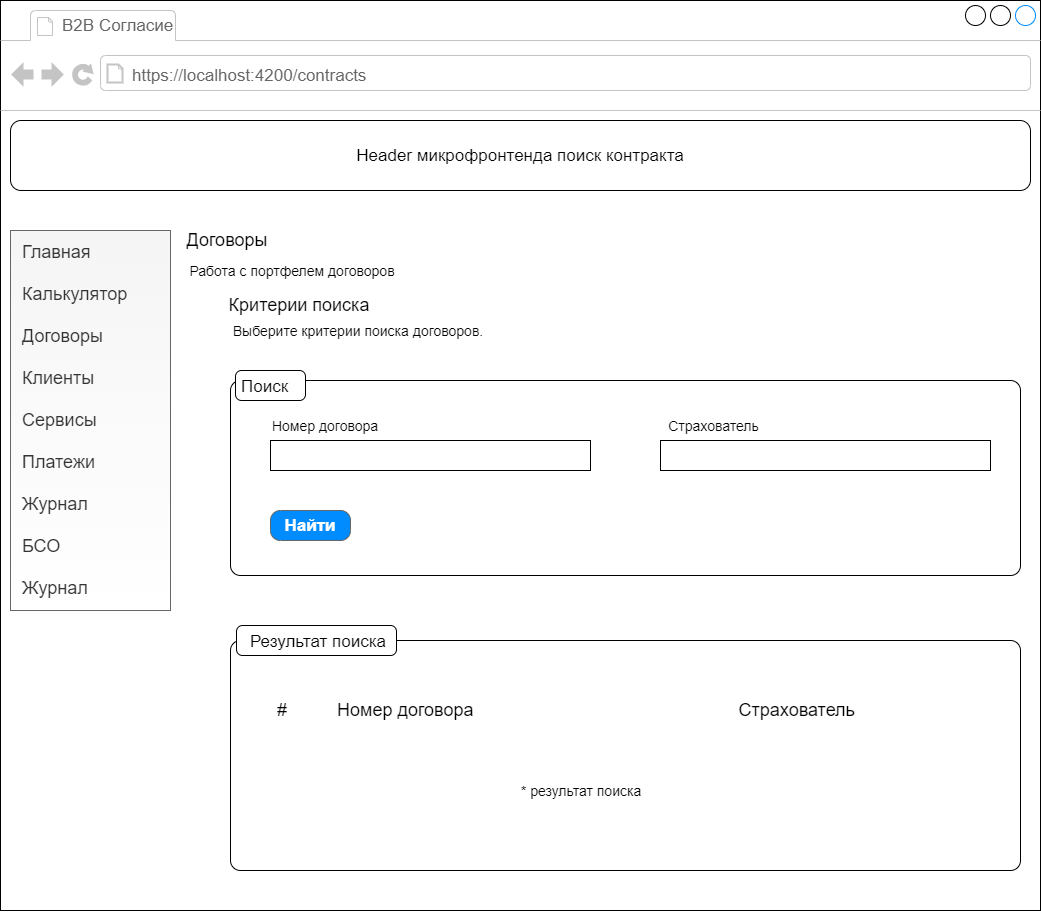


Рисунок 13. Прототип микрофронтенда поиск контракта.

Блок, в котором обозначен «Header микрофронтенда поиск контракта», должен быть идентичным, как и в микрофронтендах – меню и создание контракта.

В данной странице пользователь, заполнив определённые поля ввода, может выполнить поиск контракта. Соответственно контракт, который имеется в базе данных, высветиться в виде кнопки в модуле «результат поиска», нажав на которую можно попасть на страницу создания контракта, где будут заполнены все поля данного контракта.

* + - 1. Требования к интерфейсу

Представлены на рисунках в предыдущих пунктах.

* + - 1. Требования к надежности

К клиентской и серверной частям выдвигаются различные требования надёжно­сти. Система должна сохранять работоспособность и обеспечивать восстанов­ление своих функций при возникновении следующих внештатных ситуа­ций:

* при сбоях в системе электроснабжения аппаратной части, приводящих к перезагрузке ОС, восстановление программы не требуется, т.к. сервис находится в сети Интернет;
* при ошибках в работе аппаратных средств (кроме носителей данных и программ) восстановление функции системы возлагается на ОС;
* при ошибках, связанных с программным обеспечением (ОС и драйверы устройств), восстановление работоспособности возлагается на ОС;
* при падении сервера, функция восстановления системы возлагается на программиста.

