# ВВЕДЕНИЕ

Еще несколько десятков лет назад вычислительная техника занимала целые комнаты, на данный момент технология изготовления их модернизировалась и в целом в корне изменила темп и образ жизни общества. Процесс компьютеризации необратим, а развитие информационных систем невозможно остановить ввиду их теоретической и практической значимости во всех отраслях народного хозяйства.

Согласно обзору инвестиционного портала, инвестиции в сферу IT-технологий являются на данный момент наиболее выгодными. И в ближайшем будущем ситуация вероятней всего не изменится, а IT направления будут наиболее популярными, ведь каждый современный человек имеет косвенное или прямое отношение к вычислительным устройствам и сети Интернет. В связи с этим другие сферы жизнедеятельности людей будут вынуждены подстраиваться к новшествам, принимая при этом возможно иные формы, чем сейчас.

На современном этапе развития интернет-технологий все наибольшую популярность приобретает электронный бизнес. Современные интернет-технологии, которые реализуются в области экономики, предоставляют организациям большие возможности в организации, ведении и поддержании электронной торгово-посреднической деятельности. Средой для реализации данной коммерции выступает сеть Internet, в которой и реализуются все проекты.

Большинство людей полагают, что электронная коммерция – это покупка товаров в режиме онлайн. Обычно это называется электронной коммерцией – традиционная розничная торговля "магазинного" типа. К числу современных и крупнейших представителей такого рода электронной коммерции в сети Internet относятся сайты:

* Amazon (<http://amazon.com>)
* Ebay (<http://ebay.com>)
* Aliexpress.com (<http://aliexpress.com>)

Интернет стал доступнее, быстрее и дешевле. Сейчас уже никого не удивишь выделенной линией или безлимитным тарифом. В таких условиях удобнее открыть страницу сайта, чем покупать газету. Однако статистика показывает, что максимум активности пользователей приходится на рабочее время – многие используют доступ в Интернет на работе. Действительно, работу секретаря или бухгалтера сейчас невозможно представить без компьютера, а компьютер – без доступа в сеть.

Исходя из этого объемы продаж через Internet уже приблизились к традиционным розничным продажам. Такие сайты уже сейчас занимаю важную область рынка и, несомненно, будут продолжать расти.

Целью данного проекта является разработка интернет приложения ведения электронной коммерции.

Задачи проекта:

* реализовать возможность оформления и добавление товара\услуг в корзину;
* реализовать возможность регистрации и авторизации пользователя с разделением прав доступа;
* реализовать возможность отображения новостей;
* реализовать возможность отображение самых популярных заказываемых товаров или услуг.

# СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

# Анализ предметной области

Использование всемирной паутины для ведения электронной коммерции является сегодня наиболее актуальным и востребованным способом. Связано это с удобством совершения данных мероприятий. Для достижения цели, гражданам нужно лишь иметь персональный компьютер или смартфон с браузером и выходом в интернет. Просматривать имеющиеся сведения вполне реально в любое время дня и ночи. При этом, совершенно не важно конечное местоположение пользователя.

Тема дипломного проекта «Интернет приложение ведения электронной коммерции».

Основной целью данного проекта является разработать приложение облегчающее покупку любых товаров и услуг. Просмотр популярных товаров или услуг которые автоматически формируются исходя из заказов пользователей.

Дипломный проект будет представлять собой веб-приложение, предназначенное для просмотра товаров или услуг, добавления в корзину, просмотра новостей компании, оформления заказов. Со стороны администратора будет предусмотрена возможность манипулирования данными в базе данных с помощью MYSQL Workbench.

Аналогами данного проекта являются следующие ресурсы:

* Deal.by – орговый центр в интернете. Потребительские, промышленные и оптовые товары. Все для вашего бизнеса, быта и отдыха!
* Belassist.by – организация приема платежей по банковским картам Белкарт, Visa, MasterCard, Visa Electron, Maestro, American Express на сайтах интернет-магазинов.

Обозначим основные части дипломного проекта.

Первоначально необходимо разработать базу данных. После сайт, для чего нужно выполнить следующие этапы создания веб-сайта:

1. Определение целей веб-сайта и его позиционирование.

На этом этапе необходимо определить, для чего нужен сайт, т.е. какие задачи он должен решать: предоставить общее представление о товаре и услуге, или многосторонне осветить какую-либо сторону человеческой деятельности, увеличить продажи по традиционным каналам или организовать веб-торговлю, провести рекламную или маркетинговую кампанию.

Цели веб-сайта, в большинстве случаев, должны ставиться заказчиком, а затем, вместе с исполнителем они уточняются и корректируются.

Это один из самых важных этапов не только создания веб-сайта как такового, но важнейший этап интернет-маркетинга.

После определения целей сайта надо совершенно четко и как можно подробнее представить и описать целевую аудиторию сайта, т.к. это влияет на то, в каком виде будет представлена информация.

Определение целевой аудитории веб-сайта – это не менее важный, чем определение целей веб-сайта, этап. Не всегда целевая аудитория компании из офлайн будет прямо проецироваться на он-лайн. Зная целевую аудиторию и аудиторию части Интернета можно сделать предположения о том, кто будет являться основными посетителями веб-сайта.

Определение и как можно более подробное описание целевой аудитории сайта дает возможность разработать правильный дизайн для проекта, а также выбрать правильное направление для написания текстов.

На заключительной стадии этого этапа примерно определяют, по каким поисковым запросам сайт должен появляться в результатах поискового запроса и посмотреть интернет-проекты конкурентов будущего веб-проекта.

Дополнительную информацию о положении вещей в Интернете на вашем рынке даст обзор сайтов конкуретнов. Необходимо иметь в виду, что совсем необязательно конкуренты из реальной жизни будут являться таковыми в Интернете. Нужно просмотреть некоторое количество сайтов, которые расположены в ТОП 10 ведущих поисковых машин, обратите внимание на графическое решение, на тексты, на поисковые запросы. Это даст достаточно пищи для размышления, каким должен быть сайт вашей компании.

1. Создание Технического Задания (далее ТЗ) на разработку веб-сайта.

В ТЗ необходимо как можно более подробно описать:

* цели создания сайта и его целевую аудиторию;
* структуру веб-сайта и количество страниц в каждом разделе;
* работу динамических модулей;
* пожелания по дизайну (цвета, использоание фирменного стиля, соотношение графика/текст и т.д.;
* используемые технологии (Java 8, Spring Boot, HTML5, Spring JPA, Bootstrap, Thymeleaf, hibernate);
* порядок предоставления, обработки или создания графической и текстовой информации;
* технические требования к сайту.

ТЗ является основным документом, на основе которого осуществляются все последующие этапы разработки веб-сайта.

1. Создание дизайн-макета веб-сайта.

На этом этапе дизайнер в специальной графической программе создает дизайн страниц будущего веб-сайта с прорисовкой всех графических (баннеров, кнопок, фотографий) и текстовых элементов. Дизайнер создает дизайн веб-страниц с учетом пожеланий заказчика и задания, прописанного в ТЗ.

Современные мониторы имеют различные разрешения. В разговоре о веб-сайтах нас особенно интересует ширина. Размер монитора по ширине в пикселях может быть 800, 1024, 1280, 1600 и даже больше. Более того, пользователь может просматривать сайт в полускрученном окне или на смартфоне.

Фиксированный дизайн сайта предполагает одну единую ширину сайта для всех разрешений экрана и для всех ширин окон, т.е. если ширина сайта 1000 пикселейй, то на экране 800 пикселей будет появляться горизонтальная прокрутка, а на экране 1280 пикселей – пустые поля слева и справа (или только с одной стороны)

Такое построение сайта позволяет четко управлять композицией сайта и однозначно знать, где будет каждый элемент навигации на каждом экране. Этот тип сайта позволяет использовать более насыщенные графические элементы.

Резиновый дизайн, в отличие от фиксированного «подстраивается» под ширину экрана.

Это приводит к тому, что сайт без горизонтальной прокрутки увидит большинство пользователей, но на разных экранах композиция сайта будет непредсказуемой.

Данный сайт адаптивный, поэтому будет удобен на всех видах устройств.

1. Верстка сайта.

После того, как был утвержден дизайн – за работу принимается верстальщик – это тот человек, который переводит дизайн-макет на язык, понятный компьютеру с использованием языка HTML5 и фреймворка Bootstrap.

1. Программирование сайта.

Очень часто этап программирования и верстки объединяют в один. На мелких и средних проектах оба действия в состоянии выполнить один человек. На крупных проектах в силу специфики работ эти этапы разделяют.

На этапе программирования (как правило, с использовавнием HTML, Java, Thymeleas, Spring framework и баз данных) происходит создание всех страниц сайта, определяется порядок работы меню, расставляются гиперссылки, создается динамика на сайте, программируются такие составляющие, как гостевая книга, форум, новостная лента и проч.

Если сайт должен иметь администраторский интерфейс, то он создается именно на этапе программирования.

Очень важно на этапе программирования определить, на какой системе администрирования (Content Management Site) будет работать ваш сайт. На сегодня на рынке существует великое множество систем администрирования.

1. Наполнение сайта информацией.

На этом этапе информация размещается на сайте, т.е. путем перевода в специальный формат текст и графика располагаются на сайте на определенных страницах, и эта информация становится доступной для просмотра.

1. Тестирование сайта.

Этот этап можно осуществить как до, так и после размещения сайта по его «родному» адресу. На этом этапе выявляются все ошибки и недочеты в программировании и написании текстов.

После разработки сайта его нужно разместить в сети интернет.

Для этого он должен иметь домен и хостинг.

Доменное имя или домен – символическое имя. под которым сайт будет известен в сети. Получить его можно либо, купив у регистратора, бесплатно, но в виде поддомена от одного из бесплатных хостингов.

Хостинг – услуга размещения сайта на серверах, имеющих выход в Интернет В настоящее время компаний, предоставляющих эту услугу – огромное количество. Остается только выбрать подходящую. К критериям выбора относятся:

1. Цена есть как платные, так и бесплатные хостинги.
2. Объем предоставляемого графика и дискового пространства.

Поддержка тех или иных средств (Java, Tomcat, MySQL, возможность развертывания CMS. наличие дополнительных скриптов и т.д.).

Для данного проекта был выбран хостинг ActiveCloud был основан в 2003 году. Уже 14-ый год компания предлагает все современные облачные решения и хостинг, осуществляя поддержку более 45000 [клиентов](http://www.active.by/ru-by/company/clients/) в Беларуси и 6 странах СНГ. ActiveCloud входит в группу [Softline](http://softlinegroup.com/) – ведущую международную компанию в сфере лицензирования программного обеспечения, оказывающую полный спектр IT-услуг.

Далее необходимо разместить все файлы сайта на сервере у хостера. Как правило, предоставляется несколько вариантов для этого, а именно:

1. Обычно это загрузка содержимого по НТТР-лрогоколу, с использованием панели управления хостера или пo FTP с помощью FTP- клиента.
2. Второй вариант намного быстрее и проще. Одним из лучших FTP- клиентов является встроенный в Total Commander и бесплатный FileZilla.

Как только устанавливается связь с сервером FTP (хостер обязательно сообщит адрес, логин и пароль входа), пространство сервера станет доступным в виде логического устройства (как и обычные диски компьютера) на одной из двух панелей. После чего необходимо запустить процесс копирования и ждать его окончания. По окончании процесса копирования сайт будет опубликован в Интернете.

Для привлечения клиента на свой сайт существует большое количество способов.

Поисковые системы – по статистике большой популярностью пользуются те сайты, которые находятся в верхних строках поисковой выдачи. На первые 3-5 сайтов приходится 80% посетителей – именно поэтому все сайтовладельцы так жаждут быть в поисковых системах как можно выше.

Основные методы для привлечения посетителей:

1. Контекстная реклама – один из самых эффективных платных методов привлечения целевой аудитории – контекстная реклама. Данный вид рекламы работает через крупные поисковые системы (Яндекс, Google). Ее суть – появление рекламного объявления только тогда, когда пользователь вводит строго определенные поисковые запросы. Определенность в ключевом запросе даст гарантию показов только целевой аудитории.
2. Социальные сети в современном мире – это место, где общаются миллионы. Поскольку они собраны компактно, то велика вероятность быстро выйти на целевую аудиторию. Создание и превлечение в него людей со сходными интересами, поддерживайте жизнь. Ссылка на сайт может стоять в статусе, а можно просто давать анонсы на новый материал сайта.
3. Реклама в социальных сетях. В соцсетях есть возможность подачи рекламы. По эффективности ничто не сравнится с контекстной рекламой, но в соцсетях вы можете выбрать ту группу пользователей, которым будет показываться ваше объявление.
4. Бесплатные доски объявлений. В интернете немало бесплатных досок объявлений, но реально ценных не так уж много, а потому не стоит тратить время на доски с маленькой посещаемостью. Лучше потратить время на качественное объявление, которое обращает на себя внимание, и разместить его в трех-четырех местах, но серьезных, с вероятностью высокой конверсии.

