**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ**

**И ИНФОРМАТИКИ**

**Кафедра многопроцессорных систем и сетей**

**РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ ДЛЯ УПРАВЛЕНИЯ ЦВЕТОЧНЫМИ ЗАКАЗАМИ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКОВ SPRING И ANGULARJS**

Курсовая работа

Павловой Маргариты Валерьевны

студентки 4 курса,

специальность «информатика»

Научный руководитель:

доцент

Рафеенко Е.Д.

Минск, 2015

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc419933436)

[1 ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ 5](#_Toc419933437)

[1.1 Spring 5](#_Toc419933438)

[1.2 AngularJS 10](#_Toc419933439)

[1.3 Nexmo 13](#_Toc419933440)

[1.4 Google Maps 16](#_Toc419933441)

[2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 19](#_Toc419933442)

[2.1 Структура проекта 19](#_Toc419933443)

[2.2 Технические возможности созданного приложения 23](#_Toc419933444)

[ЗАКЛЮЧЕНИЕ 29](#_Toc419933445)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc419933446)

**РЕФЕРАТ**

Курсовой проект, 19 страниц, 4 рисунка, 4 источника.

**Ключевые слова**:SPRING, SPRING SECURITY, HIBERNATE, ANGULARJS, КОНТЕЙНЕР, ФРЕЙМВОРК, ФАБРИКА, КОМПОНЕНТА, МОДУЛЬ, ДИРЕКТИВА, NEXMO, GOOGLE MAPS.

**Объект исследования:** технологии для разработки WEB-приложений.

**Цель работы:** разработка программной системы для управления цветочными заказами с использованием фреймворков Spring и AngularJS.

**Методы работы:** исследование эффективных решений для создания WEB-приложения.

**В результате** проведенной работы были исследованы технологии для разработки WEB-приложения.

# ВВЕДЕНИЕ

WEB-приложение – это прикладное программное обеспечение, логика которого распределена между сервером и клиентом, а обмен информацией происходит по сети. Клиентская часть реализует пользовательский интерфейс, а серверная – получает и обрабатывает запросы от клиента, выполняет вычисления, формирует WEB-страницу и отправляет ее клиенту согласно протоколу HTTP.

В настоящее время возросла популярность использования WEB-приложений. Это основано на том, что WEB-приложение имеет преимущества над традиционными компьютерными программами:

1. WEB-приложение доступно в любой точке мира, где есть Интернет;
2. независимость от операционной системы и даже от устройства;
3. WEB-приложение не надо устанавливать на компьютер;
4. используя WEB-приложение, не надо беспокоиться об обновлениях. Пользователь всегда гарантировано будет работать с самой последней версией.

На данный момент в Интернете активно используются сайты, связанные с заказом цветов. Это вызвано тем, что теперь пользователю не требуется посещать места продаж цветов, а достаточно выбрать интересующие его товары (в частности, сорт цветов и количество) и оформить заказ. Далее оператор, отвечающий за обработку заказа, связывается с пользователем и подтверждает заказ.

Таким образом, в курсовом проекте будут представлены технологии разработки WEB-приложений и будет реализован каркас WEB-приложения.

1. ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

# 1.1 Spring

**Исследование алгоритмов поиска изображений по содержанию сейчас является чрезвычайно развивающейся дисциплиной. Основными направлениями алгоритмов является дискриминантный анализа на нейронных сетях и методы опорных векторов. CBIR извлекает из каждого изображения несколько ключевых характеристик, которые будут использоваться для вычисления подобия между изображениями. Эти характеристики описывают содержимое изображения, и поэтому они должны быть тщательно выбраны в соответствии с контекстом. Поэтому в этой системе будет рассмотрен поиск похожих изображений в терминах цвета и текстуры.**

**Цвет. Цветовая характеристика представлена с помощью среднего цвета**  и матрицы ковариации . Пусть

**Где – число пикселей изображения, пикселя в канале . Предполагая использование оригинального RGB (Red, Green, Blue) цветового пространства, матрица ковариаций будет:**

**Текстура. Текстурная характеристика извлечена из обобщенной матрицы смежности. Предположим, изображение содержит столбцов и   
строк и уровней серого цвета. Пусть представляют столбцы, – строки, и – множество уровней серого цвета. Смежная матрица – матрица размерности , где**

**Т.е. количество смежных пар серого уровня и , для которых расстояние Пары пикселей и имеют расстояние и ориентацию**

**Обобщенная смежная матрица это расширение смежной матрицы для мультиспектральных изображений, т.е. изображений, представленных n цветовыми каналами. Пусть и два цветовых канала. Обобщенная ковариационная матрица будет:**

**Например, в случае цветного изображения, закодированного в трех каналах (RGB), имеется шесть смежных матриц: (RR), (GG), (BB), которые так же, как и смежная матрица серого уровня, вычисляются по одному каналу, и (RG), (RB), (GB), которые учитывают корреляцию между каналами.**

**Система поиска основана на мере сходства между заданным изображением и изображением базы данных .**

**Для цветовой характеристики, основанной на матрице ковариаций, мера расстояний Бхаттачарйя используется в следующем виде:**

**Где и вектора средних цветовых характеристик, и – ковариационные матрицы изображений и , - определитель матрицы.**

**Расстояние Евклида используется как расстояние между текстурными характеристиками**

**Объедим два расстояния в функцию согласующегося подобия:**

**Где**  и веса факторов, которые должны быть выбраны экспериментально.

**Метод поиска похожих изображений с использованием индекса качества изображений.**

**Поиск похожих изображений является важным предметом исследований для мультимедийного управления базами данных. Эта статья используется модель индекса качества для поиска аналогичных изображений с цифровых баз данных. Чтобы ускорить извлечение, модель индекса качества разделяется на 3 фактора: потеря корреляции, искажение яркости и контрастности искажений.**

**В обработке изображений наиболее часто используемыми измерениями для оценки разницы между двумя изображениями являются среднеквадратичная ошибка (MSE), пиковое соотношение сигнала к шуму (PSNR), средняя абсолютная ошибка (MAE) и так далее. Эти методы измерения обычно легкие и имеют более низкую сложность вычислений. Тем не менее, в соответствии с экспериментами, большинство из методов не могут быть точно применены в строгих условиях тестирования или в искажении изображения окружающей средой. Кроме того, эти способы требуют человеческого зрения для измерения качества. Поэтому был предложен другой способ измерения качества изображения – индекс универсального качества изображения (индекс качества), чтобы оценить разницы между двумя изображениями.**

**Из-за быстрого продвижения в информации технология позволила получить доступ к большому количеству изображений в одно мгновение из любого уголка мира. Тем не менее из-за огромного количества доступных изображений, найти правильный образ из большого числа очень трудно. В начале ключевые слова были использованы для поиска требуемых изображений. Каждое изображение в образе базы данных связано с аннотацией ключевых слов. Но такие методы стали бесполезны и неэффективны, так как все больше и больше изображений поступают из Интернета и многие из них не включают ключевые слова. К тому же каждое изображение включает различные ключевые слова. Таким образом, некоторые исследователи разработали другие методы поиска подобных изображений. Один популярный метод поиска изображений по содержанию (CBIR), которых извлекает характеристики такие как цвет, текстура, форма, время, расстояние.**

**В данном алгоритме примем концепцию показателя качества для представления характеристики изображения для поиска похожих образов. Чтобы ускорить извлечение, модель индекса качества разделяется на 3 фактора: потеря корреляции, искажение яркости и контрастности искажений.**

**В 2003 году был предложен метод на основе CDESSO, который характеризовал сложность цвета, цветовые различия между соседними пикселями для 24-битного полноцветного изображения. Во-первых, был использован метод алгоритм K среднего, чтобы уменьшить цветовое пространство изображения. Он разделял все пиксели изображения на 64 кластера. Каждый кластер имел свой собственный бункер для записи разницы между соседними пикселями. Во-вторых, каждый пиксель в изображении был вставлен в ближайший кластер. Затем было просканировано уменьшенное изображение в спиральном порядке и вычислили разность между каждыми двумя соседними пикселями. Разность добавляется к соответствующему кластеру текущего пикселя. Было использовано 64 кластера, называемые цветовой гистограммой, как характеристики исследуемого изображения. С точки зрения поиска изображений, гистограмма заданного изображения сравнивается с гистограммой существующих изображений, используя Евклидово расстояние. По их экспериментальным результатам видно, что метод обеспечивает не только высокую степень точности, но и не изменяет своего качества при вращении изображения. Способ поиска изображения CDESSO, а также другие цветовые методы, основанные на CBIR, используют Евклидово расстояние, чтобы оценить разницу между двумя изображениями. Однако измерения, такие как MSE, PSNR и Евклидово расстояние, не могут быть точно применены в строгих условиях тестирования или при искажении изображения окружающей средой. Поэтому был предложен математически определенный универсальный указатель качества изображения.**

**Пусть оригинальное изображение, где - i-ый пиксель изображения , и – тестовое изображение. Расстояние между изображениями и измеряется с помощью:**

**Где – среднее изображения , определенное** ,  **– среднее изображения , определенное** , – дисперсия **,**  – дисперсия **,**  – корреляция между изображением  **и изображением , определяемая**

Диапазон Q от -1 до 1. Если значение Q близко к 1, тогда можно сказать, что оригинальное изображение и тестируемое изображение схожи. Если значение Q равно -1, тогда оригинальное изображение и тестируемое изображение абсолютны различны.