Web-приложение «B2B Согласие» на основе микрофронтендной архитектуры, должно яв­ляться надёжной системой и должна продолжать работу даже при возникнове­нии какой-либо ошибки, исключая аппаратные сбои.

Вероятность непрерывной безотказной работы клиентской части полностью воз­лагается на OC пользователя.

Возможные причины отказа серверной части:

* превышение количества клиентов;

Данная причина может привести к тому, что сервер перестанет обрабаты­вать информацию, т.е. зависать; однако для решения этой проблемы необхо­димо поднять технические характеристики сервера;

* зависание сервера;

к этой причине можно отнести программные ошибки, т.е. ошибки в разрабо­танном коде;

* сбой сети;
* аппаратный сбой.

Вероятность непрерывной безотказной работы серверной части должна быть не менее 1 460 часов (≈ 2 месяца) с вероятностью 95%.

* + - 1. Требования к составу и параметрам технических средств

Разрабатываемая система должна иметь клиент-серверную архитектуру и состо­ять, соответственно, из двух подсистем: клиентской, отвечающую за интер­фейс, и серверной, отвечающей за сбор и обработку клиентских данных.

Системные требования для клиентской части полностью совпадают с систем­ными требованиями браузера IE 8, т.к. системой пользуется аудитория, кото­рая до сих пор пользуется этой версией браузера:

1. ОС: Windows XP – Windows 10;
2. процессор x86/x64 2 ГГц;
3. 512 Мб ОЗУ;
4. 1 Гб внешней памяти;
5. сетевое подключение 100 Мбит/с.

Требования к серверной части:

1. ОС: кроссплатформенный;
2. процессор двухъядерный x86/x64 3 ГГц;
3. 2 Гб ОЗУ;
4. 1 Тб внешней памяти;
5. сетевое подключение 1 Гбит/с.

Клиентская сторона для пользователя должна создавать иллюзию единолич­ного пользования системой.

* + - 1. Требования к информационной и программной совместимости

Требования к информационной и программной совместимости для клиент­ской и серверной части различны.

Клиентская часть должна эксплуатироваться под любой операционной систе­мой, которая имеет браузер для выхода в интернет. Серверная часть должна быть платформонезависимой.

Для просмотра результатов использовать один из браузеров Google Chrome 26.0, Mozilla Firefox 21.0, Internet Explorer 8.

Форматы сообщений, которые могут передаваться с клиентской части на сервер­ную и обратно, могут быть следующие:

* JSON;
* TEXT.
  + - 1. Специальные требования

Клиентская часть не может полноценно работать отдельно от серверной части.

* + 1. Требования к документации на дипломный проект (работу)

Программная документация должна содержать:

1. техническое задание;
2. пояснительную записку, имеющую структуру:

* назначение и область применения;
* технические характеристики;
* ожидаемые технико-экономические показатели;

1. описание программы, имеющее структуру:

* общие сведения;
* функциональное назначение;
* описание логической структуры;
* используемые технические средства;
* вызов и загрузка;
* входные данные;
* выходные данные.

1. программа и методика испытаний.

Документация должна быть представлена на русском языке, все используе­мые в ней аббревиатуры расшифрованы и переведены.

* 1. Пояснительная записка к программному обеспечению
     1. Анализ предметной области

Анализ предметной области приведен в разделе 1.

* + 1. Обзор аналогов

Обзор аналогов выполнен в подразделе 1.2.

* + 1. Цели создания системы

Реализация микрофронтендной архитектуры, которая обеспечит системе децентрализованную маршрутизацию и унифицированного событийно – ориентированного обмена между продуктами страховой компании и компаниями – партнерами.

* + 1. Назначение системы

Разрабатываемый продукт позволит снизить нагрузку на систему страховой компании, увеличит скорость взаимодействия с компаниями – партнерами, а также увеличит количество систем, способных взаимодействовать с системой страховой компании.

* + 1. Технические характеристики
       1. Общая структурная схема системы

Данная система выполнена на основе микрофронтендной архитектуры. Микрофронтендная архитектура – это такой подход создания веб-приложения, при которой вместо того чтобы исполнять все контексты приложения на одном веб-приложении, используется несколько небольших веб-приложений – микрофронтендов каждое из которых соответствуют какому-либо контексту, при этом все микрофронтенды независимы друг от друга в плане технологий, и также они могут быть расположены в различных путях и машинах.