# Структура программы

Бизнес-процессы программного продукта можно сформировать и описать BPWin-диаграммой (см. рисунок 1.1).

IDEF0 – нотация графического моделирования, используемая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции. Нотация IDEF0 является одной из самых популярных нотаций моделирования бизнес-процессов. К ее особенностям можно отнести:

* использование контекстной диаграммы;
* поддержка декомпозиции;
* доминирование;
* выделение 4 типов стрелок.

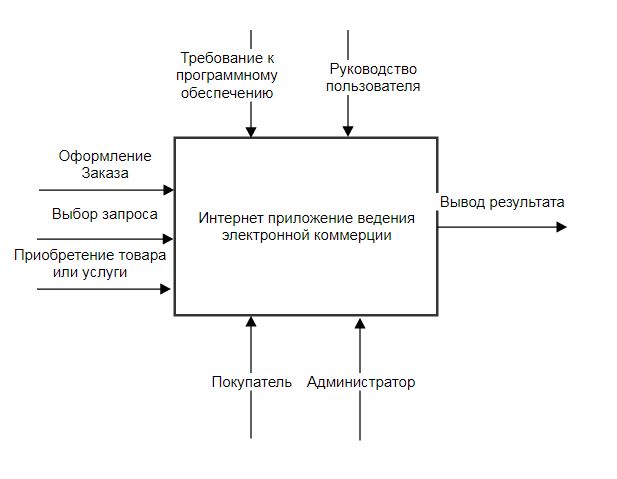


Рисунок 1.1 – Главный процесс «Интернет приложения ведения электронной коммерции»

# Выбор и обоснование среды разработки

Основной средой разработкой является Intellij IDEA – нтегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains. Эти же технологии лежат в основе самого редактора, что обеспечивает его кроссплатформенность, т.е. совместимость с операционными системами Mac, Windows и Linux. Редактор содержит огромное количество функций, позволяющих упростить и ускорить разработки, например функцией Live templates, которая позволяет быстро по ключевому слову подставить готовый цикл или создать новый класс. Так же имеется подсветка синтаксиса, подсказки при редактировании CSS, JS и HTML-файлов и огромное количество скачиваемых плагинов, позволяющих сделать редактор еще удобнее.

Для разработки базы данных использовалась приложение MySQL WorkBench.

MySQL Workbench – приложение для визуального проектирования баз данных, в составе которого инструменты проектирования, моделирования, создание и эксплуатация БД.

MySQL Workbench – преемник систем DBDesigner 4 из FabForce и предлагается в двух вариантах:

Community Edition – распространяется под свободной лицензией GNU GPL Standard Edition – доступна по ежегодной оплачиваемой подписке. Эта версия включает в себя дополнительные функции, которые повышают производительность разработчиков и администраторов БД.

Возможности программы MySQL Workbench:

* Позволяет наглядно представить модель базы данных в графическом виде.
* Наглядный, функциональный механизм установки связей между таблицами, в том числе «многие ко многим» с созданием таблицы связей.
* Reverse Engineering – восстановление структуры таблиц из уже существующей на сервере БД (связи восстанавливаются в InnoDB, при использовании MyISAM – связи необходимо устанавливать вручную).
* Удобный редактор SQL запросов, позволяющий сразу же отправлять их серверу и получать ответ в виде таблицы.
* Возможность редактирования данных в таблице в визуальном режиме.

Для создания сайта на локальном компьютере используется технология Spring Boot для быстрого развертывания веб сервера на локальном компьютере.

Состав программного обеспечения, нужного для данного дипломного проекта:

* Intellij idea;
* MySQL 5.1.73 / 5.5.53 / 5.6.34 / 5.7.16;
* Java 8 и выше;
* Apache maven 3.3.2;

Apache Maven – это фреймворк для автоматизации сборки проектов на основе описания их структурыв в файлах на языке POM, являющемся подмножеством XML. Maven обеспечивает [декларативную](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%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), а не [императивную](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%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) (в отличие от средства автоматизации сборки [Apache Ant](https://ru.wikipedia.org/wiki/Apache_Ant)) сборку проекта. В файлах описания проекта содержится его спецификация, а не отдельные команды выполнения. Все задачи по обработке файлов, описанные в спецификации, Maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов.

MySQL – это одна из самых популярных и самых распространенных СУБД (система управления базами данных) в интернете. Она не предназначена для работы с большими объемами информации, но ее применение идеально для интернет сайтов, как небольших, так и достаточно крупных. MySQL отличатся хорошей скоростью работы, надежностью, гибкостью. Работа с ней, как правило, не вызывает больших трудностей.

Немаловажным фактором является ее бесплатность. MySQL распространяется на условиях общей лицензии GNU (GPL, GNU Public License).

Java – это сильно типизированный [объектно-ориентированный язык программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%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%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%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%D1%8F), разработанный компанией [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) (в последующем приобретённой компанией [Oracle](https://ru.wikipedia.org/wiki/Oracle)). Программы на Java [транслируются](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в [байт-код Java](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4_Java), выполняемый [виртуальной машиной Java](https://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как [интерпретатор](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80). Достоинством подобного способа выполнения программ является полная независимость байт-кода от [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и [оборудования](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0), что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

# ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

# Входная информация

Входной информацией является, информация которую пользователь вносит в базу данных, заполняя необходимые поля выбранной таблицы, а также вводя информацию в базу данных с помощью запросов SQL.

Для разрабатываемого программного продукта входной будет служить следующая информация:

* Регистрация пользователя.
* Добавление товара.
* Добавление новостей.
* Регистрация заказов.

# Организация данных

Для хранения данных часто используются таблицы реляционных баз данных. Обычно данные могут быть представлены как абстрактные пары «ключ-значение». Данные могут быть организованы в различные виды структур данных, таких как массивы, связанные списки или объекты. Структуры данных могут хранить данные различных типов, включая числа, строки и другие структуры данных. Ввод и вывод данных в компьютеры производится через периферийные устройства.

Модель данных (datamodel) отображает состав и связи таблиц БД, использующихся в программном проекте. Модель данных для данного приложения смотреть на рисунке 2.1.

Модель данных генерируется в базе данных автоматически по модели классов интернет приложения.

Модель можно условно разделить на 4 блока которые имеют связи между собой.

* Продукты (Products).
* Заказы (Order).
* Корзина (CartInfo).
* Пользователи (Users).

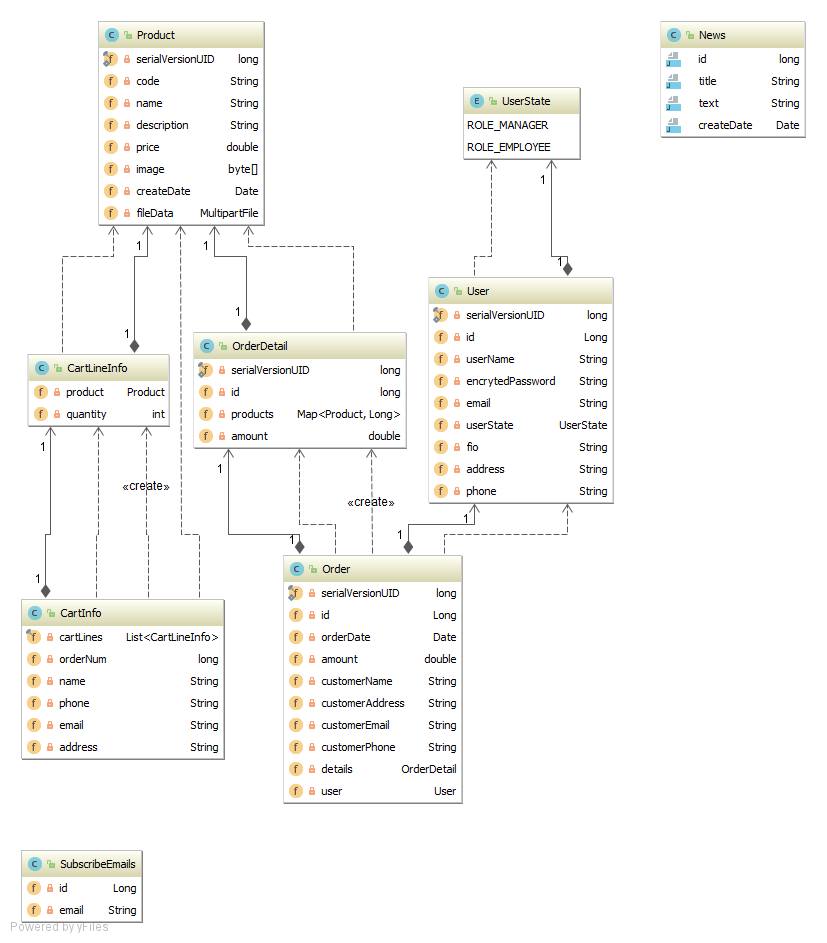


Рисунок 2.1 – Модель данных «Интернет приложение ведения электронной коммерции»

Все данные в таблицах соответвуют полям и имени классов, типы данных устанавливаются автоматически.

# Выходная информация

Выходная информация – результат выполнения запросов, выбранных страниц, вывод необходимой информации в отчет, печать информации.

Информация, получаемая в результате выполнения функций информационной системы и выдаваемая на объект ее деятельности, пользователю или в другие системы.

Выходной информацией для данного приложения является:

* Списки товаров и услуг.
* Профиль пользователя.
* Перечень заказов пользователя.
* Популярные товары или услуги в разделе «Лучшее предложение».

# Возможности системы

Возможности системы можно представить в виде диаграммы деятельности (см. прил. A).

Диаграмма деятельности – это диаграмма, на которой показано расположение некоторой деятельности на её составные части. Под деятельностью понимается спецификация исполняемого поведения в виде последовательного и параллельно выполнения подчинённых элементов, вложенных видов деятельности и отдельных действий, связанных между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграмма показывает последовательность действий, необходимых для ее достижения.  Действия (action) это элементарные шаги, которые не предполагают дальнейшую декомпозицию.

Деятельность может содержать входящие и/или исходящие дуги деятельности, показывающие потоки управления и потоки данных. Если поток соединяет две деятельности, он является потоком управления. Если поток заканчивается объектом, он является потоком данных.

Деятельность выполняется, только тогда, когда готовы все его «входы», после выполнения, деятельность передает управление и(или) данные на свои «выходы».

В данной диаграмме показано, как пользователь использует веб-приложение. Первоначально проходит процесс регистрации, где пользователь заполняет поля. При вводе логина, уже существующего в базе, предусмотрен вывод сообщения об ошибке.

Далее представлено, как пользователь выполняет процедуру заказа выбранного или выбранных товаров и услуг.

# Описание взаимодействия объектов системы

Взаимодействие объектов в системе можно представить диаграммой последовательностей (см. прил. Б).

Главный акцент в диаграмме последовательностей – порядок и динамика поведения, то есть, как и в каком порядке происходят события. Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

В данной диаграмме показано, как пользователь взаимодействует с интернет приложением, а оно в свою очередь с базой данных. Показано что пользователь должен зарегистрироваться, после войти, а дальше заполнить данные о себе включающие данные о адресе доставки заказа. Далее он добавляет товары в корзину и формирует заказ.

# 

# Варианты использования системы

Взаимодействие между актерами и вариантами использования представлено диаграммой вариантов использования (см. прил. В).

Диаграмма вариантов использования – это исходное концептуальное представление или концептуальная модель системы в процессе ее проектирования и разработки. Создание диаграммы вариантов использования имеет следующие цели:

Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.

Сформулировать общие требования к функциональному поведению проектируемой системы.

Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.

Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Назначение данной диаграммы состоит в следующем: проектируемая программная система представляется в форме так называемых вариантов использования, с которыми взаимодействуют внешние сущности или актеры. При этом актером или действующим лицом называется любой объект, субъект или система, взаимодействующая с моделируемой бизнес системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему так, как определит разработчик. Вариант использования служит для описания сервисов, которые система предоставляет актеру.

В данной диаграмме показано, что администратор может создать новость, добавить продукт, а конечный пользователь использовать данные созданные администратором. Варианты использования добавления товаров в корзину, оформление заказа может использовать любой пользователь вне зависимости от прав пользователя. Также показано что администратору и пользователю дана возможность редактирования контактнх данных (удаление, изменение, добавление).

# РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

# Разработка класса User

Класс User описывает сущность пользователя. Основной задачей данного класса является вход и регистрация на сайте. Данный функционал реализован следующим кодом:

**package** org.solteh.model;  
  
**import** com.fasterxml.jackson.annotation.JsonIgnoreProperties;  
  
**import** javax.persistence.Column;  
**import** javax.persistence.Entity;  
**import** javax.persistence.Enumerated;  
**import** javax.persistence.GeneratedValue;  
**import** javax.persistence.GenerationType;  
**import** javax.persistence.Id;  
**import** javax.persistence.Table;  
**import** java.io.Serializable;  
  
@Entity  
@Table(name = **"Users"**)  
@JsonIgnoreProperties({**"hibernateLazyInitializer"**, **"handler"**})  
**public class** User **implements** Serializable {  
 **private static final long *serialVersionUID*** = -812013033671051986L;  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 **private** Long **id**;  
  
 @Column(name = **"User\_Name"**, length = 20, unique = **true**, nullable = **false**)  
 **private** String **userName**;  
  
 @Column(name = **"Encryted\_Password"**, length = 128, nullable = **false**)  
 **private** String **encrytedPassword**;  
 @Column(name = **"email"**, unique = **true**)  
 **private** String **email**;  
  
 @Enumerated  
 @Column(name = **"User\_State"**)  
 **private** UserState **userState**;  
  
 @Column  
 **private** String **fio**;  
 @Column  
 **private** String **address**;  
 @Column  
 **private** String **phone**;  
  
 **public** String getUserName() {  
 **return userName**;  
 }  
  
 **public void** setUserName(String userName) {  
 **this**.**userName** = userName;  
 }  
  
 **public** String getEncrytedPassword() {  
 **return encrytedPassword**;  
 }  
  
 **public void** setEncrytedPassword(String encrytedPassword) {  
 **this**.**encrytedPassword** = encrytedPassword;  
 }  
  
 **public** UserState getUserState() {  
 **return userState**;  
 }  
  
 **public void** setUserState(UserState userState) {  
 **this**.**userState** = userState;  
 }  
  
 **public** Long getId() {  
 **return id**;  
 }  
  
 **public void** setId(Long id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getEmail() {  
 **return email**;  
 }  
  
 **public void** setEmail(String email) {  
 **this**.**email** = email;  
 }  
  
 **public** String getFio() {  
 **return fio**;  
 }  
  
 **public void** setFio(String fio) {  
 **this**.**fio** = fio;  
 }  
  
 **public** String getAddress() {  
 **return address**;  
 }  
  
 **public void** setAddress(String address) {  
 **this**.**address** = address;  
 }  
  
 **public** String getPhone() {  
 **return phone**;  
 }  
  
 **public void** setPhone(String phone) {  
 **this**.**phone** = phone;  
 }  
  
 **public boolean** isCorrectPersonalInfo() {  
 **return** getAddress() != **null** && !getAddress().isEmpty() &&  
 getFio() != **null** && !getFio().isEmpty() &&  
 getEmail() != **null** && !getEmail().isEmpty() &&  
 getPhone() != **null** && !getPhone().isEmpty();  
 }  
}

# Разработка класса Product

Основной задачей класса Product является хранение информации о товаре или услуге, их описании и дате создания. Его код представлен ниже:

**package** org.solteh.model;  
  
**import** org.springframework.web.multipart.MultipartFile;  
  
**import** javax.persistence.Column;  
**import** javax.persistence.Entity;  
**import** javax.persistence.Id;  
**import** javax.persistence.Lob;  
**import** javax.persistence.Table;  
**import** javax.persistence.Temporal;  
**import** javax.persistence.TemporalType;  
**import** javax.persistence.Transient;  
**import** java.io.Serializable;  
**import** java.util.Date;  
  
@Entity  
@Table(name = **"Products"**)  
**public class** Product **implements** Serializable {  
  
 **private static final long *serialVersionUID*** = -1000119078147252957L;  
  
 @Id  
 @Column(name = **"Code"**, length = 20, nullable = **false**)  
 **private** String **code**;  
  
 @Column(name = **"Name"**, length = 255, nullable = **false**)  
 **private** String **name**;  
  
 @Column(name = **"Description"**)  
 **private** String **description**;  
  
 @Column(name = **"Price"**, nullable = **false**)  
 **private double price**;  
  
 @Lob  
 @Column(name = **"Image"**, length = Integer.***MAX\_VALUE***, nullable = **true**)  
 **private byte**[] **image**;  
  
 @Temporal(TemporalType.TIMESTAMP)  
 @Column(name = **"Create\_Date"**, nullable = **false**)  
 **private** Date createDate;  
  
 @Transient  
 **private** MultipartFile fileData;  
  
 **public static long** getSerialVersionUID() {  
 **return** serialVersionUID;  
 }  
  
 **public** String getCode() {  
 **return** code;  
 }  
  
 **public void** setCode(String code) {  
 **this**.code = code;  
 }  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
  
 **public double** getPrice() {  
 **return** price;  
 }  
  
 **public void** setPrice(**double** price) {  
 **this**.price = price;  
 }  
  
 **public byte**[] getImage() {  
 **return** image;  
 }  
  
 **public void** setImage(**byte**[] image) {  
 **this**.image = image;  
 }  
  
 **public** Date getCreateDate() {  
 **return** createDate;  
 }  
  
 **public void** setCreateDate(Date createDate) {  
 **this**.createDate = createDate;  
 }  
  
 **public** String getDescription() {  
 **return** description;  
 }  
  
 **public void** setDescription(String description) {  
 **this**.description = description;  
 }  
  
 **public** MultipartFile getFileData() {  
 **return** fileData;  
 }  
  
 **public void** setFileData(MultipartFile fileData) {  
 **this**.fileData = fileData;  
 }  
}

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

# Типы ошибок

Тестирование программных продуктов – процесс исследования, испытания программного обеспечения, имеющий две цели:

* продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;
* выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

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

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

Качество программного обеспечения можно определить, как совокупную характеристику исследуемого ПО с учётом следующих составляющих:

* надёжность;
* сопровождаемость;
* практичность;
* эффективность;
* мобильность;
* функциональность.

Так же существуют различные уровни тестирования:

* Модульное тестирование – тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. Часто модульное тестирование осуществляется разработчиками программного обеспечения.
* Интеграционное тестирование – тестируются интерфейсы между компонентами, подсистемами или системами. При наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.
* Системное тестирование – тестируется интегрированная система на её соответствие требованиям.

Существует множество подходов к решению задачи тестирования и верификации ПО, но эффективное тестирование сложных программных продуктов – это процесс в высшей степени творческий, не сводящийся к следованию строгим и чётким процедурам или созданию таковых.

# Функциональное тестирование

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

Во-первых, тестирование дает уверенность в том, что в ходе разработки были реализованы все функциональные требования заказчика к продукту.

Во-вторых, тестирование обеспечивает надежность и безопасность продукта на предмет постороннего вмешательства или неправомерного доступа к конфиденциальной информации. Данные задачи стоят перед таким тестированием безопасности.

Функциональное тестирование является одним из ключевых видов тестирования, задача которого – установить соответствие разработанного программного обеспечения (ПО) исходным функциональным требованиям заказчика. То есть проведение функционального тестирования позволяет проверить способность информационной системы в определенных условиях решать задачи, нужные пользователям.

Его ключевые преимущества:

* ПО полностью имитирует фактическое использование системы.
* Позволяет своевременно выявить системные ошибки ПО и, тем самым, избежать множества проблем при работе с ним в дальнейшем.
* Экономия за счет исправления ошибок на более раннем этапе жизненного цикла ПО.

Функциональные требования включают в себя:

* Функциональная пригодность (англ. suitability).
* Точность (англ. accuracy).
* Способность к взаимодействию (англ. interoperability).
* Соответствие стандартам и правилам (англ. compliance).
* Защищённость (англ. security).

При проведении тестирования были испытаны все функции, которые были задуманы при разработке программного продукта.

Вот некоторые из них:

* Регистрация пользователя;
* Вход в системы пользователя;
* Создание сессии для работы с корзиной;
* Добавление товара или услуг;
* Добавление новостей;
* Вывод динамического отчета;
* Редактирование профиля пользователя;
* Управление товарами или услугами в корзине пользователя.

# Полное тестирование

Тестирование программы включает в себя проверку её работы в различной среде и разных условиях с целью выявления ошибок и проверки корректности её работы.

Тест – это последовательность входных данных и соответствующих им набор эталонных результатов, которые должны быть получены при верном функционировании программы.

Так как разрабатываемый программный продукт главным образом предназначен для ведения электронной коммерции. Исходя из этого, особое внимание уделялось основным функциям заказа и просмотра товаров и услуг. Также особое внимание было уделено интерфейсу программного продукта. За время пользования программой у пользователя не должны уставать глаза, а так же он должен быть интуитивно понятным. Полное тестирование можно проанализировать в тест кейсе (см. таблицу 4.1).

Таблица 4.1 – Тест кейсы

|  |  |  |
| --- | --- | --- |
| Действие | Ожидаемый результат | Результат теста |
| Адаптация страницы под различное разрешение | При работе с различными устройствами, веб-приложения должно изменять свои размеры | Тестирование прошло успешно. Веб-приложение изменяет размеры в соответствии с разрешением |
| Тестирование формы авторизации пользователя | После заполнения формы и нажатие кнопки, должна быть создана сессия, и изменено основное меню | Тестирование прошло успешно. Сессия создана, меню изменено |
| Тестирование формы регистрации пользователей | После заполнения формы с определёнными условиями и отправки данных, должна произойти регистрация пользователя | Тестирование прошло успешно. Данные добавлены в базу |
| Тестирование ввода товара или услуги с загрузкой изображения | При вводе данных товара или услуги должны добавиться в базу данных, включая изображение в двоичном виде. | Тестирование прошло успешно. Данные добавлены в базу |
| Тестирование отчета о заказах пользователя | После нажатия на кнопку «Статистика заказов», данных загружаются из базы данных | Тестирование прошло успешно. Данные загружены и отображены пользователю |

Окончание таблицы 4.1

|  |  |  |
| --- | --- | --- |
| Тестирование отправки электронной почты | При вводе электронного адреса для подписки на последние новости, введенный email добавляется в базу данных и отправляется оповещение о подписке. | Тестирование прошло успешно |
| Тестирование выхода из сессии | После нажатия на кнопку «Выход» сессия должна завершится и страница перезагрузится | Тестирование прошло успешно |
| Тестирование разбиения на страницы | В зависимости от страницы указанной в url должен выполняться запрос на вывод данных с лимитов вывода указанным в функции | Тестирование прошло успешно |

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

# Назначение

Тема дипломного проекта «Интернет приложение ведения электронной коммерции».

Основной целью данного проекта является предоставление людям простого, интуитивно понятного доступа к сайту с предложением услуг или товаров.

В данной системе не содержится категорий что позволяет искать пользоватлю нужный товар сразу при первом посещении сайта. И это экономит время на поиск нужной категории в которой размещен услуга или товар.

Дипломный проект представляет собой интернет приложение предназначенное для ведения электронной коммерции. Так же для администратора есть возможность манипулировать данными в базе данных с помощью MySQL Workbench.

# Требования к компьютерному и аппаратному обеспечению

Минимальные требования, которым должен соответствовать компьютер для того, чтобы на нем работала данная программа, следующие:

* процессор: Intel Pentium III или AMD Sempron 2600+ и выше;
* объем оперативной памяти: 4 ГБ и выше;
* операционная система: ®Windows™XP, Android 4.4;
* веб-браузер Google Chrome и другие браузеры актуальных версий;
* наличие интернет-соединения.

Оптимальные требования, которым должен соответствовать компьютер для того, чтобы на нем работала данная программа, а пользователь чувствовал себя комфортно, следующие:

* процессор: IntelPentiumE5300 или AMD Sempron 3600+;
* 8 ГБ оперативной памяти;
* операционная система: ®Windows™XP/Vista/7/8.1/10, Android 7.0;
* веб-браузер Google Chrome и другие браузеры актуальных версий;
* наличие интернет-соединения.

Функционирование программы на ПЭВМ, характеристики которой удовлетворяют этим требованиям, приведет к ускорению обработки данных приложением и повышению удобства и эффективности его использования.

# Работа с приложением

Для запуска проекта на обычном локальном компьютере рекомендуется использовать Java класс SpringBootApplication.class с методом main. Для того, чтобы запустить проект, требуется запустить Intellij Idea нажать на правую кнопку по методу main и запустить с помощью копанды «Run SpringBootApplication.class».

После чего запустить компиляция проекта и запуск приложения, в ходе запуска будет отображена консоль для мониторинга состояния сервера и приложения. После завершения запуска будет написана строка в консоли:

«»

Чтобы открыть сайт сайт требуется в адресную строку браузера вести название домена (стандартный «http://localhost:8080»).