Предположим, что изображение А представляет собой цифровой образ размера . Большинство цифровых изображений представлены в цветовом пространстве RGB. Таким образом, каждый пиксель может быть интерпретирован в 3 измерениях: красном, зеленом и синем. Однако пространство RGB не очень удобно для анализа изображений. Следовательно, первый шаг предложенного метода состоит в преобразовании цветового 3-размерного пространства в 1-размерное серое пространство. Пусть  **есть изображение А в 1-размерном сером пространстве. Пусть В – тестовое изображение. Первым шагом в поиске изображений, похожих на В заключается в преобразовании изображения В в 1-размерное серое пространство . Следующим шагом является использование уравнения (1) для расчета индекса качества между изображениями и . Изображение с большим показателем индекса качества – изображение, которое наиболее схоже с тестовым. Поскольку диапазон Q от -1 до 1, то когда Q равно 1 можно сказать, что изображения и одинаковы. В модели индекса качества значение Q вычисляется с помощью среднего, дисперсии и корреляции между и . Среднее и дисперсия могут быть вычислены независимо. Но корреляцию нужно вычислять динамически, что требует больше времени. Поэтому формулу показателя качества (1) можно преобразовать. Новая формула может отфильтровать несовпадающие изображения заранее, чтобы ускорить поиск схожих изображений. В соответствии с определением модели, индекс качества является комбинацией трех факторов: потери корреляции, искажение яркости и контраст искажений. Измененное уравнение индекса качества:**

**Где**  – коэффициент корреляции, – искажение

яркости, – контраст искажения. Диапазон от -1 до 1; диапазон

от 0 до 1; диапазон от 0 до 1. Можно заметить, что только значение касается каждого пикселя  **и** . Величины  **, ,** . Пусть . Тогда индекс качества может быть представлен . Зная диапазон значений , любое значение может уменьшить значение Q, за исключением . Следовательно, можно заранее определить порог Т, чтобы отфильтровать маловероятные изображения, для которых . Другими словами, сравнивать необходимо те изображения, для которых .

**Spring – это свободно распространяемый фреймворк, созданный Родом Джонсоном (Rod Johnson) и описанный в его книге «Expert One-on-One: J2EE Design and Development». Он был создан с целью устранить сложности разработки корпоративных приложений и сделать возможным использование простых компонентов JavaBean для достижения всего того, что ранее было возможным только с использованием EJB. Однако область применения Spring не ограничивается разработкой программных компонентов, выполняющихся на стороне сервера. Любое Java-приложение может использовать преимущества фреймворка в плане простоты, тестируемости и слабой связанности.**

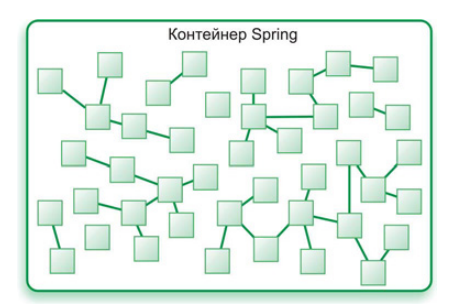
**Фреймворк Spring обладает весьма широкими возможностями. Но в основе практически всех его особенностей лежат несколько фундаментальных идей, направленных на достижение главной цели – упрощение разработки приложений на языке Java. Целью фреймворка Spring является упрощение разработки приложений Java вообще.**

**В своем устремлении на сложности, связанные с разработкой на языке Java, фреймворк Spring использует четыре ключевые стратегии:**

* **легковесность и ненасильственность благодаря применению простых Java-объектов (POJO);**
* **слабое связывание посредством внедрения зависимостей и ориентированности на интерфейсы;**
* **декларативное программирование через аспекты и общепринятые соглашения;**
* **уменьшение объема типового кода через аспекты и шаблоны.**

**В приложениях на основе фреймворка Spring прикладные объекты располагаются внутри контейнера Spring. Как показано на рис. 1.1, контейнер создает объекты, связывает их друг с другом, конфигурирует и управляет их полным жизненным циклом, от зарождения до самой их смерти (или от оператора new до вызова метода finalize()).**

**Контейнер находится в ядре фреймворка Spring Framework. Для управления компонентами, составляющими приложение, он использует прием внедрения зависимостей (DI). Управление включает создание взаимосвязей между взаимодействующими компонентами.**



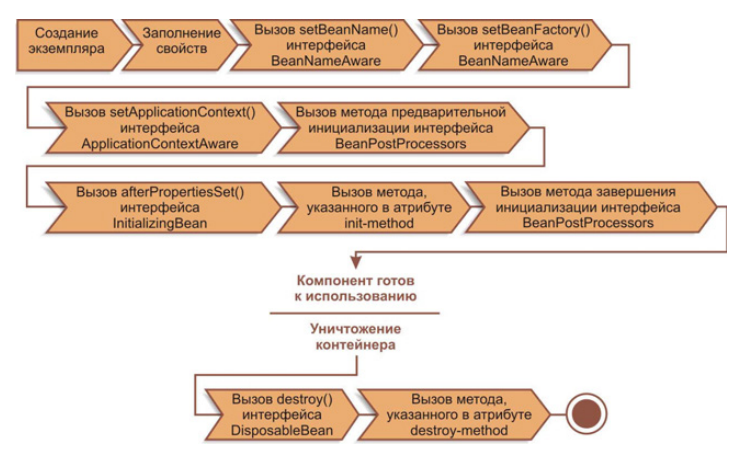
**Рисунок 1.1 – Контейнер Spring**

**Фреймворк Spring имеет не один контейнер. В его состав входят несколько реализаций контейнера, которые подразделяются на два разных типа. Фабрики компонентов (bean factories) – самые простые из контейнеров, обеспечивающие базовую поддержку DI. Контекст приложений (application contexts) основан на понятии фабрик компонентов и реализует прикладные службы фреймворка, такие как возможность приема текстовых сообщений из файлов свойств и возможность подписывать другие программные компоненты на события, возникающие в приложении [1].**

**С фреймворком Spring можно работать, используя и фабрики компонентов, и контексты приложений, но для большинства приложений фабрики компонентов часто оказываются слишком низкоуровневым инструментом. Поэтому контексты приложений выглядят более предпочтительно, чем фабрики компонентов.**

**В составе Spring имеется несколько разновидностей контекстов приложений. Три из них используются наиболее часто:**

* **ClassPathXmlApplicationContext – загружает определение контекста из XML-файла, расположенного в библиотеке классов (classpath), и обрабатывает файлы с определениями контекстов как ресурсы;**
* **FileSystemXmlApplicationContext – загружает определение контекста из XML-файла в файловой системе;**
* **XmlWebApplicationContext – загружает определение контекста из XML-файла, содержащегося внутри веб-приложения.**

**Жизненный цикл компонента внутри контейнера Spring очень сложный. Рисунок 1.2 иллюстрирует начальные этапы жизненного цикла типичного компонента, которые он минует, прежде чем будет готов к использованию.**

**Рисунок 1.2 – Жизненный цикл компоненты внутри контейнера Spring**

**Как показано на рисунке 1.2, фабрика компонентов выполняет несколько подготовительных операций, перед тем как компонент будет готов к использованию. Жизненный цикл компонента состоит из следующих этапов:**