Архитектура используема на предприятии, представлена на рисунке 14.

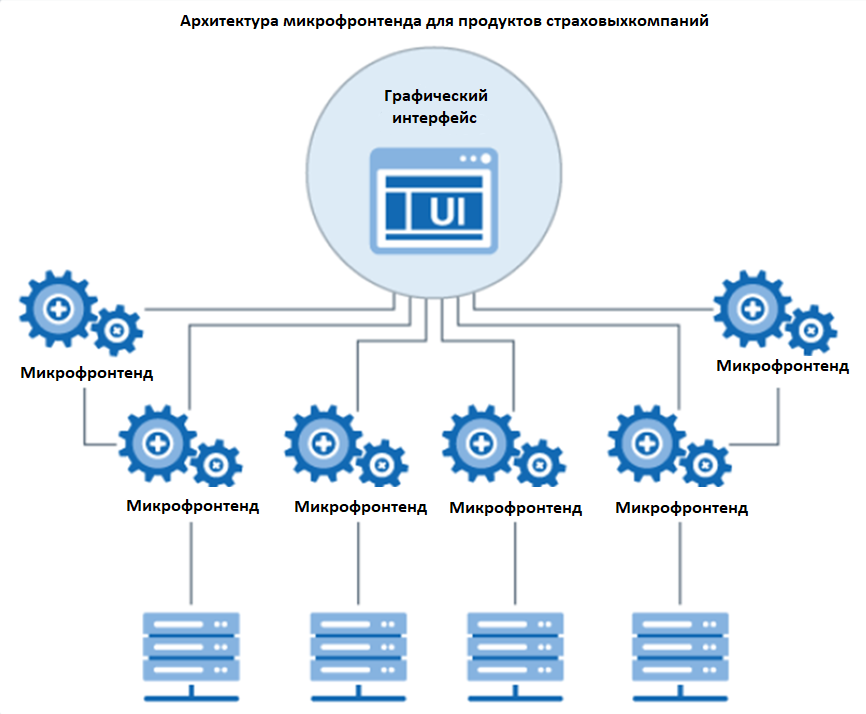


Рисунок 14. Микрофронтендная архитектура.

* + - 1. Описание и обоснование выбора состава технических и программных средств

**AngularJS**

Фреймфорк на базе языка – JavaScript, предназначенный для разработки одностраничных прило­жений. Его цель – расширение браузерных приложений на основе MVC шаб­лона, а также упрощение тестирования и разработки.

AngularJS спроектирован с убеждением того, что декларативное программирова­ние лучше всего подходит для построения пользовательских интерфей­сов и описания программных компонентов, в то время как императивное программирование отлично подходит для описания бизнес-логики. Фреймворк адаптирует и расширяет традиционный HTML, чтобы обеспечить двустороннюю привязку данных для динамического контента, что позволяет автоматически синхрони­зировать модель и представление. В результате AngularJS уменьшает роль DOM-манипуляций и улучшает тестируемость.

Данный фреймворк имеет функционал зонирования (Zone) всего веб-приложения. Зона хранит в себе все свойства, переменные, функции и т.д., определённого веб-приложения. Далее эти зоны можно отобразить, в определённом разработчиком участке страницы, определённую зону. Таким образом мы сможем организовать микрофронтендную архитектуру.

Также данный фреймворк позволяет разворачивать приложение на определённом хост-сервере. Это поможет создать определённый микрофронтенд и размещать его на хост-сервере, для того чтобы мы смогли в дальнейшем использовать его удобно и просто в плане маршрутизации.  
 AngularJs имеет встроенную маршрутизацию между компонентами, но при разработке центрального микрофронтенда, который должен будет распределять все остальные микрофронтенды – можно реализовать маршрутизацию между микрофронтендами.

**HTML5**

Стандартизированный язык разметки документов во Всемирной паутине. Язык HTML интерпретируется браузерами; полученный в результате интерпрета­ции форматированный текст отображается на экране монитора компьютера или мобильного устройства. В пятой версии языка улучшение уровня поддержки мульти­медиа-технологий с одновременным сохранением обратной совместимо­сти, удобочитаемости кода для человека и простоты анализа для парсеров.