Для манипулирования данными в базе данных рекомендуется использовать MySQL Workbench скачиваемый отдельно.

Для входа на главную страницу приложения введите в url-строку адрес “http://localhost:8080”.

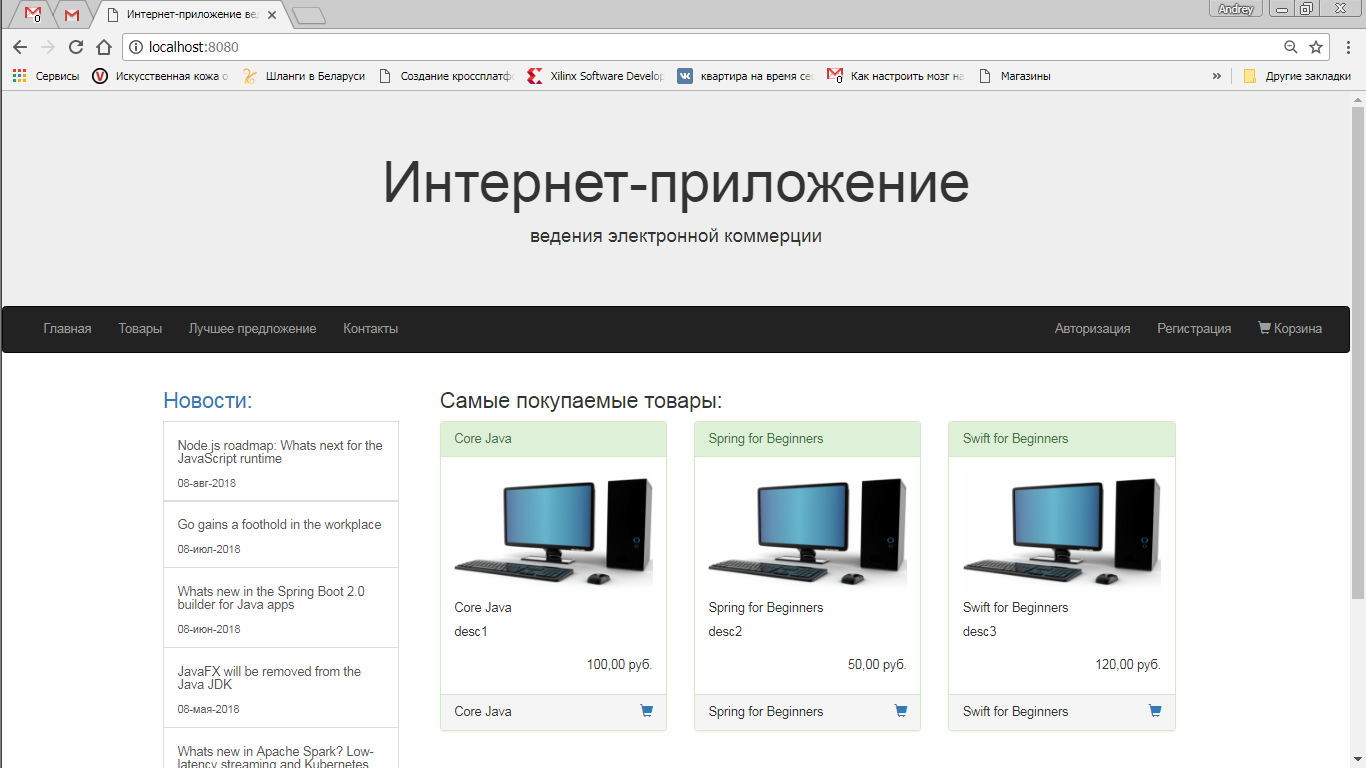


Рисунок 5.1 – Главная страница «Интернет приложения ведения электронной коммерции»

Для регистрации аккаунта нужно нажать на кнопку «Регистрация», откроется новая страница, в котором нужно ввести данные пользователя (см. рисунок 5.2).

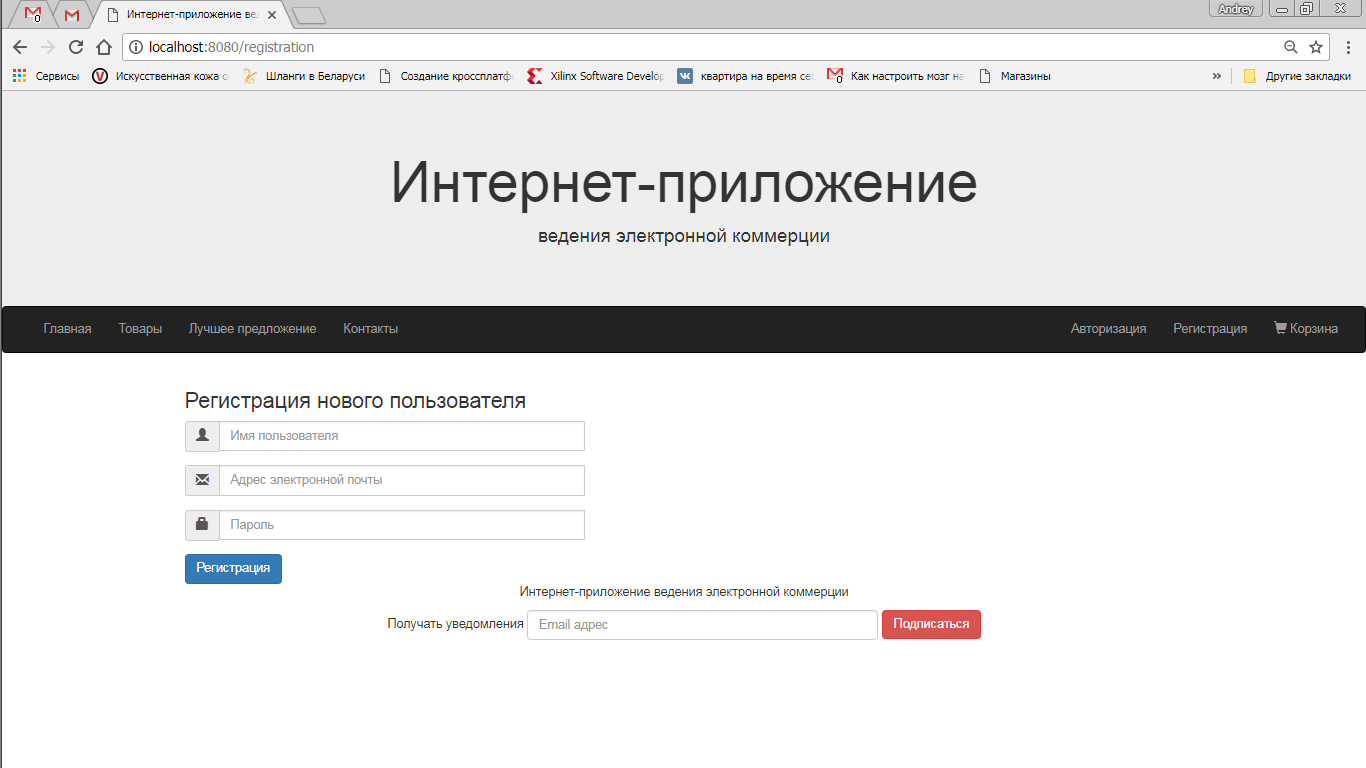


Рисунок 5.2  – Страница регистрации «Интернет приложения ведения электронной коммерции»

Для входа в аккаунт нужно нажать на кнопку «Войти», после чего откроется страница входа в аккаунт, где нужно вписать данные указанные при регистрации (см. рисунок 5.3).

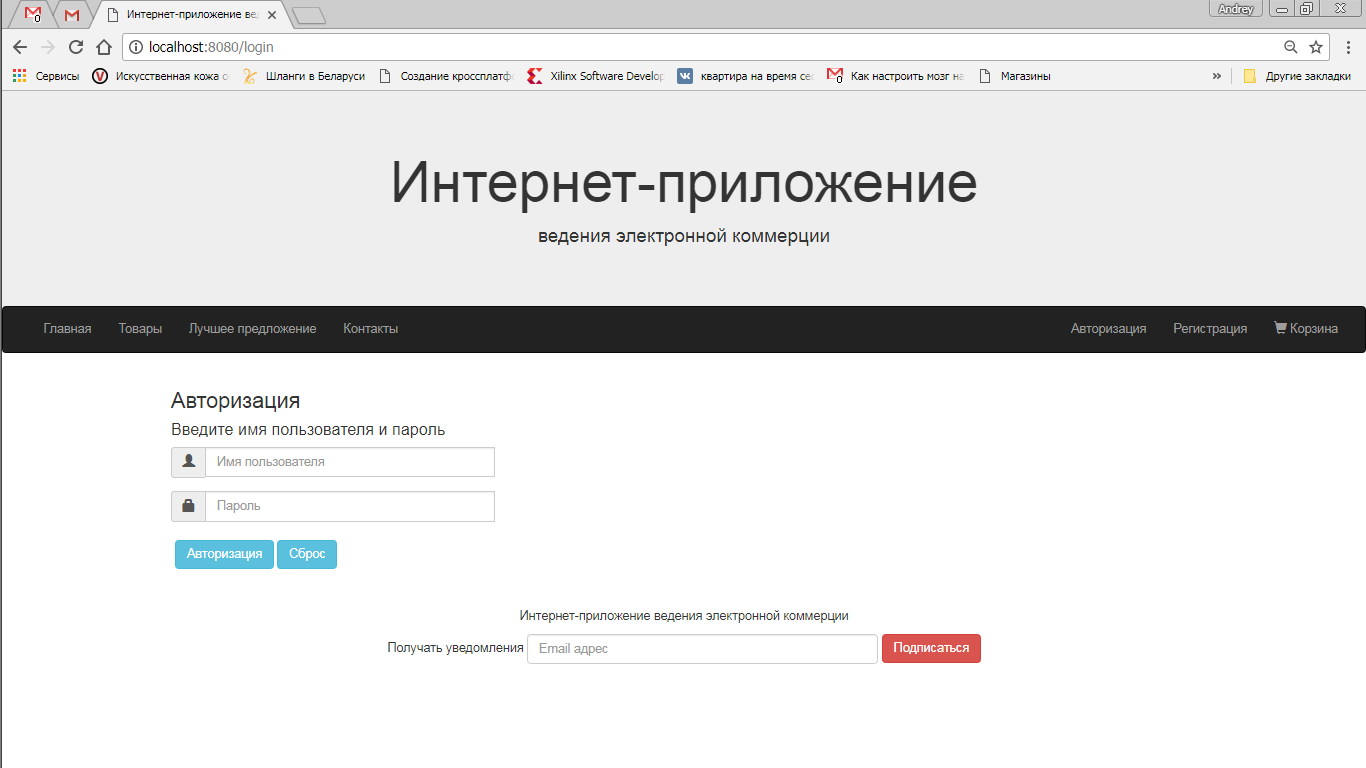


Рисунок 5.3 – Страница входа «Интернет приложения ведения электронной коммерции»

После входа откроется возможность редактирования данных.

Так как редактирование отличается только полями ввода, оно будет показано на примере заполнения профиля.

Для того что открыть окно редактирования нужно нажать на кнопку «Привет …. » (см. рисунок 5.4).