**1. Spring создает экземпляр компонента;**

**2. Spring внедряет значения и ссылки на компоненты в свойства данного компонента;**

**3. Если компонент реализует интерфейс BeanNameAware , Spring пе-редает идентификатор компонента методу setBeanName();**

**4. Если компонент реализует интерфейс BeanFactoryAware , Spring вызывает метод setBeanFactory() , передавая ему саму фабрику компонентов;**

**5. Если компонент реализует интерфейс ApplicationContextAware, Spring вызывает метод setApplicationContext(), передавая ему ссылку на вмещающий контекст приложения;**

**6. Если какие-либо из компонентов реализуют интерфейс BeanPost Processor, Spring вызывает их методы postProcessBeforeInitialization();**

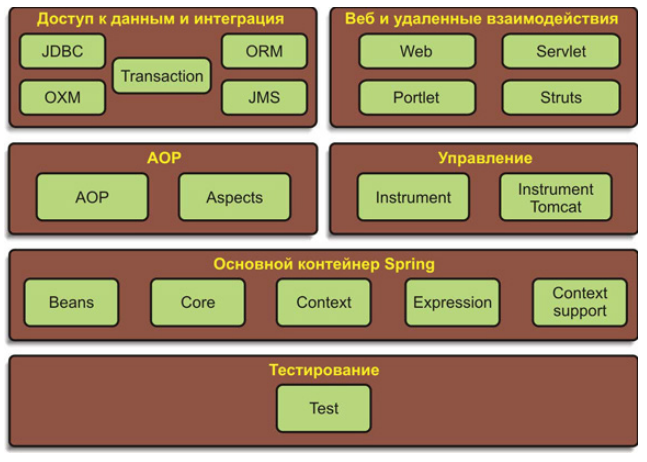
**7. Если какие-либо из компонентов реализуют интерфейс InitializingBean , Spring вызывает их методы afterPropertiesSet() . Аналогично, если компонент был объявлен с атрибутом initmethod, вызывается указанный метод инициализации;**

**8. Если какие-либо из компонентов реализуют интерфейс BeanPostProcessor , Spring вызывает их методы postProcessAfterInitialization();**

**9. В этот момент компонент готов к использованию приложением и будет сохраняться в контексте приложения, пока он не будет уничтожен;**

**10. Если какие-либо из компонентов реализуют интерфейс DisposableBean, Spring вызывает их методы destroy(). Аналогично, если компонент был объявлен с атрибутом destroy-method, вызывается указанный метод.**

**При работе с фреймворком Spring Framework можно обнаружить еще несколько скрытых путей, упрощающих разработку на языке Java. Однако за пределами Spring Framework существует гигантская экосистема проектов, построенных на базе основного фреймворка и расширяющих возможность применения Spring на такие области, как веб-службы, OSGi, Flash и даже .NET.**

**Фреймворк Spring состоит из нескольких модулей. После загрузки и распаковки архива с дистрибутивом Spring Framework можно обнаружить 20 различных JAR-файлов.**

**Рисунок 1.3 – Функциональные категории, составляющие Spring**

**20 JAR-файлов, составляющих Spring, распределяются по шести функциональным категориям, как показано на рис. 1.3**

**В совокупности эти модули-категории предоставляют все необходимое для разработки корпоративных приложений.**

**Фактически фреймворк Spring имеет точки интеграции с некоторыми другими фреймворками и библиотеками, что устраняет необходимость писать их самостоятельно.**

**Центральное положение в фреймворке Spring занимает контейнер, управляющий процессом создания и настройки компонентов приложения. Этот модуль содержит фабрику компонентов, обеспечивающую внедрение зависимостей. На фабрике компонентов покоятся несколько реализаций контекста приложения Spring, каждый из которых предоставляет различные способы конфигурирования Spring.**

**В дополнение к фабрике компонентов и контексту приложения этот модуль также предоставляет несколько корпоративных служб,** **таких как электронная почта, доступ к JNDI, интеграция с EJB и выполнение заданий по расписанию.**

**Все модули Spring основаны на реализации основного контейнера. Эти классы неявно используются на этапе настройки приложения**

**Фреймворк Spring обеспечивает богатую поддержку аспектно-ориентированного программирования в своем модуле AOP . Данный модуль служит основой при разработке аспектов в приложении, построенном на основе Spring. Как и DI, AOP способствует ослаблению связей между прикладными объектами. Однако поддержка AOP позволяет отделять аспекты приложения (такие как транзакции и безопасность) от объектов, к которым они применяются.**

**Работа с JDBC зачастую сводится к обширному использованию шаблонного кода, который устанавливает соединение, создает SQL-запрос, обрабатывает результаты запроса, а затем закрывает соединение. Модуль поддержки JDBC объектов доступа к данным (DataAccess Objects, DAO) в Spring абстрагирует шаблонный код и позволяет сохранить простым и прозрачным программный код, реализующий операции с базами данных, а также предотвратить проблемы, возникающие в результате ошибки освобождения ресурсов.**

**Этот модуль также образует слой важных исключений, основанных на сообщениях об ошибках, посылаемых некоторыми серверами баз данных.**

**Фреймворк Spring предоставляет модуль ORM. Поддержка ORM в Spring основана на поддержке DAO, обеспечивающей удобный способ создания объектов доступа к данным для некоторых ORM-решений. Фреймворк Spring не пытается реализовать свое собственное ORM-решение, а просто предоставляет рычаги управления некоторыми популярными фреймворками ORM, включая Hibernate , Java Persistence API , Java Data Objects и iBATIS SQL.**

**В дополнение к поддержке создания пользовательского интерфейса в веб-приложениях этот модуль также предоставляет поддержку удаленных взаимодействий для создания приложений, взаимодействующих с другими приложениями. В состав средств удаленных взаимодействий в Spring входят механизм вызова удаленных методов (Remote Method Invocation , RMI), Hessian ,Burlap , JAX-WS и собственный механизм вызова через протокол HTTP.**

**Учитывая важность тестов, написанных разработчиками, в состав фреймворка Spring был включен модуль поддержки тестирования приложений на основе Spring.**

**Внутри этого модуля можно обнаружить коллекцию фиктивных объектов для применения в модульных тестах, проверяющих работу с JNDI, сервлетами и портлетами. Для нужд интеграционного тестирования этот модуль предлагает поддержку загрузки коллекций компонентов в контекст приложения Spring и выполнение операций с этими компонентами [4].**

# 1.2 AngularJS

**AngularJS — [JavaScript-фреймворк](https://ru.wikipedia.org/w/index.php?title=JavaScript-%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA&action=edit&redlink=1" \o "JavaScript-фреймворк (страница отсутствует)) с**[**открытым исходным кодом**](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)**. Предназначен для разработки**[**одностраничных приложений**](https://ru.wikipedia.org/wiki/Single_Page_Application)**. Его цель — расширение браузерных приложений на основе**[**MVC**](https://ru.wikipedia.org/wiki/Model-View-Controller)**шаблона, а также упрощение тестирования и разработки.**

**Фреймворк работает с**[**HTML**](https://ru.wikipedia.org/wiki/HTML)**, содержащим дополнительные пользовательские атрибуты, которые описываются директивами, и связывает ввод или вывод области страницы с моделью, представляющей собой обычные переменные JavaScript. Значения этих переменных задаются вручную или извлекаются из статических или динамических JSON-данных.**

**Angular не зависит от внешних библиотек и написан на чистом JavaScript. При этом, как правило, его рекомендуют использовать в связке с Bootstrap CSS. Так же Angular не ограничивает совместное использование с JQuery, например, для визуальных эффектов.**

**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.**

**Angular придерживается MVC-шаблона проектирования и поощряет слабую связь между представлением, данными и логикой компонентов. Используя**[**внедрение зависимости**](https://ru.wikipedia.org/wiki/%D0%92%D0%BD%D0%B5%D0%B4%D1%80%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B7%D0%B0%D0%B2%D0%B8%D1%81%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8)**, Angular переносит на клиентскую сторону такие классические серверные службы, как видозависимые контроллеры. Следовательно, уменьшается нагрузка на сервер и веб-приложение становится легче.**