В HTML5 реализовано множество новых синтаксических особенностей. Напри­мер, элементы <video>, <audio> и <canvas>, а также возможность использова­ния SVG и математических формул. Эти новшества разработаны для упрощения создания и управления графическими и мультимедийными объектами в сети без необходимости использования сторонних API и плагинов. Другие но­вые элементы, такие как <section>, <article>, <header> и <nav>, разработаны для того, чтобы обогащать семантическое содержимое документа (страницы). Новые атрибуты были введены с той же целью, хотя ряд элементов и атрибутов был уда­лён. Некоторые элементы, например – <a>, <menu> и <cite>, были изменены, переопре­делены или стандартизированы. API и DOM стали основными частями спецификации HTML5. HTML5 также определяет некоторые особенности обра­ботки ошибок вёрстки, поэтому синтаксические ошибки должны рассматриваться одинаково всеми совместимыми браузерами.

HTML5 был выбран для данной разработки так как разработка тесно связана с версткой. Задача состоит в разработке веб-приложения, без HTML5 данную задачу не достичь.

**CSS3**

**Cascading Style Sheets (CSS)** — это язык иерархических правил ([таблиц стилей](https://developer.mozilla.org/ru/docs/DOM/stylesheet)), используемый для представления внешнего вида документа, написанного на [HTML](https://developer.mozilla.org/ru/docs/HTML) или [XML](https://developer.mozilla.org/ru/docs/XML) (включая различные языки XML, такие как [SVG](https://developer.mozilla.org/ru/docs/SVG) и [XHTML](https://developer.mozilla.org/ru/docs/XHTML)). CSS описывает, каким образом элемент должен отображаться на экране, на бумаге, голосом или с использованием других медиа средств.

**CSS** является одним из основных языков свободной веб-разработки, который стандартизован [спецификацией W3C](http://w3.org/Style/CSS/#specs). Стандарт CSS делится на уровни: CSS1 в настоящее время устарел, CSS2.1 — рекомендован для применения, а [CSS3](https://developer.mozilla.org/ru/docs/CSS/CSS3), разбитый на более мелкие модули, развивается на пути стандартизации.

**Формат JSON**

Данный формат имеет следующие преимущества:

1. стандартность;
2. читаемость людьми;
3. структурированность;
4. лаконичность.

JSON (англ. JavaScript Object Notation) – текстовый формат обмена дан­ными, основанный на JavaScript и обычно используемый именно с этим язы­ком. Как и многие другие текстовые форматы, JSON легко читается людьми. Несмотря на происхождение от JavaScript, формат считается языконезависи­мым и может использоваться практически с любым языком программирова­ния. Для многих языков существует готовый код для создания и обработки данных в формате JSON. За счёт своей лаконичности по сравнению с XML, формат JSON может быть более подходящим для сериализации сложных струк­тур.

**Node.js**

Node или Node.js  – программная платформа, основанная на движке [V8](https://ru.wikipedia.org/wiki/V8_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript)) (транслирующем [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) в [машинный код](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4)), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) взаимодействовать с устройствами [ввода-вывода](https://ru.wikipedia.org/wiki/%D0%92%D0%B2%D0%BE%D0%B4-%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4) через свой [API](https://ru.wikipedia.org/wiki/API) (написанный на [C++](https://ru.wikipedia.org/wiki/C%2B%2B)), 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](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи [NW.js](https://ru.wikipedia.org/wiki/NW.js), AppJS или [Electron](https://ru.wikipedia.org/wiki/Electron) для [Linux](https://ru.wikipedia.org/wiki/Linux), [Windows](https://ru.wikipedia.org/wiki/Windows) и [macOS](https://ru.wikipedia.org/wiki/MacOS)) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит [событийно-ориентированное](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [асинхронное (или реактивное)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) программирование с [неблокирующим вводом/выводом](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B0%D1%8F_%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F).

Серверная часть, данной разработки, написана на платформе Node.Js, так как передо мной стояла задача – быстрая разработка серверной задачи, а данная платформа знакома мне своим синтаксисом, так как я в свободное время разрабатывал приложения на JavaScript. Данная платформа не хуже остальных на рынке.

* + - 1. Постановка задачи на разработку системы

Для заключения договора между страховой компанией и клиентом, человеку необходимо представить личные данные и ответить на ряд вопросов, связанных с его транспортным средством.