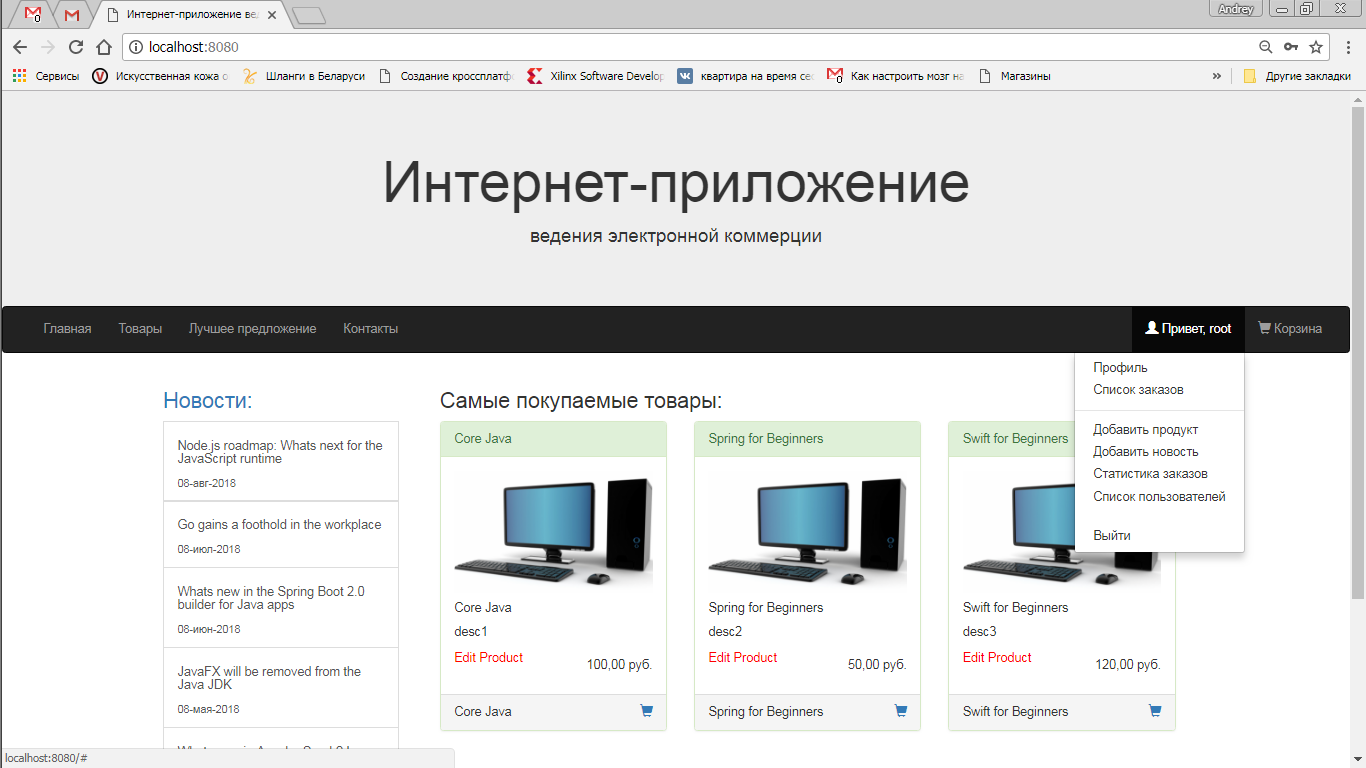


Рисунок 5.4 – Кнопка «Привет, … » и ее выпадающий список «Интернет приложения ведения электронной коммерции»

И нажать на кнопку «Профиль», после откроется страница, в котором нужно ввести данные (см. рисунок 5.5).

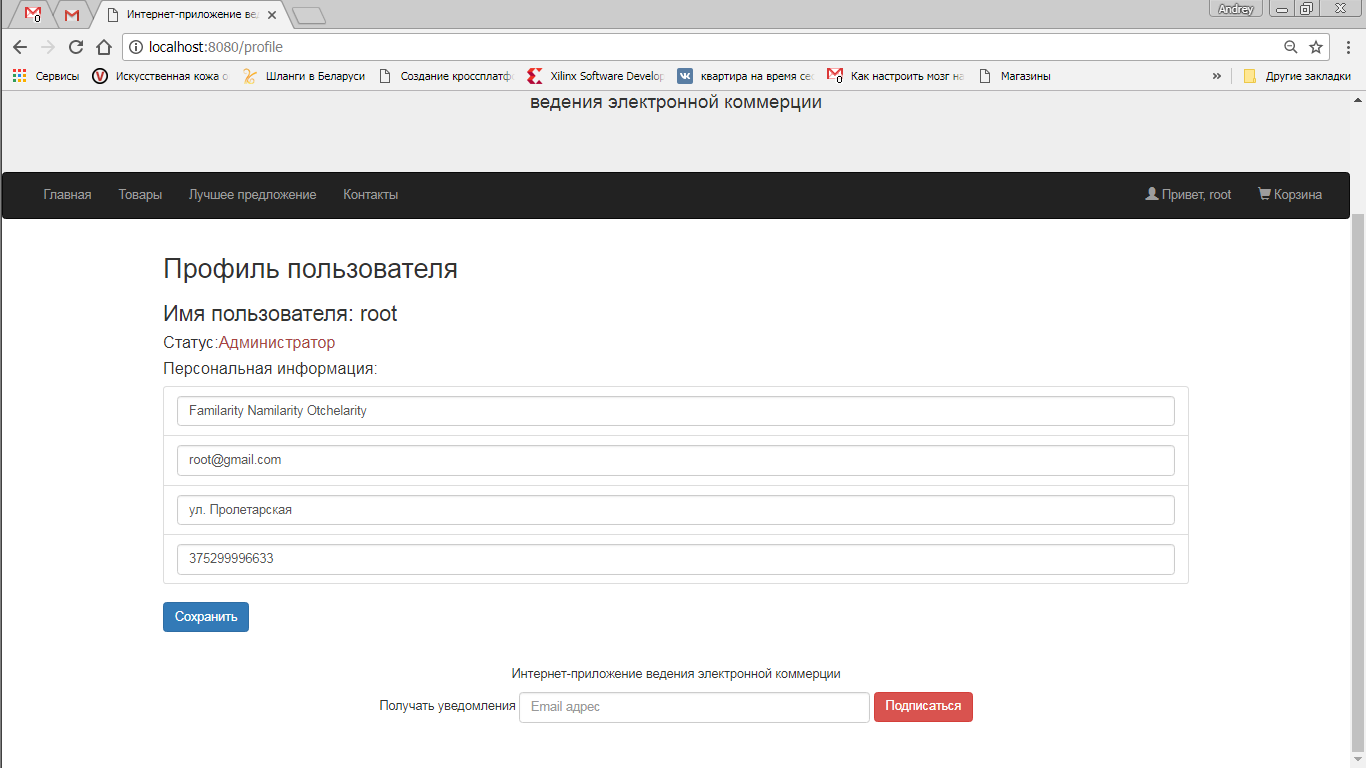


Рисунок 5.5 – Страница редактирования профиля «Интернет приложения ведения электронной коммерции»

Далее нужно ввести данные, нажать на кнопку «Сохранить» и после вас перенаправит на главную страницу.

Для открытия списка товара и услуг, нужно нажать на соответствующую кнопку (см. рисунок 5.6).

Пример списков показана на рисунках 5.7, 5.8, 5.9.

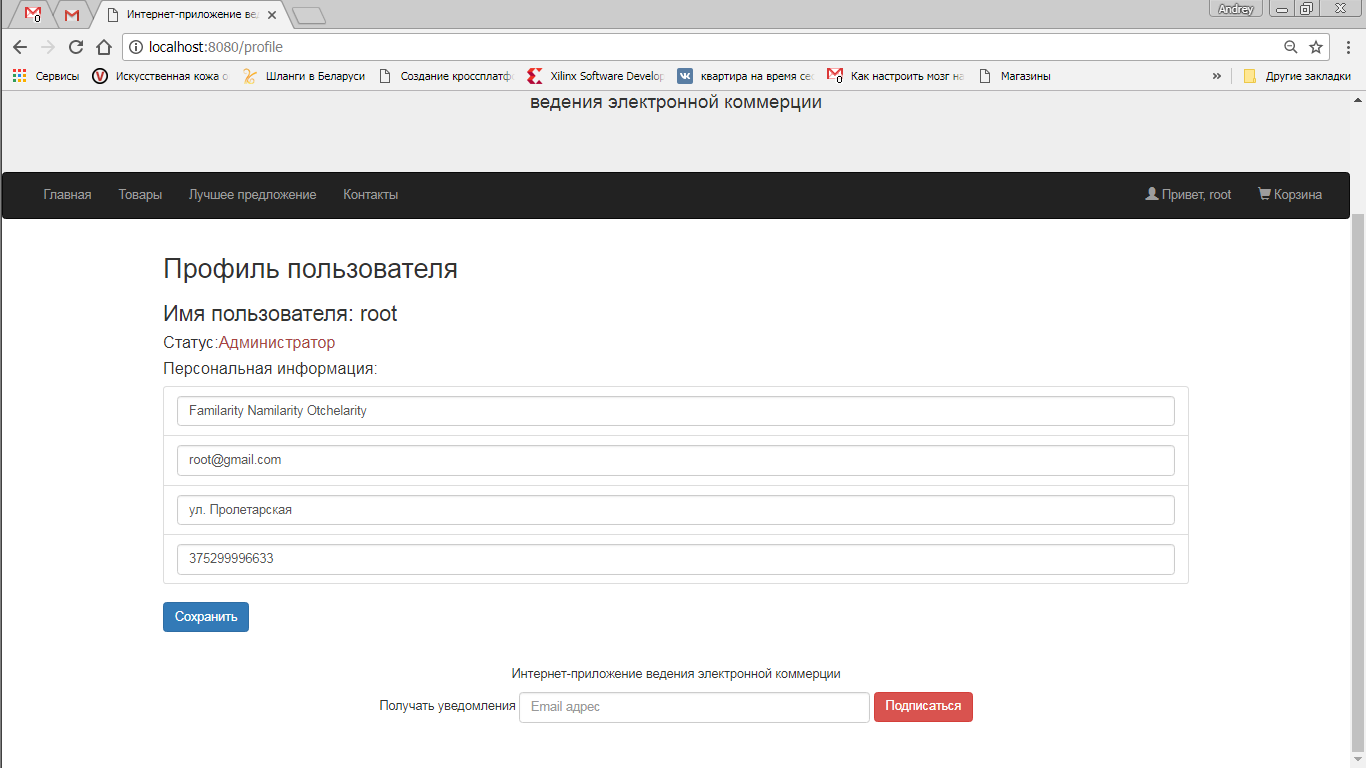


Рисунок 5.6 – Кнопки открытия списков «Интернет приложения ведения электронной коммерции».

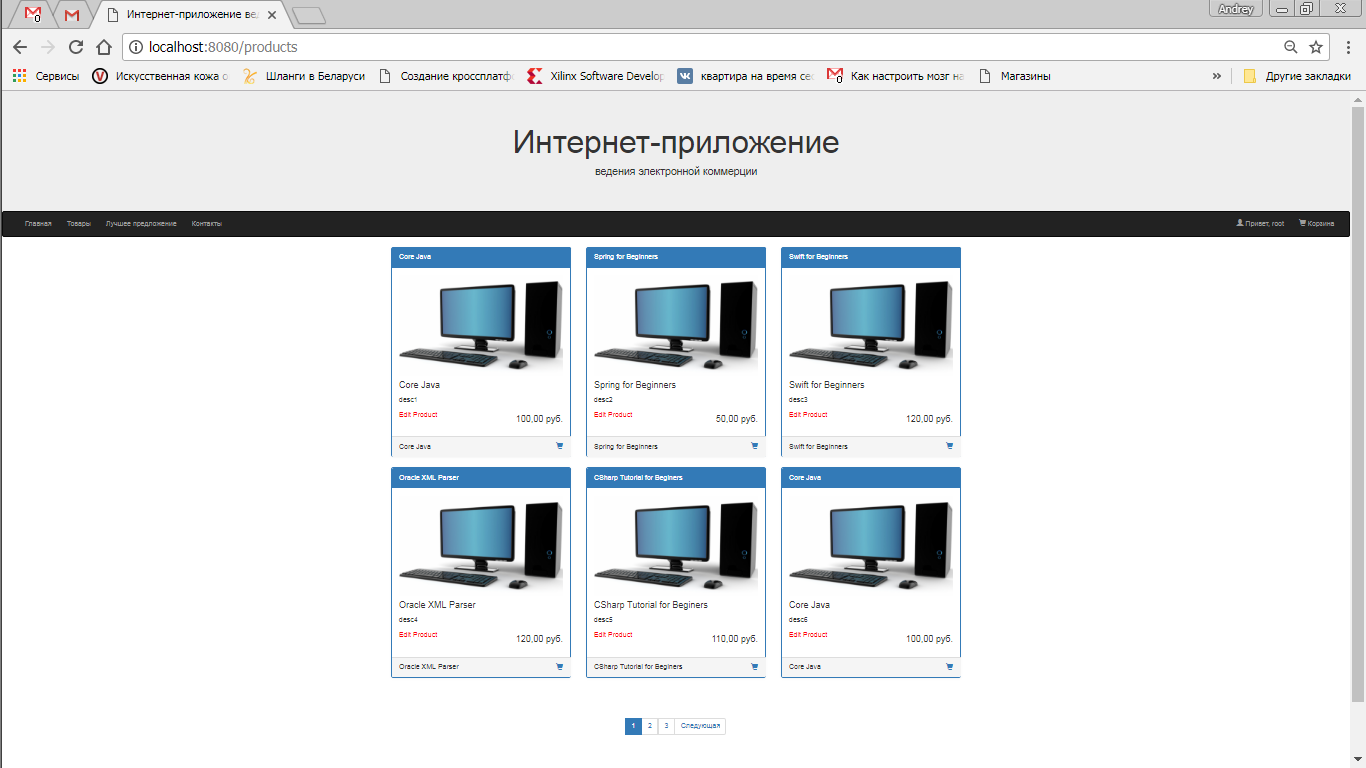


Рисунок 5.7 – Список товаров и услуг «Интернет приложения ведения электронной коммерции».