**Базовой структурой в Angular является модуль. Модуль - это контейнер для взаимосвязанных функций. Поэтому создание приложения на Angular начинается с создания базового модуля, в котором необходимо указать имя приложения. За создание модуля отвечает следующий фрагмент кода:**

**var app = angular.module(‘AppName’,[ ]);**

**AppName — имя приложения. В квадратных скобках [ ] указываются зависимости на другие модули, необходимые для работы созданного модуля. Если их нет, указывается пустой массив.**

**Следующим этапом является создание контроллера. Контроллер необходим для отслеживания событий на странице. Создание контроллера реализуется с помощью следующего фрагмента кода:**

**app.controller(‘ControllerName’, function(){ });**

**Для того чтобы связать модуль и HTML-страницу используется атрибут ng-app:**

**<html ng-app=”AppName”>**

**Теперь Angular знает, какой модуль использовать внутри HTML. Так же необходимо подключить библиотеку AngularJS:**

**<script type="text/javascript" src="angular.min.js"> </script>**

**Для того чтобы передать на HTML-страницу значения с сервера, необходимо внутри HTML в фигурных скобках {{}} указать переменную, в которую AngularJS контроллер запишет значение. Так же внутри {{}} мы можем указать не только переменную, но и выражение. Например:**

**{{'Цена: '+store.data.price+'$'}}.**

**Директива - специальный атрибут, который используется непосредственно внутри HTML тегов и обрабатывается Angular в соответствии с определенной для данной директивы логикой. Например, внутри блока, где нам нужно отображать данные, мы указываем имя контролера, который будет нам эти самые данные предоставлять. Делается это при помощи встроенной директивы ng-controller:**

**<div ng-controller=”ControllerName”>**

**Другими часто используемыми встроенными директивами являются:**

* **ng-show — если ее значение является False, то HTML элемент (используя display:none);**
* **ng-hide — обратный аналог ng-show. То есть скрывает элемент если его значение равно True;**
* **ng-repeat — аналогичен по смыслу функции foreach. То есть циклично повторяет включенные в блок HTML теги.**

**Двустороннее связывание данных в AngularJS является наиболее примечательной особенностью и уменьшает количество кода, освобождая сервер от работы с шаблонами. Вместо этого, шаблоны отображаются как обычный HTML, наполненный данными, содержащимися в области видимости, определенной в модели. Сервис $scope в Angular следит за изменениями в модели и изменяет раздел HTML-выражения в представлении через контроллер. Кроме того, любые изменения в представлении отражаются в модели. Это позволяет обойти необходимость манипулирования DOM и облегчает инициализацию и прототипирование веб-приложений.**

**Особенностью Angular является возможность создания собственных директив. Другими словами, можно создавать собственные HTML-теги, которые могут иметь собственную логику и заменять собой целые куски HTML. Это позволяет писать выразительный код, который полностью передает структуру нашего приложения. И конечно, это позволяет сильно улучшить повторное использование кода.**

**Что касается повторного использования фрагментов HTML кода, Angular имеет в своем распоряжении директиву ng-include, которая позволяет вынести часть кода в отдельный файл. Выглядит это следующим образом:**

**<h3 ng-include="'product-title.html'"></h3>**

**Кавычки вокруг имени указывают Angular, что это строка с именем, а не переменная, содержащая имя файла.**

**Зависимости – это модули, которые необходимы для обеспечения функциональности создаваемого нами нового модуля. Их так же можно рассматривать как способ улучшить структуру приложения, распределив самостоятельные части логики между различными модулями [2].**

# 1.3 Nexmo

Nexmo предоставляет инновационные интерфейсы связи, который преодолели традиционные голосовые и текстовые у**слуги облачных коммуникаций. Nexmo позволяет приложениям с легкостью делать телефонные звонки и отправлять или получать текстовые сообщения, независимо от того, в какой точке земного шара находятся клиенты. Для каждой страны Nexmo предоставляет свой перечень цен за услуги. Так для Беларуси стоимость одного исходящего SMS сообщения стоит €0.055, одна минута исходящего вызова на мобильный телефон – €0.3139, а на городскую линию – €0.2235 за минуту.**

**Из преимуществ Nexmo, по сравнению с аналогичными сервисами, выделяют:**

* **нет первоначальных контрактов или гонораров;**
* **не фиксированы сроки пополнения счета;**
* **посекундная тарификация голосовых звонков;**
* **входящие SMS бесплатные.**

**Благодаря тому, что сервис постоянно совершенствуется, можно отметить ряд аспектов, которые выгодно отличают Nexmo от обычных телефонных звонков:**

* **беспрецедентная исполнимость в более чем 200 странах**
* **время ожидания любого сообщения меньше**
* **возможность анализировать статистику звонков и сообщений**
* **сохранность денег благодаря отсутствию посредников в осуществлении связи.**

**Для того чтобы воспользоваться услугами Nexmo, необходимо иметь:**

* **аккаунт на сайте nexmo.com**
* **веб-браузер**
* **номер мобильного телефона, который будет значиться в качестве отправителя**
* **номер мобильного телефона, который будет значиться в качестве получателя**

**Стоит обратить внимание, что номера телефонов, которые используются в сервисе Nexmo, не должны содержать пробелов, ведущих нулей и знака «+». На рисунке 1.4 изображен пример номеров телефонов, которые удовлетворяют всем правилам сервиса.**

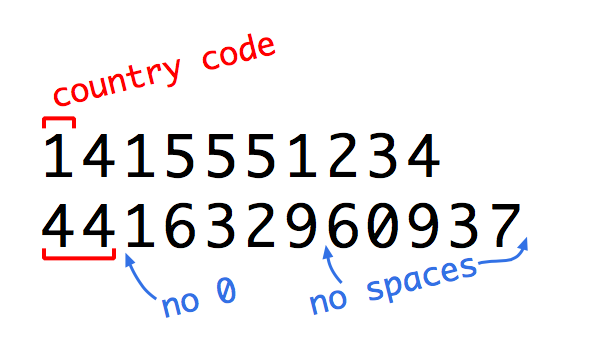


Рисунок 1.4 – Пример допустимого номера телефона для сервиса Nexmo

При регистрации на сайте nexmo.com пользователь получает на счет 2$, которые может потратить на тестовые звонки и сообщения. Стоит отметить, что при использовании этих 2$ для отправки SMS, текст сообщения будет содержать запись [NEXMO DEMO], которая изображена на рисунке 1.5. После пополнения счета, этой надписи не будет.

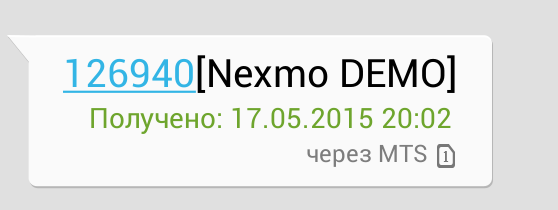


Рисунок 1.5 – Демонстрация сообщения, полученного через сервис Nexmo

Все запросы к Nexmo должны быть представлены с помощью базовой URL:

<https://rest.nexmp.com/sms/xml>

<https://rest.nexmp.com/sms/json>

Nexmo предоставляет возможность выбрать ответ в виде объекта JSON или XML строки. Все запросы подаются через HTTP POST или GET метод с использование UTF-8 кодировки. Для обеспечения конфиденциальности рекомендуется использование HTTPS для все запросов Nexmo API.

Пример SMS сообщения, послаемого с помощью сервиса Nexmo:

curl "https://rest.nexmo.com/sms/json?api\_key={api\_key}

&api\_secret={api\_secret}&from=MyCompany20&to=447525856424

&text=D%c3%a9j%c3%a0+vu"

Nexmo используется Unicode для поддержки нескольких языков. Однако, длина текста ограничена 70 символами. При превышении этого объема символов сообщение будет разиваться на части. Кроме того, мобильное устройство должно поддерживать кодровку получаемого сообщения. Так, например, американские устройства могут не поддерживать арабский.

После отправки запроса на API Nexmo, вы получите HTTP ответ. Ответ будет содержать либо объект JSON, либо XML строку, в зависимости от базового URL, который вы выбрали, чтобы отправить запрос. И будет указывать на успех или неудачу своего запроса на платформе Nexmo.

Ниже представлен пример ответа, пришедший в виде XML строки.