Также клиент, помимо страховки транспортного средства, может заодно и оформить страхование квартиры и изменить страхователя. Это все накладывает на одну систему «B2B Согласие» большое количество страховых продуктов. Для упрощения работы страховой компании было бы удобно, с технической точки зрения и разработки в целом, разделить систему на подсистемы, а именно разделения веб-приложения на микрофронтенды.

* + - 1. Ожидаемые технико-экономические показатели

Результатом разработки должно стать действующее веб-приложение «B2B Согласие».

Техническая сторона реализации должна предусматривать гибкое измене­ние используемых внутри неё структур для адаптации под решение новых задач. Любой дополнительный функционал может быть добавлен без коренных изменений в системе центрального микрофронтенда, а лишь путём модификации как интерфейсных частей, так и северной части.

Разрабатываемое приложение позволит сократить время работы разработчика страховых продуктов по обработке одного заявления на страхование, что приведёт к увеличению производительности сотрудника.

Для страховой компании отдача от приложения будет практически сразу, после обучения сотрудников, занимающихся разработкой микрофронтендов для страховых продуктов.

* 1. Программа и методика испытаний
     1. Объект испытаний

Объектом испытаний является веб-приложение «B2B Согласие», на основе микрофронтендной архитектуры. Испытания проводятся для клиентской части, для серверной части и для системы в целом.

* + 1. Цель испытаний

Целью испытаний является проверка:

* общей работоспособности;
* стабильности работы;
* корректности результатов и выходных данных;
* работоспособности при неверных входных данных;
* соответствия имеющегося функционала требованиям, предъявленным объ­екту испытаний в техническом задании.
  + 1. Требования к программе

Для проверки написанного функционала, будем использовать один из продуктов – «B2B Согласие». Для него был создан микрофронтенд, для бо­лее уверенного тестирования.

Требования к клиентской и серверной частям различны. Перечислим основ­ные требования, проверяемые при проведении испытаний.

Основные требования к клиентской части:

1. клиент должен корректно отправлять данные;
2. при попытке ввода логина при авторизации, которого нет в си­стеме, интерфейс должен уведомить пользователя сообщением;
3. при попытке авторизации, пользователю должен прийти сгенерирован­ный код;
4. корректный поиск в системе.

Основные требования к серверной части:

1. сервер корректно принимает все данные;
2. сохраняемая в базе данных информация соответствует передаваемым данным;
3. при создании нового контракта, сервис должен сохранить данный контракт в базе данных;
4. загрузка данных определённого контракта должна производиться корректно (все данные должны быть такими, как при создании и не меняться каж­дый раз после того, как клиент выходит из ПК).
   * 1. Методы испытаний

Методика испытаний состоит в проверке эквивалентности реакции си­стемы на те, или иные действия эталонной реакции.

Необходимо проверить соответствуют ли программа требованиям, кото­рые предоставлены заказчиком. Для проверки развернем приложение на тестовом сервере и подготовим список необходимых тестов (таблица 1).

Таблица 1 - испытания.

|  |  |  |
| --- | --- | --- |
| Действие | Реакция | |
| 1. **Испытания клиентской части** | | |
| При отправке верных данных (в верном формате) на сервер, сер­вер отрабатывает без ошибок | | Получение, клиентом, статуса сообщения, который приходит с сервера, должен быть – «OK». |
| При отправке неверных данных (в неверном формате) на сервер, сервер возвращает статус ошибки. | | Получение, клиентом, статуса сообщения, который приходит с сервера, должен быть – «Введены неправильно данные», либо – «Данного пользователя не существует». |
| При вводе неверных данных, клиент должен уведомлять об возникших ошибках в виде сооб­щения. | | При соответствующих ошибках, система должна вывести ошибку с указанием проблемы. |
| 1. **Испытания серверной части** | | |
| Сервер корректно прини­мает все данные. | | Данные соответствуют типу, отправляемыми клиентом. |
| Сохраняемая в базе данных инфор­мация соответствует пере­даваемым данным. | | Все данные, которые приходят с клиента, должны сохранятся в том же типе, что и прихо­дят. |
| При создании нового контракта, сервис должен сохранить данный контракт в базу данных. | | Сервис должен корректно, с необходи­мыми данными, создать новый контракт в базе данных. |
| Загрузка данных определённого контракта, должна произво­диться корректно (все данные должны быть такими, как при создании не меняться каж­дый раз после того, как клиент выходит из ПК). | | Данные, которые должна загрузить си­стема, должны быть привязаны к одному и тому же клиенту, и быть неизмен­ными до тех пор, пока их не сменит пользо­ватель. |

1. Акт испытаний программного продукта
   1. Объект испытаний

Объектом испытаний является web-приложение «B2B Согласие». Испытания проводятся для клиентской ча­сти, для серверной части и для системы в целом.