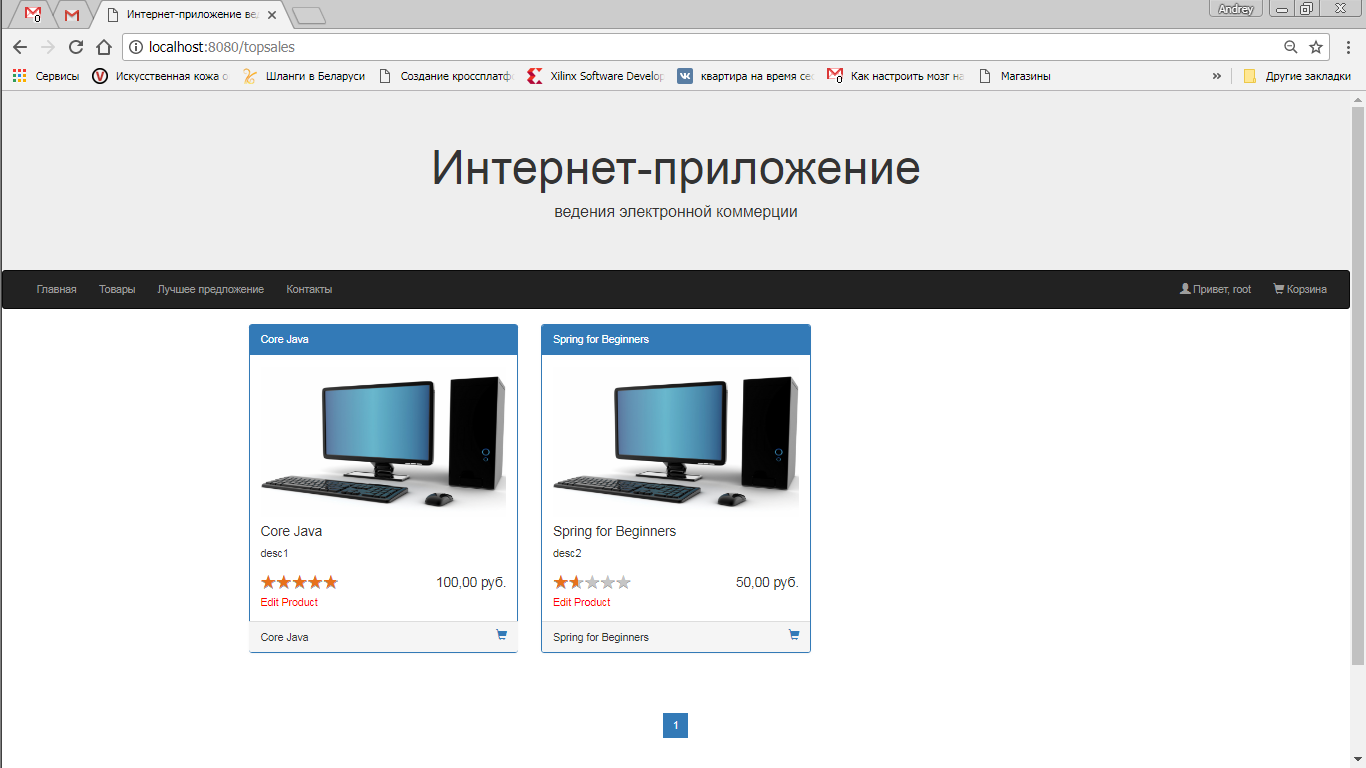


Рисунок 5.8 – Лучшее предложение, автоматически подобранное в «Интернет приложения ведения электронной коммерции»

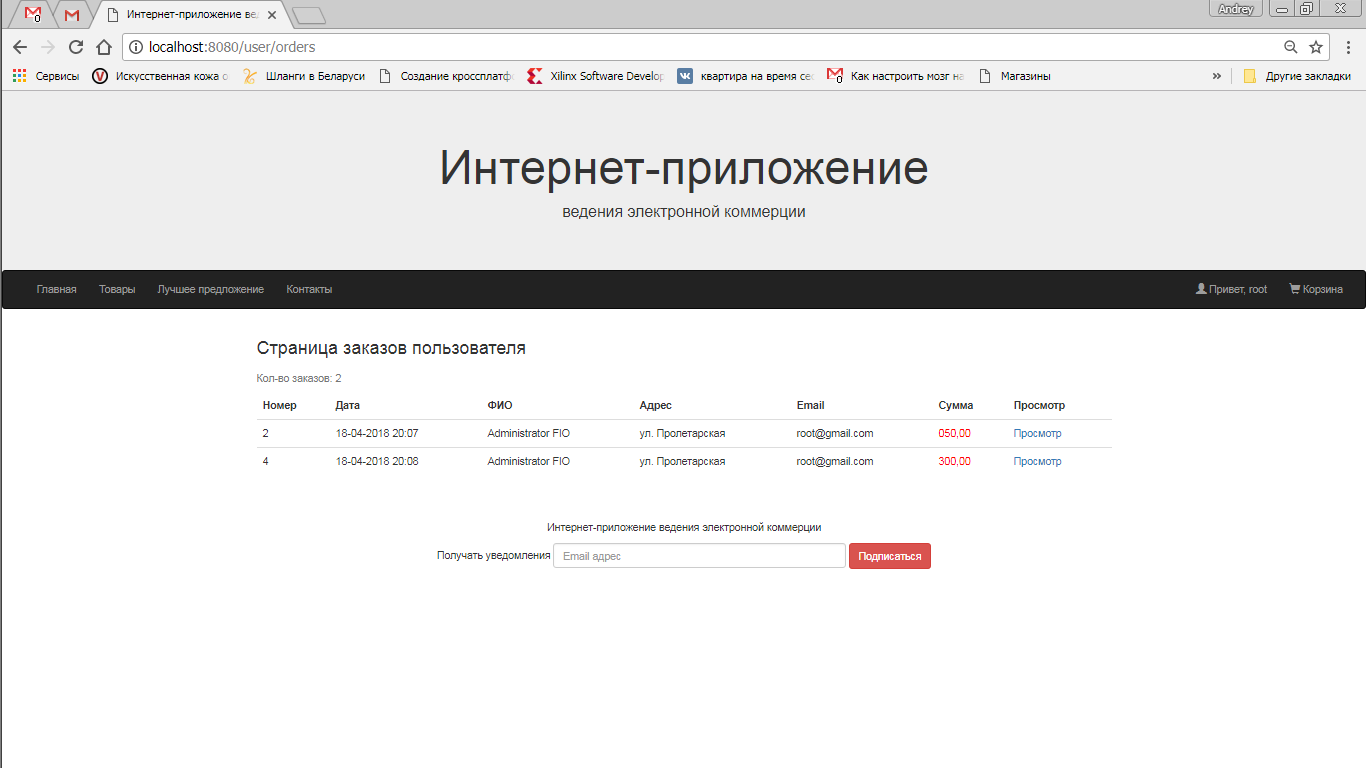


Рисунок 5.9 – Список заказов «Интернет приложения ведения электронной коммерции».

Для добавления товара или услуги в корзину нужно нажать на значек «Корзина» (см. рисунок 5.10). После чего вы будете перенаправлены на страницу со списков товаров добавленных в корзину пользователя.

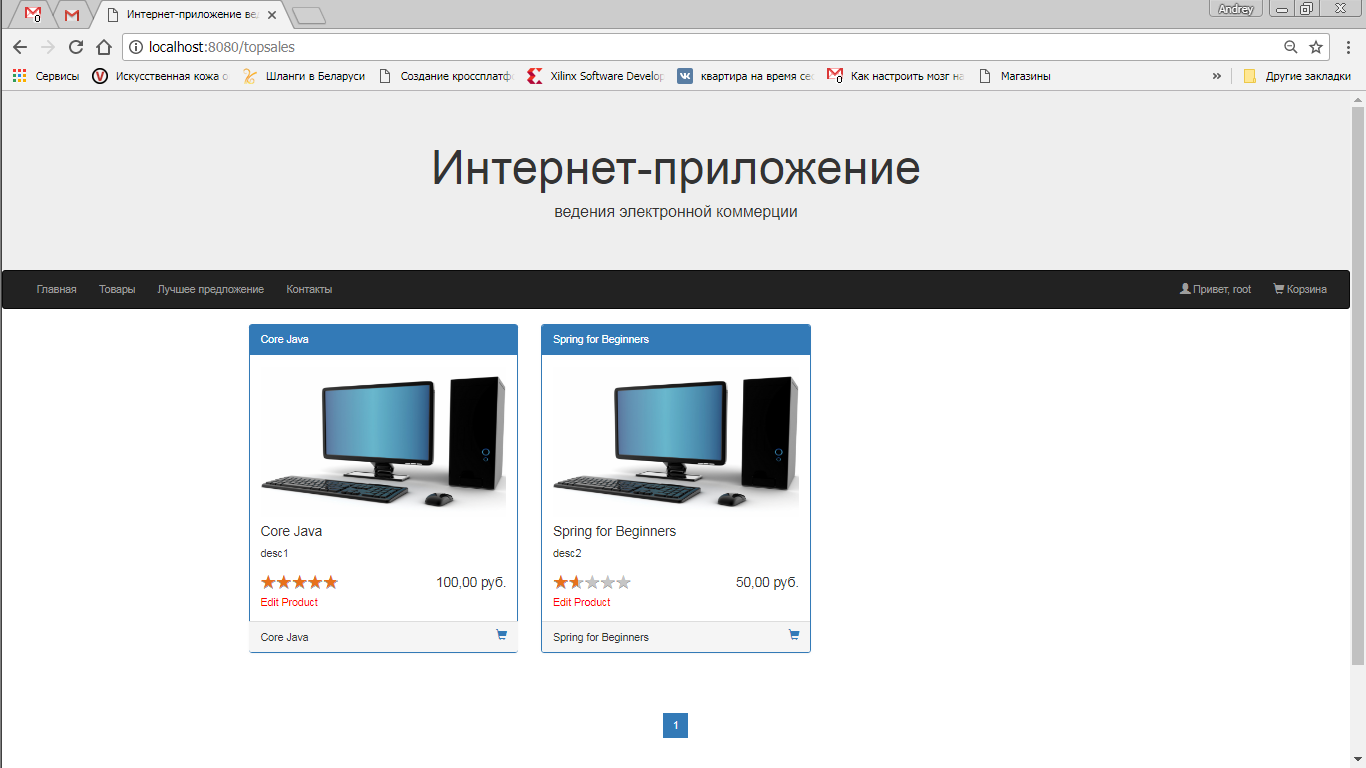


Рисунок 5.10 – Кнопка «В корзину» в «Интернет приложение ведения электронной коммерции».

Для просмотра списка заказов нужно нажать на кнопку «Список заказов» в списке (см. рисунок 5.4). Откроется окно, в котором отобразятся все заказы авторизованного пользователя (см. рисунок 5.9).

При нажатии на кнопку «Подписаться» (см. рисунок 5.11)  в поле «Получать уведомления» автоматически добавится email в базу данных и отправлется уведомление на почту об успешной подписке на свежие новости.

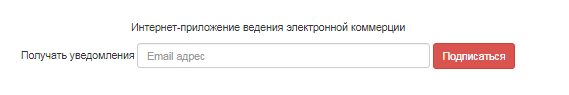


Рисунок 5.11 – Кнопка «Подписаться» в «Интернет приложение ведения электронной коммерции».

Для открытия страницы динамического отчета «Статистика заказов» (см. рисунок 5.12) нужно нажать на кнопку «Статистика заказов» (см. рисунок 5.4).

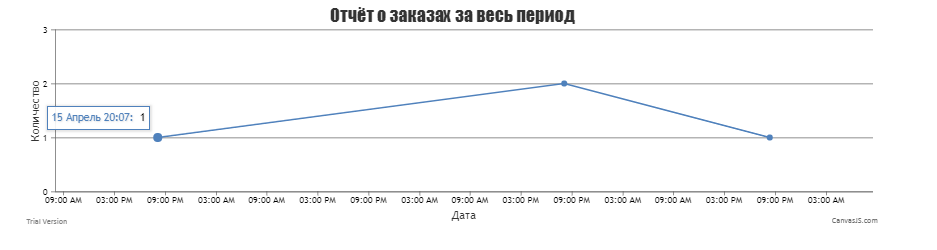


Рисунок 5.12 – Статистика заказов в «Интернет приложение ведения электронной коммерции».