<?xml version='1.0' encoding='UTF-8' ?>  
<mt-submission-response>  
  <messages count='x'>  
   <message>  
    <messageId>${messageId}</messageId>  
    <to>${to}</to>  
    <clientRef>${client-ref}</clientRef>  
    <status>${returnCode}</status>  
    <errorText>${error-message}</errorText>  
    <remainingBalance>${account-balance}</remainingBalance>  
    <messagePrice>${message-price}</messagePrice>  
    <network>${network}</network>  
   </message>  
  </messages>  
</mt-submission-response>

Объект messages содержит в себе части исходного сообщения, на которые разбил сервис Nexmo. Каждая такая часть содержит объект status, который представляет собой код, с помощью которого можно понять, успешно ли дошло сообщение. Если же сообщение не дошло, то этот код поможет узнать причину неудачи[5].

# 1.4 Google Maps

Карты Google – набор приложений, построенных на основе бесплатного картографического сервиса и технологии, представляемых компанией Google. Сервис представляет собой карту и спутниковые снимки планеты Земля, которые изображены на рисунке 1.6. Для многих регионов доступны высокодетальные аэрофотоснимки, а для некоторых – под углом 45° с 4 сторон света. С сервисом интегрирован бизнес-справочник и карта автомобильных дорог, с поиском маршрутов.



Рисунок 1.6 – Спутниковый снимок планеты Земля

Google Street View позволяет пользователям Google Maps посмотреть трехмерную проекцию города или некоторых его улиц через интернет. Такая функциональность достигается при помощи кругового фотографирования реальной местности специальным оборудованием в режиме реального времени. В итоге создается множество стререосфреческих панорам с привязкой к географическим координатам и предоставляется возможность переключаться между ними, используя интуитивно понятный интерфейс, создавая ощущение перемещения в пространстве[6].

Существует возможность использования сервиса Google Maps в качестве основы для сторонних сервисов. Используя Google Maps API, возможно включить любую карту из Google Maps на вешнем сайте, управляя этой картой через JavaScript, например, для добавления маркеров географических точек, приближения или удаления при просмотре карт.

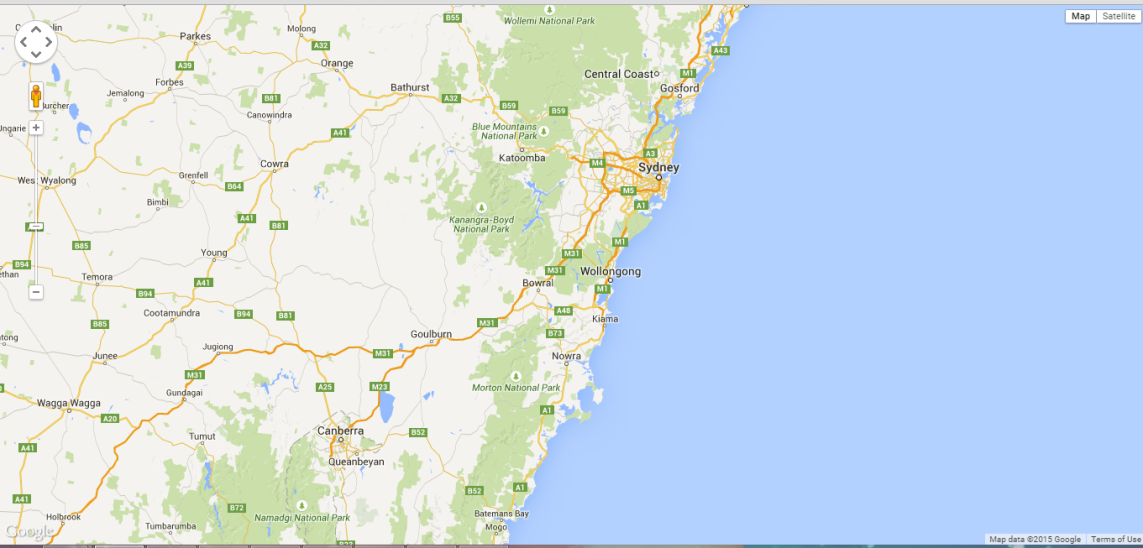
**Для того чтобы загрузить карту Google Maps на страницу, необходимо выполнить следующие действия:**

* **Декларировать страницу как HTML5, используя директиву <!DOCTYPE html>**
* **Подключить Maps API JavaScript, используя тег script**
* **Создать div элемент с именем “map-canvas”, который будет содержать карту**
* **Создать JavaScript объект, содержащий свойства карты**
* **Создать JavaScript объект map, поместить его в div элемент**

**Таким образом, страница имеет следующий вид:**

**<!DOCTYPE html>  
<html>  
  <head>  
    <style type="text/css">  
      html, body, #map-canvas { height: 100%; margin: 0; padding: 0;}  
    </style>  
    <script type="text/javascript"  
      src="https://maps.googleapis.com/maps/api/js?key=API\_KEY">  
    </script>  
    <script type="text/javascript">  
      function initialize() {  
        var mapOptions = {  
          center: { lat: -34.397, lng: 150.644},  
          zoom: 8  
        };  
        var map = new google.maps.Map(document.getElementById('map-canvas'),  
            mapOptions);  
      }  
      google.maps.event.addDomListener(window, 'load', initialize);  
    </script>  
  </head>  
  <body>  
<div id="map-canvas"></div>  
  </body>  
</html>**

**На рисунке 1.7 представлено отображение страницы в браузере**



**Рисунок 1.7 – HTML страница с Google картой.**

1. ****ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ****

# ****2.1 Структура проекта****

Разрабатываемое приложение предназначено для управления цветочными заказами. Для хранения информации будет использоваться база данных MySQL, которая будет содержать таблицы, изображенные на рисунке 2.1:

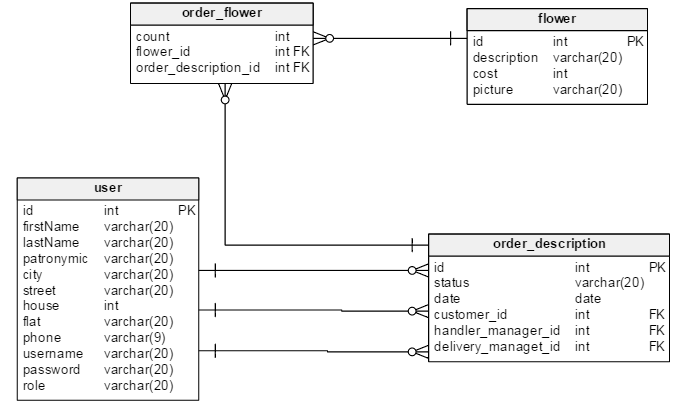
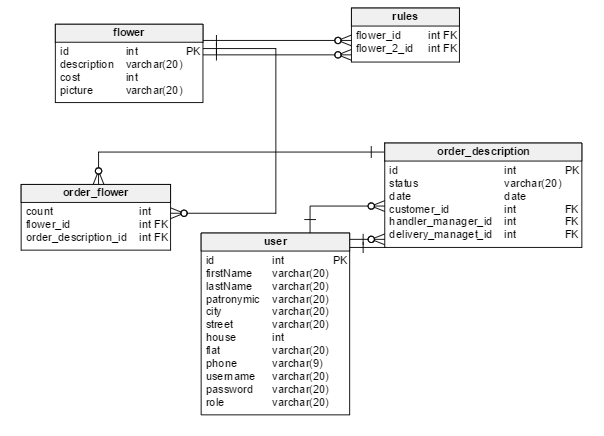


Рисунок 2.1 – Структура базы данных

Работа над созданием приложения начинается с создания каркаса приложения. Каркас включает в себя следующие этапы:

1. Подключение Hibernate и настройка соответствующих параметров [3].

Это демонстрирует следующий фрагмент кода:

<session-factory>

<property name="hibernate.connection.driver\_class"> com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.url"> jdbc:mysql://localhost:3306/course\_work</property>

<property name="hibernate.connection.password"> 1234567 </property>

<property name="hibernate.connection.username"> root</property>

<property name="hibernate.dialect"> org.hibernate.dialect.MySQLDialect</property>

<property name="show\_sql">true</property>

<property name="hbm2ddl.auto">create</property>

<mapping class="bsu.model.Flower" />

<mapping class="bsu.model.Order" />

<mapping class="bsu.model.User" />

<mapping class="bsu.model.Rules" />

<mapping class="bsu.model.OrderElement" />

</session-factory>

Для работы с информацией, хранимой в базе данных, в приложении должны быть классы-сущности, которые отображаются на таблицы. Также необходима настройка классов модели для отображения в базу данных. Это возможно благодаря аннотациям:

@Entity

@Table(name = "user")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column(columnDefinition = "INT unsigned")

private Long id;

@Column(nullable = false)

private String firstName;

}

1. Настройка Spring Security.