* 1. Цель испытаний

Целью испытаний является проверка:

* общей работоспособности;
* стабильности работы;
* корректности результатов и выходных данных;
* работоспособности при неверных входных данных;
* соответствия имеющегося функционала требованиям, предъявленным объ­екту испытаний в техническом задании.
  1. Результат испытаний

Испытания проводились в соответствии разделом 2.3 – «Программа и мето­дика испытаний».

Испытания проводились в отделе разработки. Для испытаний использова­лась корпоративная локальная сеть, включающая 10 машин. Каж­дая из машин значительно превосходила заявленные минимальные требова­ния.

Проведение испытаний проходило строго в соответствии с таблицей «Ис­пытания» (см. раздел 2.3.4 – «Методы испытаний»). Результаты всех испыта­ний совпали с эталонными, приведёнными в таблице.

* 1. Выводы

По результатам проведения тестирования программного продукта в соот­ветствии со всеми пунктами методики испытаний можно сделать следую­щие выводы:

* разработанная система полностью удовлетворяет требованиям, сформули­рованным в техническом задании;
* разработанная система является законченным программным продук­том;
* разработанная система может применять в реальных условиях.

Работу можно считать завершенной с положительным результатом.

Руководитель дипломного проекта:    Волков М.Л.   
  
Разработчик дипломного проекта, студент группы ИПБ-16:    Рахимов К.С.

1. Экономическая часть
   1. Экономическое обоснование разрабатываемого программного обеспечения
   2. Описание функций, назначения и потенциальных пользователей программного обеспечения
   3. Расчет затрат на разработку программного обеспечения
   4. Оценка результата от продажи программного обеспечения
   5. Расчет показателей эффективности инвестиций в разработку программного обеспечения.

Акт о внедрении

Разработка была полностью разработана и внедрена в организацию ООО «БиАйВи», а также принято в профессиональную эксплуатацию.

Акт изображен на рисунке 15.



Рисунок 1. Прототип микрофронтенда авторизации.

Заключение

В ходе выполнения дипломного проекта была спроектирована система с учётом предъявленных требований и различные её части, выбраны основ­ные платформы для реализации и форматы представления данных, спроектиро­ваны структуры представления данных в различных частях си­стемы.

В результате выполнения работы по разработке микрофронтендного подхода, можно сделать вывод о том, что разработанный продукт полностью удовлетворяет потребностям системы страховой компании в быстродействии, надежности, безопасности при взаимодействии со сторонними системами. Также есть потенциальная возможность расширения функционала интеграции путем добавления новых методов для центрального микрофронтенда и микрофронтендов в целом.

Итогом выполнения дипломного проекта стала реализованная и ре­ально действующая система, которая сокращает время разработки для разработчиков страховых продуктов, а также имеет независимость в плане выбора технологий.

Как покажет анализ экономической эффективности, который будет представлен в главе 5, система будет являться эко­номически эффективной.

Литература

1. Страхование сегодня [Электронный ресурс]. — Режим доступа: URL: <http://www.insur-info.ru>
2. Сущность и функции страхования [Электронный ресурс]. — Режим доступа: URL: <https://kosago.ru/strahovka/funktsii-strakhovoy-kompanii/>
3. Сухоруков М.М. Организация продаж страховых продуктов [Электронный ресурс]. — Режим доступа: URL: <https://www.lawmix.ru/bux/51621>
4. Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.: ил. (Серия «Библиотека программиста»)
5. Микрофронтенды на tinkoff.ru [Электронный ресурс]. — Режим доступа: URL: <https://frontendconf.ru/moscow/2020/abstracts/5993>
6. Микрофронтенды: о чем это мы? [Электронный ресурс]. — <https://habr.com/ru/company/raiffeisenbank/blog/459540/>
7. Angular для профессионалов. — СПб: Питер, 2018.
8. Angular и TypeScript. Сайтотворение для профессионалов. — СПб: Питер, 2018.
9. JavaScript в примерах и задачах. — СПб: Питер, 2017.