Для выхода из сессии можно нажать на кнопку «Выйти» в списке (см. рисунок 5.4) или закрыть браузер.

# ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

# Затраты на используемые материалы

В данном разделе пояснительной записки рассчитаем затраты на разработанное программное обеспечение, тема дипломного проекта «Интернет приложение ведения электронной коммерции».

Для начала рассчитаем затраты на используемые материалы, к ним относятся оптический диск (CD-R), бумага, канцтовары (папка-скоросшиватель), картридж с краской.

Стоимость бумаги рассчитаем по формуле:

 (6.1)

где *N*пачк – количество листов в пачке (*Nпачк* =500), лист;

*С*пачк – стоимость одной пачки бумаги (*Спачк*=6), руб.



Затраты по остальным материалам определяются по экспертным оценкам. Результаты расчёта представлены в виде таблицы (см. таблицу 6.1).

Таблица 6.1 – Расчёт стоимости материалов

|  |  |  |
| --- | --- | --- |
| Материалы | Количество, шт. | Стоимость, руб. |
| Магнитный носитель | 1 | 1,5 |
| Бумага | 100 | 1,2 |
| Папка-скоросшиватель | 1 | 3,06 |
| Краска для принтера | 1 | 36 |
| Итого | | 41,76 |

# Расчёт трудоёмкости

После того как рассчитали затраты на использованные материалы перейдем к расчету трудоёмкости.

Трудоёмкость – это затраты рабочего времени на производство единицы продукции. Так через трудоемкость в конечном итоге, оценивается один из основных затратных показателей на оплату труда исполнителей.

Общие затраты труда на разработку и внедрение изделия (проекта) ** определяют по следующей формуле:

 (6.2)

где *t****i*** – затраты труда на выполнение *i*-го этапа проекта.

Для заполнения столбцов 3, 4, 5 в таблице 6.2 воспользуемся методом экспертной оценки. При экспертной оценке ожидаемая продолжительность работ вычисляется по формуле:

 (6.3)

где *T*min – минимальная продолжительность работы;

*T*max – максимальная продолжительность работы.

Таблица 6.2 – Перечень выполняемых работ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  раработы | Стадии разработки | Трудоемкость, чел.дни | | |
| *T*min | *T*max | *T*i |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Постановка задачи | 1 | 1 | 1 |
| 2 | Сбор материалов и анализ существующих разработок | 1 | 3 | 2 |
| 3 | Подбор литературы | 1 | 1 | 1 |
| 4 | Определение требований к системе | 1 | 1 | 1 |
| 5 | Определение стадий, этапов и сроков разработки | 1 | 2 | 1 |
| 6 | Анализ программных средств схожей тематики | 1 | 2 | 1 |
| 7 | Разработка функциональной схемы программы | 2 | 3 | 2 |
| 8 | Разработка структуры программы управления по подсистемам | 3 | 5 | 4 |
| 9 | Документирование | 3 | 4 | 3 |
| 10 | Определение требований к программе управления | 1 | 1 | 1 |
| 11 | Выбор инструментальных средств | 1 | 2 | 1 |
| 12 | Определение свойств и требований к аппаратному обеспечению | 2 | 2 | 2 |
| 13 | Программирование | 15 | 20 | 20 |
| 14 | Тестирование и отладка программы управления | 3 | 5 | 3 |
| 15 | Разработка программной документации | 2 | 3 | 2 |
| 16 | Согласование и утверждение работоспособности системы | 2 | 3 | 2 |
| 17 | Опытная эксплуатация | 3 | 5 | 4 |
| 18 | Анализ данных, полученных в результате эксплуатации | 3 | 5 | 4 |

Окончание таблицы 6.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 19 | Корректировка технической документации по результатам испытаний | 3 | 5 | 4 |
| Итог | | | | 59 |

Воспользуемся формулой общие затраты труда на разработку и внедрение проекта (см. формулу 6.2) и рассчитаем общие затраты на разработку продукта, исходя из таблицы 6.2.

*Qp*=1+2+1+1+1+1+2+4+3+1+1+2+21+3+2+2+4+4+4=59 дней.

# Построение графика разработки программного продукта

В качестве инструмента планирования работ используем ленточный график. Ленточный график позволяет наглядно представить логическую последовательность и взаимосвязь отдельных работ, срок начала и срок окончания работ.

Он представляет собой таблицу, где перечислены наименования стадий разработки и видов работ, длительность выполнения каждого вида работ. Продолжением таблицы является график, отражающий продолжительность каждого вида работ в виде отрезков времени, которые располагаются в соответствии с последовательностью выполнения работ (см. таблицу 6.3).

Таблица 6.3 – Ленточный график

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Наименование работ | Календарные месяцы, дни | | | | | | | | |
| Апрель | | | Май | | | Июнь | | |  |  |
| 0 10 20 30 | | | 10 20 30 | | | 10 20 30 | | |
| ТЗ |  |  |  |  |  |  |  |  |  |
| ЭП |  |  |  |  |  |  |  |  |  |
| ТП |  |  |  |  |  |  |  |  |  |
| РП |  |  |  |  |  |  |  |  |  |
| ВП |  |  |  |  |  |  |  |  |  |

# Расчёт основной заработной платы

Основная заработная плата – заработная плата, начисленная рабочим и служащим за фактически проработанное время и выполнение работы на предприятии, рассчитывается по формуле:

 (6.4)

где *Т*ЗАН – число дней, отработанных исполнителем проекта; *О*ДН – дневной оклад исполнителя.

Величина месячного оклада определяется ставкой тарифного разряда так называемой Единой Тарифной Сетки (ЕТС). Тарификация специалистов по ЕТС в соответствии с Инструкцией о порядке применения Единой тарифной сетки работников Республики Беларусь относит техника-программиста со средне специальным образованием без опыта работы в диапазоне с 7 по 10 разряд. Отнесем техника-программиста без опыта работы к 7-му разряду ЕТС.

На предприятии тарифная ставка первого разряда равна 31 рубль. Тарифные коэффициенты представлены в единой тарифной сетке работников РБ (см. таблицу 6.4).

*О*мес = 31 \* 2,03 = 62,93 руб.

Исходя из этого, дневной оклад исполнителя будет равен:

*О*дн = 62,93/ 21= 3 руб.

Таблица 6.4 – Единая тарифная сетка работников РБ

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Разряды | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Коэфф. | 1 | 1,16 | 1,35 | 1,57 | 1,73 | 1,9 | 2,03 | 2,17 | 2,32 | 2,48 | 2,65 |

Теперь можно рассчитать основную ЗП, по формуле:

*С* З.ОЗН = 3 \* 59 = 177 руб.

Умножим на повышающий коэффициент, который в данном случае будет равен 2.

*С*З.ОЗН = 177 \* 2 = 354 руб.

# Отчисления в фонд социальной защиты населения

Отчисления в фонд социальной защиты населения (*З*фсзн) определяются в соответствии с действующими законодательными актами по нормативу, в процентном отношении к фонду основной и дополнительной зарплаты исполнителей, установленному в целом по организации и рассчитывается по формуле:

 (6.5)

где *Н*оз – норматив отчислений в фонд социальной защиты населения (%).



# Дополнительная заработная плата

Зная основную заработную плату можно рассчитать дополнительную заработную плату.

Дополнительная заработная плата – заработная плата, начисленная рабочим и служащим не за фактически выполненные работы или проработанное время, а в соответствии с действующим законодательством включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей). Дополнительная ЗП определяется по нормативу в процентах к основной заработной плате по формуле:

 (6.6)

где *З*Дi– дополнительная заработная плата исполнителей на конкретное ПС (руб.); *Н*Д – норматив дополнительной заработной платы рассчитывается по формуле:



Налоги, рассчитываемые от фонда оплаты труда, определяются в соответствии с действующими законодательными актами по нормативам, в процентном отношении к сумме основной и дополнительной заработной платы, относимой на ПС (Не) и рассчитывается по формуле:

 (6.7)

где *Н*не – норматив налога, уплачиваемого единым платежом (%).



Затраты, связанные с обеспечением работ оборудованием, рассчитаем как амортизационные отчисления, т.к. оборудование специально не закупалось.

При создании программного продукта используется стационарный компьютер (Intel Core i5-4690k , 8GB DDR3, DVD-Multi Recorder).

Закупочная цена ПЭВМ составила – 1500 руб. Примем норму амортизации на технические средства На = 20%, на основании трёх лет.

Для создания пакета программ, являющегося конечным результатом исследований, применялось следующее программное обеспечение:

* платформа Windows 10.

Общая стоимость программного обеспечения *Ц*по составляет 280 рублей.

Общую стоимость технических средств и программного обеспечения *Цо*, руб., определим по формуле:

*Ц*о *= Ц*тс*+Ц*по (6.8)

*Цо* = 1500+280=1780 руб.

Годовые амортизационные отчисления на полное восстановление технических средств и программного обеспечения *Ао*, руб., рассчитываются по формуле:

*А*о *= Ц*о*\*Н*а (6.9)

*А*о = 1780\*0,2=356 руб.

Амортизационные отчисления *А*п, руб. за период создания программного продукта определим по формуле:

*А*п *= А*о*/К*рг*\*К*дн (6.10)

где *К*дн = 59 дн. – количество отработанных дней;

*К*рг = 280 дн. – количество рабочих дней в году.

*А*п = 356\*60/280 = 75,01 руб.

# Расчёт расходов на энергопотребление

Так же необходимо рассчитать расходы электроэнергии. ПЭВМ, на которой была разработана программа, является потребителем электрической энергии сети переменного тока, напряжением 220 В.

Расход денежных средств *Р*э, руб., связанный с энергопотреблением технических средств можно вычислить по формуле:

*Р*э*=К*дн*\*В*раб*\* М*с *\* Ц*эн (6.11)

где *К*дн– период написания программы, дн.; *В*раб – длительность рабочей смены, ч.;

*М*с – мощность, потребляемая техническими средствами, кВт·ч;

*Ц*эн – стоимость электроэнергии по действующим тарифам (0,1013).

*Р*э = 59\*6\*0,2\*0,1013=7,17 руб.

# Общая сумма расходов по всем статьям сметы

Общая сумма расходов по всем статьям сметы (*СРi*) на ПС рассчитывается по формуле:

 (6.12)



# Расходы по статье «Прочие затраты»

Расходы по статье «Прочие затраты» определяются по нормативу, разрабатываемому в целом по научной организации, в процентах к основной заработной плате (см. формулу 6.13)

 (6.13)

где *Н*ПЗ – норматив прочих затрат в целом по научной организации.



# Рентабельность и прибыль

Рентабельность и прибыль **-** определяются исходя из результатов анализа рыночных условий, переговоров с заказчиком (потребителем) и согласования с ним отпускной цены, включающей дополнительно налог на добавленную стоимость. Рассчитывается по формуле:

 (6.14)

где *П*р – прибыль от реализации ПС заказчику (руб.);

*У*пр – уровень рентабельности ПС (%);

*С*п – себестоимость ПС (руб.).



# Прогнозируемая цена ПС без налогов

Прогнозируемая цена ПС без налогов (*Ц*п) рассчитывается, как сумма расходов по всем статьям смены и прибыли.

Рассчитывается по формуле:

. (6.15)



Необходимо рассчитать налог на добавленную стоимость. Поскольку мы уже рассчитали прогнозируемую стоимость и нам известно, что % НДС = 20% то расчет проведем по формуле:

 (6.16)

где *С*тндс – норматив НДС(%).



# Прогнозируемая отпускная цена

Прогнозируемая отпускная цена рассчитывается, как сумма прогнозируемой цены ПС без налогов и налог на добавленную стоимость(НДС).

Рассчитывается по формуле:

 (6.17)

*Ц*о*=* 871,29 +174,39 = 1045,68 *руб.*

# Расчет чистой прибыли

Имея все необходимые данные, рассчитаем чистую прибыль.

Рассчитывается по формуле:

 (6.18)

где *%Н*п – 18%.



*Ч*п *= П*р *– Н*п (6.19)

*Ч*п = 145,32 – 26,16 = 119,16 руб.

# Выводы по организационно-экономической части дипломного проекта

После произведенных расчетов, было выявлено, что разработка программного продукта составила 59 дней. Основная и дополнительная заработная плата работника составляет – 407,1 руб. Обязательные отчисления в ФСЗН составляет – 138,41 руб. Амортизационные отчисления за период создания программного обеспечения составляет – 75,01 руб. Расходы на энергопотребление, а также расходы на прочие затраты составляют соответственно – 7,17 руб. и 53,1 руб. Чистая прибыль за разработку программного обеспечения составляет – 119,16 руб. Отпускная цена программного продукта – 1045,68. Данный реализованный проект выгодный.