Приложения включает несколько ролей пользователей: администратор, менеджер обработки заказа, менеджер доставки заказа и пользователь. Для того чтобы обезопасить действия пользователя и сервер от несанкционированного доступа, используется Spring Security. Следующий фрагмент кода демонстрирует работу Spring Security:

<http auto-config="true" use-expressions="true">

<access-denied-handler error-page="/#/login" />

<form-login

login-page="/#/login"

authentication-failure-url="/#/login?error"

default-target-url="/#/login?success"

always-use-default-target="true"/>

<logout invalidate-session="true"

logout-success-url="/#/login?logout"

delete-cookies="JSESSIONID, userInfo"/>

</http>

<authentication-manager>

<authentication-provider user-service-ref="customUserDetailsService">

</authentication-provider>

</authentication-manager>

1. Настройка Spring.

Spring настройка – конфигурационные файлы. Использование фреймворка Spring:

<context:component-scan base-package="bsu.service.impl"/>

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="com.mysql.jdbc.Driver"/>

<property name="url" value="jdbc:mysql://localhost:3306/course\_work"/>

<property name="username" value="root"/>

<property name="password" value="pavlovamarisha"/>

</bean>

<bean id="entityManagerFactory"

class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">

<property name="dataSource" ref="dataSource"/>

<property name="persistenceUnitName" value="myPersistenceUnit"/>

<property name="packagesToScan" value="bsu.model"/>

<property name="jpaVendorAdapter">

<bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">

<property name="databasePlatform" value="org.hibernate.dialect.MySQL5InnoDBDialect"/>

<property name="showSql" value="false"/>

<property name="generateDdl" value="true"/>

</bean>

</property>

<property name="persistenceProvider">

<bean class="org.hibernate.jpa.HibernatePersistenceProvider"></bean>

</property>

</bean>

<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">

<property name="entityManagerFactory" ref="entityManagerFactory"/>

</bean>

<tx:annotation-driven transaction-manager="transactionManager"/>

<jpa:repositories base-package="bsu.repository"/>

<bean id="customUserDetailsService"

class="bsu.service.impl.CustomUserDetailService">

</bean>

<task:annotation-driven />

1. Настройка AngularJS.

Настройка AngularJS включает в себя создание статической страницы:

<html>

<head> </head>

<body ng-app="OrderFlower">

<header>

</header>

<div class="content" id="content" ng-view></div>

</body>

</html>

А также создание модуля и контроллера:

var app = angular.module("OrderFlower", ['ngRoute, 'ngCookies']);

app.controller("authorizationController", function ($scope, $http, $location, $rootScope, $cookieStore) {});

1. Настройка Nexmo.

Для того чтобы использовать сервис Nexmo в Java, достаточно создать клиента, создать сообщение и указать клиенту послать сообщение. В программе это выглядит следующим образом:

NexmoSmsClient client = null;  
try {  
 client = new NexmoSmsClient(*API\_KEY*, *API\_SECRET*);  
} catch (Exception e) {  
 e.printStackTrace();  
}  
Random rand = new Random();  
int max = 999999;  
int min = 100000;  
int randomNum = rand.nextInt((max - min) + 1) + min;  
TextMessage message = new TextMessage(*SMS\_FROM*, phone, new Integer(randomNum).toString());  
SmsSubmissionResult[] results = null;  
try {  
 results = client.submitMessage(message);  
} catch (Exception e) {  
 e.printStackTrace();  
}

1. Подключение Google Maps.

Для использования карт необходимо указать на html место, в которое будет помещена карта:

<div id="map" style="height: 350px;"></div>  
<div id="class" ng-repeat="marker in markers | orderBy : 'title'">  
 <a href="#" ng-click="openInfoWindow($event, marker)">{{marker.title}}</a>  
</div>

И с помощью JavaScript инициализировать карту:

var mapOptions = {  
 zoom: 10,  
 center: new google.maps.LatLng(53.55, 27.33),  
 mapTypeId: google.maps.MapTypeId.TERRAIN  
};  
  
$scope.map = new google.maps.Map(document.getElementById('map'), mapOptions);  
$scope.markers = [];

var marker = new google.maps.Marker({  
 map: $scope.map,  
 position: new google.maps.LatLng(53, 27)  
});  
$scope.markers.push(marker);

# ****2.2 Технические возможности созданного приложения****

Созданное приложение для заказа цветов позволяет осуществлять следующие действия:

* Просматривать информацию о цветах и букетах, вводить необходимое количество экземпляров и делать заказ (рисунок 2.2)
* Создавать собственный вариант букета, путем указания наименования цветка и количества экземпляров. Перед оформлением заказа может быть показано сообщение о том, что некоторые из выбранных для букета цветов не сочетаются друг с другом. Это сообщение не препятствует оформлению заказа, а лишь рекомендует вам изменить свой выбор. Данный случай представлен на рисунке 2.3.
* Запрет на оформление заказа для неавторизированного пользователя
* Возможность создания нового пользователя с ролью «Пользователь». Клиенту необходимо заполнить информацию о себе, ввести адрес проживания, который будет отображен на карте, добавить номер телефона и информацию для авторизации (рисунок 2.4). Для того чтобы убедиться, что информацию вводит реальный человек, добавлена функция проверки по номеру телефон. Идея этой функции в том, что на указанный номер посылается шестизначный код, который пользователь должен ввести в окошко, представленное на рисунке 2.5
* При авторизации клиента, имеющего роль «Менеджер», появляется возможность просматривать никем не занятые заказы и заказы, ответственным за которые является авторизованный клиент. Данная возможность изображена на рисунке 2.6.

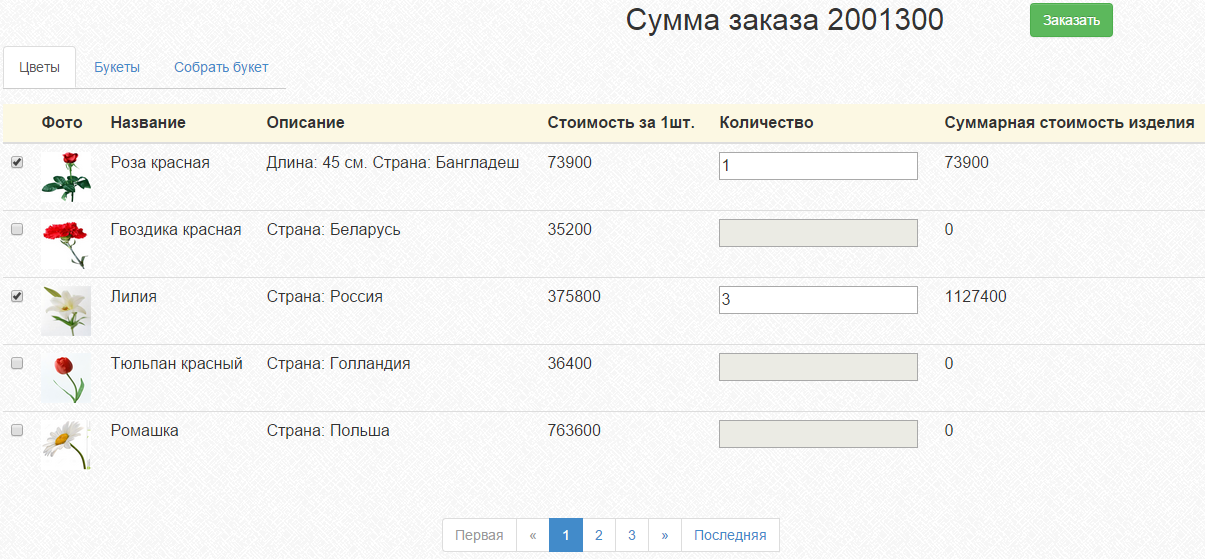


Рисунок 2.2 – Просмотр информации о цветах. Указание необходимого числа экземпляров цветка для заказа