Приложение

## Приложение А

Разметка и код HTML документа, в котором содержится центральный микрофронтенд:

<!DOCTYPE html>

<html>

  <head>

    <meta http-equiv="Content-Security-Policy" content="default-src \*  data: blob: 'unsafe-inline' 'unsafe-eval'; script-src \* 'unsafe-inline' 'unsafe-eval'; connect-src \* 'unsafe-inline'; img-src \* data: blob: 'unsafe-inline'; frame-src \*; style-src \* data: blob: 'unsafe-inline'; font-src \* data: blob: 'unsafe-inline';">

    <meta charset="utf-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <title>Microservice</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <meta name="importmap-type" content="systemjs-importmap">

    <script type="systemjs-importmap">

      {

        "imports": {

          "auth": "http://localhost:4201/main.js",

          "autocontract": "http://localhost:4202/main.js",

          "contracts": "http://localhost:4203/main.js",

          "main": "http://localhost:4300/main.js",

          "single-spa": "https://cdnjs.cloudflare.com/ajax/libs/single-spa/4.3.5/system/single-spa.min.js"

        }

      }

    </script>

    <link rel="preload" href="https://cdnjs.cloudflare.com/ajax/libs/single-spa/4.3.5/system/single-spa.min.js" as="script" crossorigin="anonymous" />

    <script src='https://unpkg.com/core-js-bundle@3.1.4/minified.js'></script>

    <script src="https://unpkg.com/zone.js"></script>

    <script src="https://unpkg.com/import-map-overrides@1.6.0/dist/import-map-overrides.js"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/4.0.0/system.min.js"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/4.0.0/extras/amd.min.js"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/4.0.0/extras/named-exports.js"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/4.0.0/extras/named-register.min.js"></script>

    <script src="https://code.jquery.com/jquery-3.4.1.slim.js" integrity="sha256-BTlTdQO9/fascB1drekrDVkaKd9PkwBymMlHOiG+qLI=" crossorigin="anonymous"></script>

</head>

<body>

  <script>

      System.import('single-spa').then(function (singleSpa) {

        singleSpa.registerApplication(

          'main',

          function () {

            return System.import('main');

          },

          function (location) {

            return location.pathname.startsWith('/main');

          }

        );

        singleSpa.unloadApplication('main', {waitForUnmount: true});

        singleSpa.registerApplication(

          'auth',

          function () {

            return System.import('auth');

          },

          function (location) {

            return location.pathname.startsWith('/auth');

          }

        );

        singleSpa.unloadApplication('auth', {waitForUnmount: true});

        singleSpa.registerApplication(

          'autocontract',

          function () {

            return System.import('autocontract');

          },

          function (location) {

            return location.pathname.startsWith('/auto/contract');

          }

        );

        singleSpa.unloadApplication('autocontract', {waitForUnmount: true});

        singleSpa.registerApplication(

          'contracts',

          function () {

            return System.import('contracts');

          },

          function (location) {

            return location.pathname.startsWith('/contracts');

          }

        );

        singleSpa.unloadApplication('contracts', {waitForUnmount: true});

        singleSpa.start();

      })

    </script>

  </body>

</html>

## Приложение Б

Пример кода и разметки одного из микрофронтендов:

Код документа, где инициализируется жизненный цикл микрофронтенда:

import { enableProdMode, NgZone } from '@angular/core';

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { Router } from '@angular/router';

import { AppModule } from './app/app.module';

import { environment } from './environments/environment';

import singleSpaAngular from 'single-spa-angular';

import { singleSpaPropsSubject } from './single-spa/single-spa-props';

if (environment.production) {

  enableProdMode();

}

const lifecycles = singleSpaAngular({

  bootstrapFunction: singleSpaProps => {

    singleSpaPropsSubject.next(singleSpaProps);

    return platformBrowserDynamic().bootstrapModule(AppModule);

  },

  template: '<auth-root />',

  Router,

  NgZone: NgZone,

});

export const bootstrap = lifecycles.bootstrap;

export const mount = lifecycles.mount;

export const unmount = lifecycles.unmount;