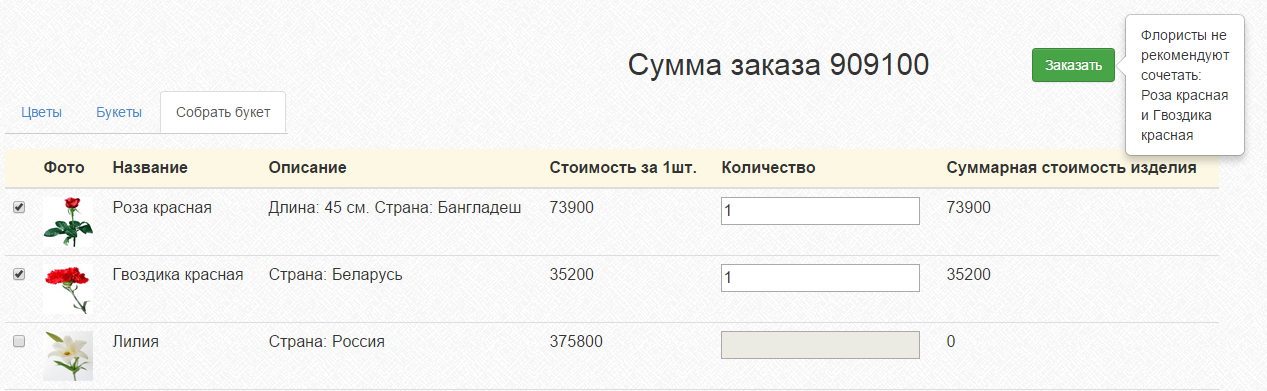
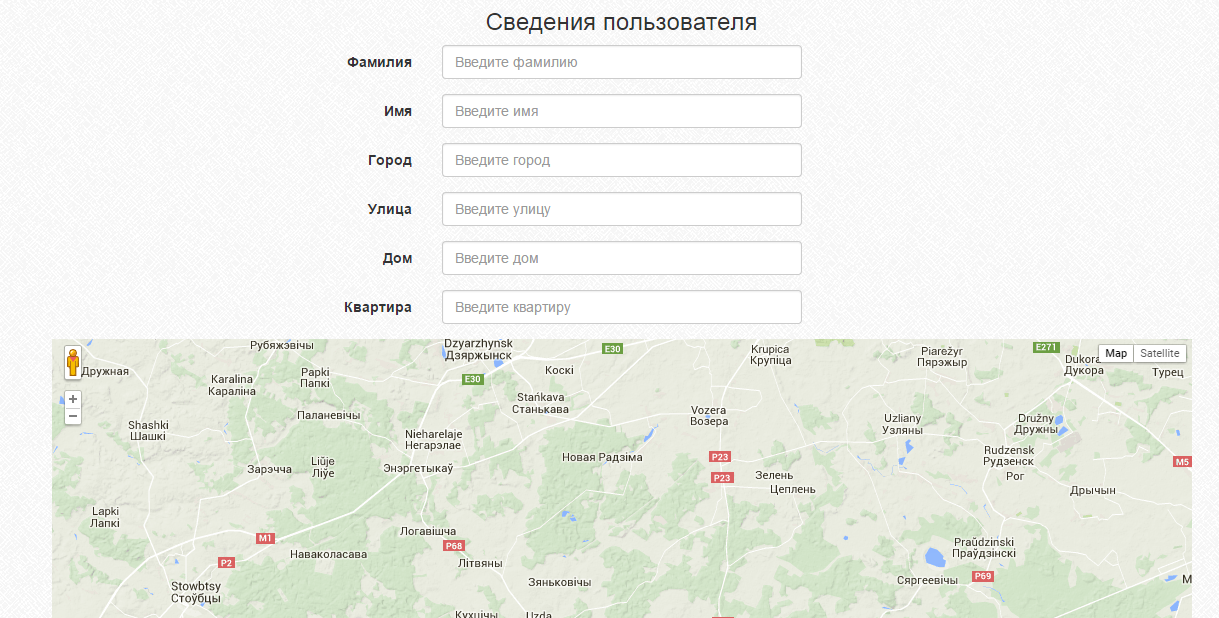
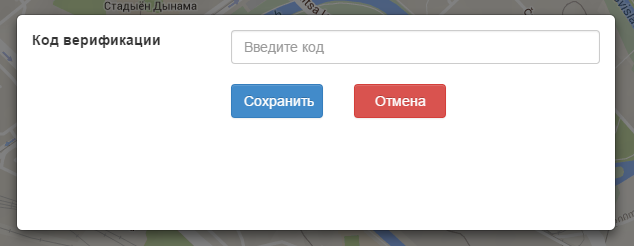


Рисунок 2.3 – Сообщение о не сочетаемости цветов



2.4 – Форма для заполнения при создании нового клиента



2.5 – Форма для ввода кода верификации

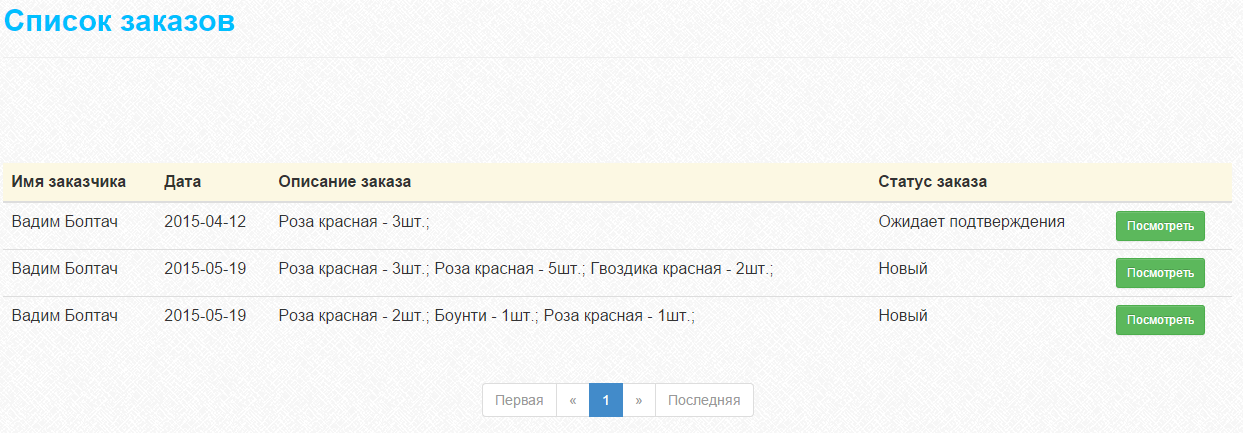


Рисунок 2.6 – Список заказов для авторизированного клиента

* Авторизированный с ролью «Менеджер» клиент имеет возможность просматривать список пользователей (рисунок 2.7) и смотреть информацию о них (рисунок 2.8). Также он обладает возможностью создавать пользователя с ролью «Менеджер». Менеджер может изменять статус заказа (рисунок 2.9).
* Каждый авторизированный пользователь может просматривать свои заказы. Но редактировать статус заказа клиент с ролью «Пользователь» не может. Этот вариант представлен на рисунке 2.10.

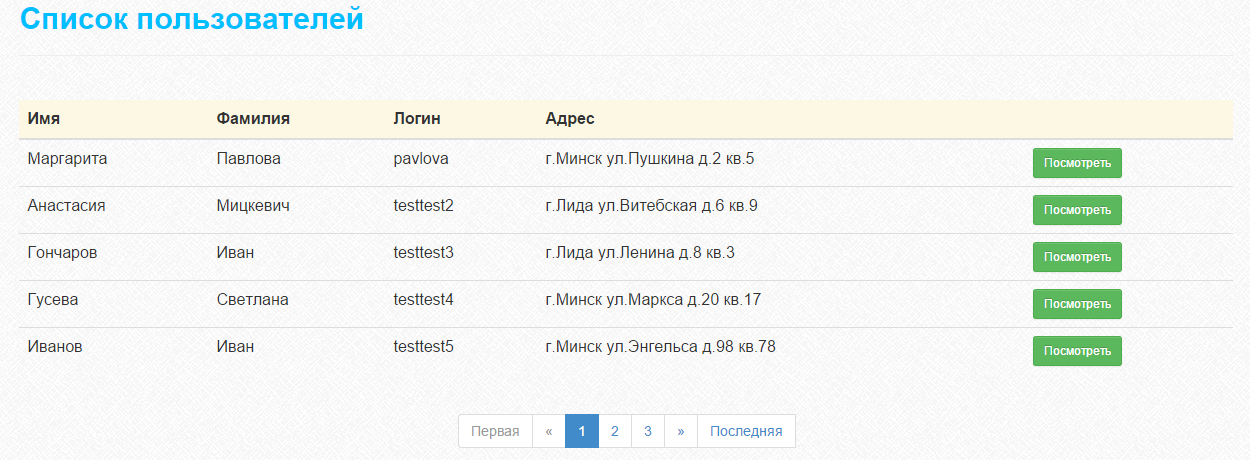


Рисунок 2.7 – Список пользователей

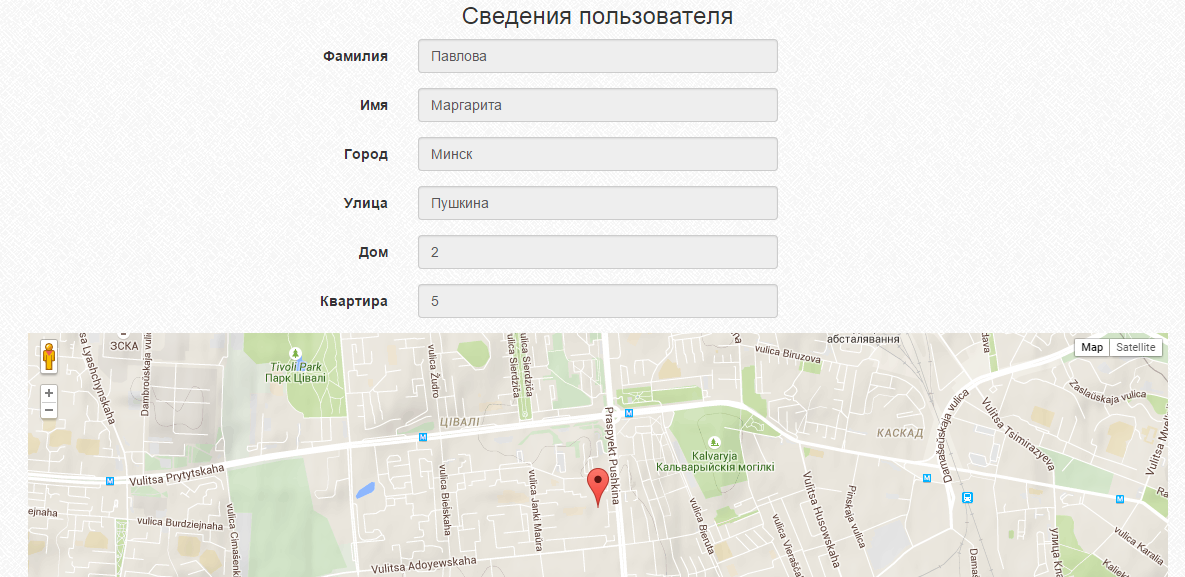


Рисунок 2.8 – Информация о пользователе

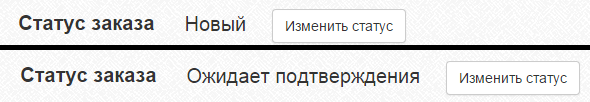


Рисунок 2.9 – Изменение статуса заказа

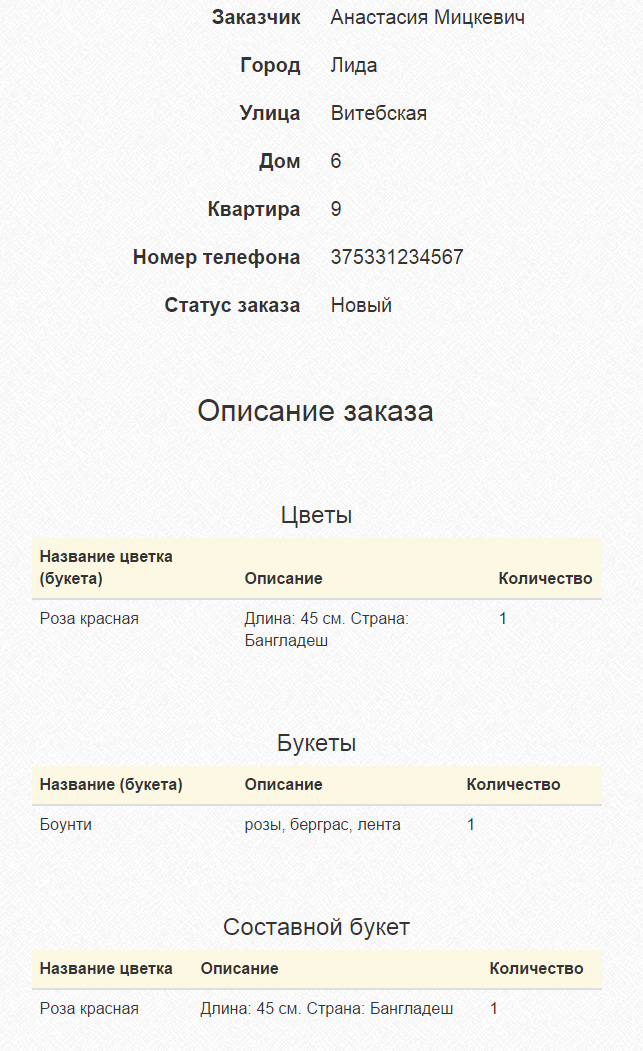


Рисунок 2.10 – Просмотр информации о заказе

Для каждого авторизированного пользователя представлена своя панель навигации (рисунок 2.11 и рисунок 2.12). Но для каждого пользователя в правой верхней части панели отображается его роль и логин, нажав на который можно редактировать свой профиль.



Рисунок 2.11 – Навигационная панель для клиента, авторизированного с ролью «Менеджер»



Рисунок 2.12 – Навигационная панель для клиента, авторизированного с ролью «Пользователь»

# ЗАКЛЮЧЕНИЕ

В результате проведенной работы были изучены методы создания WEB-приложений. Вследствие чего для проектирования WEB-приложения были выбраны технологии: фремворки Sping и AngularJS. Также были добавлены сервисы Nexmo и Google Maps.

В процессе проектирования WEB-приложения был реализован следующий набор функционала:

* Просмотр ассортимента цветов и букетов, составление собственного букета
* Вход в приложение
* Создание нового пользователя с разными ролями
* Проверка клиента на подлинность
* Оформление заказа
* Продвижение заказа от одного менеджера к другому путем изменения статуса заказа
* Просмотр личных заказов
* Просмотр доступных заказов в зависимости от роли пользователя
* Изменение собственного профиля
* Возможность просмотра профилей пользователей для менеджера
* Указание местоположения на карте

Используемый сервис Nexmo позволяет проводить проверку клиента на стадии создания аккаунта. Это позволяет выявить людей, которые ввели неверные данные и которые впоследствии могли бы нанести вред приложению. Сервис Google Maps позволяет курьерам с помощью карт узнавать местоположение клиента, что позволяет ускорить доставку цветов.

Реализованное приложение является актуальным решением проблемы возрастающего интереса со стороны пользователей к быстрому оформлению заказов на покупку цветов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Spring [Electronic resource]. – Mode of access : <http://www.spring.io>. – Date of access : 15.10.2014.
2. AngularJS [Electronic resource] / AngularJS – Superheroic JavaScript MVW Framework. – Mode of access : <https://www.angularjs.com>. – Date of access : 9.11.2014.
3. Hibernate [Electronic resource] / Hibernate. Everything data. - Hibernate. – Mode of access : <http://hibernate.org/>. – Date of access : 13.10.2014.
4. Walls C. Spring in action / C. Walls. – 3 edition. – New York : Manning Publications Co., 2011. – 400 p.
5. Nexmo [Electronic resource] / Nexmo – APIs for SMS, Voice and Phone Verification. - Nexmo. – Mode of access : <http://nexmo.org/>. – Date of access : 21.03.2015.
6. Google Maps [Electronic resource] / Google Maps API – Google Developers . – Google Maps. – Mode of access : <https://developers.google.com/maps>. – Date of access : 25.03.2015.