# ОХРАНА ТРУДА И ОКРУЖАЮЩЕЙ СРЕДЫ

# Правовые, нормативные, социально-экономические и организационные вопросы охраны труда

Охрана труда – это система обеспечения безопасности жизни и здоровья работников в процессе трудовой деятельности, включающая в себя правовые, социально-экономические, организационно-технические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия и средства.

Здоровые и безопасные условия труда способствуют повышению производительности, удовлетворенности  работников своим  трудом, созданию хорошего психологического климата в трудовых коллективах, что ведет к снижению текучести кадров, созданию стабильных трудовых коллективов [8].

Недостатки в работе по охране труда обуславливают значительные экономические потери. Заболеваемость и травматизм работников, затраты на компенсации за работу в неблагоприятных условиях труда приводят к ухудшению экономических результатов работы предприятия.

Основные законодательные акты регулирующие охрану труда: Конституция, закон Республики Беларусь «Об охране труда», Трудовой кодекс Республики Беларусь, законы Республики Беларусь, декреты президента, постановления правительства и другие.

Согласно законодательству, каждый работник имеет право на благоприятные условия труда, а работодатель обязан создать эти условия.

При работе с видеодисплейным терминалом (далее ВДТ) или персональной электро-вычислительной машиной (далее ПЭВМ) происходит воздействие вредных факторов (излучение, блики, шум, зрительное напряжение, нервно-эмоциональные перегрузки и др.) на организм работника, для предотвращения и минимизации их воздействия необходимо организовать рабочее место и труд работника в соответствии с Санитарными нормами и правилами, а также с правилами безопасности. Правильная организация рабочего места увеличивает производительность труда.

# Требования к помещениям для эксплуатации ВДТ и ПЭВМ

Помещения для эксплуатации ВДТ и ПЭВМ должны иметь естественное и искусственное освещение.

Запрещается выполнение основной работы с использованием ВДТ и ПЭВМ на постоянных рабочих местах без естественного освещения, если это не обусловлено технологическим процессом.

Естественное освещение на рабочих местах с ВДТ и ПЭВМ должно осуществляться через световые проемы, ориентированные преимущественно на север, северо-восток, восток, запад или северо-запад и обеспечивать коэффициент естественной освещенности не ниже 1,5 %.

Оконные проемы должны быть оборудованы регулируемыми устройствами типа жалюзи, занавесей, внешних козырьков и другое.

Площадь одного рабочего места для пользователей ВДТ и ПЭВМ на базе плоских дискретных экранов (жидкокристаллические, плазменные и другое) должна составлять не менее 4,5 м2.

Площадь одного рабочего места для пользователей ВДТ и ПЭВМ на базе электронно-лучевой трубки (далее – ЭЛТ) должна составлять не менее 6 м2.

Запрещается размещать рабочие места с ВДТ и ПЭВМ на расстоянии менее 10 м от силовых кабелей, вводов и высоковольтных трансформаторов.

Помещения, в которых для работы используются преимущественно ВДТ и ПЭВМ, не должны граничить с помещениями, в которых уровни шума и вибрации превышают нормируемые значения для данной категории проводимых в них работ и их типа. А так же поверхность пола должна быть ровной, нескользкой, без выбоин, удобной для очистки и влажной уборки, обладать антистатическими свойствами. Внутреняя отделка должна осуществлятся при помощи диффузно отражающих материалов с коэффициентом отражения для потолка – 0,7-0,8; для стен – 0,5-0,6; для пола – 0,3-0,5.

Помещения с ВДТ и ПЭВМ должны оборудоваться системами отопления, кондиционирования воздуха или эффективной приточно-вытяжной вентиляцией.

Помещения, где размещаются рабочие места с ВДТ и ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации[11].

# Требования к производственному оборудованию и рабочим местам

Оборудование должно удовлетворять следующим требованиям:

* конструкция ВДТ и ПЭВМ должна обеспечивать возможность поворота корпуса в горизонтальной и вертикальной плоскости с фиксацией в заданном положении для обеспечения фронтального наблюдения экрана, должна предусматривать регулирование яркости и контрастности;
* дизайн ВДТ и ПЭВМ и периферийных устройств должен предусматривать окраску в мягких тонах, иметь матовую поверхность, и не иметь блестящих деталей, создающих блики;
* ВДТ и ПЭВМ и периферийные устройства подлежат государственной санитарно-гигиенической экспертизе на соответствие требованиям;
* допустимые визуальные параметры устройств отображения ВДТ и ПЭВМ, а также уровни физических факторов (уровни вибрации, ультрафиолетового, инфракрасного, мягкого рентгеновского излучения и др.) должны быть в пределах допустимых уровней.

Высота рабочей поверхности стола для взрослых пользователей должна регулироваться в пределах 680-800 мм. При отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм.

При размещении нескольких рабочих мест с ВДТ и ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора) должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. Экран видеомонитора должен находиться на расстоянии 600-700 мм от глаз пользователя, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов. Клавиатуру следует располагать на поверхности стола на расстоянии 100-300 мм от края, обращенного к взрослому пользователю, или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Поверхность сиденья, спинки и других элементов рабочего стула должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений. Не допускается использование табуреток, скамеек без опоры вместо стульев.

# Обеспечение электробезопасности при работе с ВДТ и ПЭВМ

Рабочее помещение с ВДТ и ПЭВМ относится к классу помещений без повышенной опасности с точки зрения поражения электрическим током. Безопасность для работников в помещениях с ВДТ и ПЭВМ обеспечивается следующим образом:

* применение надлежащей изоляции;
* соблюдение соответствующих расстояний до токоведущих частей, ограничение доступа к ним;
* заземления или зануления корпусов электрооборудования, которые могут оказаться под напряжением вследствие повреждения изоляции.

Зоной повышенной электроопасности являются места подключения электроприборов. Нередко подключающие розетки располагают на полу, что недопустимо. Часто совершается другая ошибка – перегрузка розеток по мощности, и, как следствие, происходит нарушение изоляции, приводящее к короткому замыканию.

Для предотвращения поражения электрическим током не допускается:

* включение сильно охлаждённого (принесённого в зимнее время с улицы) оборудования;
* производить влажную уборку при включённом оборудовании;
* при включённом питании прикасаться к панелям с разъёмами оборудования, разъёмам питающих и соединительных кабелей, экрану монитора.

# Требования производственной санитарии

Производственная санитария – это система организационных мероприятий и технических средств, предотвращающих или уменьшающих воздействие на работающих вредных производственных факторов.

Основными вредными факторами при работе с ВДТ, ЭВМ и ПЭВМ являются:

* напряжение зрения, памяти и внимания;
* длительное статическое напряжение;
* монотонность труда;
* эмоциональные перегрузки;
* запылённости воздуха рабочей зоны;
* повышенные уровни электромагнитного, инфракрасного и ультрафиолетового излучения, шума и вибрации;
* повышенный или пониженный уровень освещения рабочей зоны.

Для снижения уровня вредных факторов, перечисленных выше, следует обеспечить в холодный период года температуру 22-24 °С, влажность воздуха 40-60 %, скорость движения воздуха 0.1 м/с – 0.3 м/с, а для теплого периода года температуру 23-25 °С, влажность воздуха 40-60 %, скорость движения воздуха 0.1 м/с – 0.3 м/с.

В связи с тем, что состояние производственного воздуха в значительной мере зависит от эффективности его обмена, необходимо уделять вентиляции помещения.

В соответствии с требованиями действующего законодательства работники должны обеспечиваться гардеробными, умывальниками, душевыми, помещениями для личной гигиены женщин, туалетами.

В помещениях, оборудованных ВДТ, ЭВМ и ПЭВМ, должна проводиться ежедневная влажная уборка и систематическое проветривание после каждого часа работы с ВДТ, ЭВМ и ПЭВМ.

Нормирование параметров микроклимата в рабочей зоне производится в зависимости от времени года, категории работ по энергозатратам и избытку явного тепла.

Содержание вредных химических веществ в воздухе помещений, предназначенных для использования ВДТ, ЭВМ и ПЭВМ не должно превышать предельно допустимых среднесуточных концентраций для атмосферного воздуха, в соответствии с нормативами предельно допустимых концентраций, загрязняющих веществ в атмосферном воздухе и ориентировочно безопасных уровней воздействия загрязняющих веществ в атмосферном воздухе населенных пунктов.

Для оптимизации микроклимата в помещении с ВДТ, ЭВМ и ПЭВМ используется:

* кондиционер – устройство для поддержания оптимальных климатических условий;
* увлажнитель воздуха – устройство для повышения влажности воздуха.
* вентиляционная система – совокупностью устройств для обработки, транспортирования, подачи и удаления воздуха;
* система отопления – комплекс устройств, выполняющих функцию отопления;

По принципу организации производственное освещение подразделяется на естественное, искусственное и совмещенное.

Рабочие столы следует размещать таким образом, чтобы экраны ВДТ, ЭВМ или ПЭВМ были ориентированы боковой стороной к световым проемам (исключение составляет периметральная расстановка рабочих мест), чтобы естественный свет падал преимущественно слева.

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500 люкс. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 люкс[11].

Необходимо ограничивать отраженную блесткость на рабочих поверхностях (экран, стол, клавиатура и другое) за счет правильного выбора типов светильников и расположения рабочих мест по отношению к источникам естественного и искусственного освещения, при этом яркость бликов на экране ВДТ и ПЭВМ не должна превышать 40 кд/м2.

Искусственное освещение в помещениях для эксплуатации ВДТ, ЭВМ и ПЭВМ должно осуществляться системой общего равномерного освещения. В административных помещениях в случаях преимущественной работы с документами следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Также для обеспечения нормируемых значений освещенности в помещениях для использования ВДТ, ЭВМ и ПЭВМ следует проводить чистку стекол оконных рам и светильников со своевременной заменой перегоревших ламп.

Звукоизоляция ограждающих конструкций помещений с ВДТ, ЭВМ и ПЭВМ должна обеспечивать нормируемые параметры шума в них.

При выполнении основной работы на ВДТ, ЭВМ и ПЭВМ (дис­петчерские, операторские, расчетные кабины и посты управления, залы вычислительной техники и др.), уровень шума на рабочем месте не должен превышать 50 дБ [11].

Для снижения шума применяют различные методы коллективной защиты: уменьшение уровня шума в источнике его возникновения; рациональное размещение оборудования; борьба с шумом на путях его распространения, в том числе изменение направленности излучения шума, использование средств звукоизоляции, звукопоглощение и установка глушителей шума, в том числе акустическая обработка поверхностей помещения.

Для ослабления передачи вибрации от источников ее возникновения полу, рабочему месту, сиденью, рукоятке и так далее, широко применяют методы виброизоляции в виде виброизоляторов из резины, пробки, войлока, асбеста, стальных пружин.

Важным фактором для снижения опасного воздействия вибрации на организм человека является правильная организация режима труда и отдыха, постоянное медицинское наблюдение за состоянием здоровья, лечебно-профилактические мероприятия.

Допустимые уровни напряженности (плотности потока мощности) электромагнитных полей, излучаемых клавиатурой, системным блоком, компьютерной мышью, беспроводными системами передачи информации на расстояния и иными вновь разработанными устройствами в зависимости от основной рабочей частоты изделия, не должны превышать значений.

Для того, чтобы избежать вредного влияния электромагнитного излучения, следует соблюдать следующие правила:

* использовать жидкокристаллический монитор, поскольку его излучение значительно меньше, чем у распространённых ЭЛТ мониторов (монитор с электроннолучевой трубкой);
* в связи с тем, что электромагнитное излучение от стенок монитора намного больше, по возможности монитор поставить в угол, так что бы излучение поглощалось стенами;
* выполнять ежедневную влажную уборку помещения;
* проветривание помещений, а в теплое время года работа с открытыми фрамугами;
* экран монитора должен быть удален от глаз минимум на 50-60 сантиметров.
* не игнорировать регламентированные перерывы.

# Организация режима труда и отдыха при работе с ВДТ и ПЭВМ

Для обеспечения оптимальной работоспособности и сохранения здоровья профессиональных пользователей, на протяжении рабочего дня (смены) должны устанавливаться регламентированные перерывы.

К работе допускаются только те, кто прошёл медицинскую комиссию и не имеет никаких противопоказаний.

Режимы труда и отдыха при работе с ВДТ и ПЭВМ должны определяться видом и категорией трудовой деятельности.

Виды трудовой деятельности разделяются на 3 группы:

* группа А – работа по считыванию информации с экрана ВДТ, ЭВМ или ПЭВМ с предварительным запросом;
* группа Б – работа по вводу информации;
* группа В – творческая работа в режиме диалога с ЭВМ.

Вид трудовой деятельности относится к группе В – творческая работа в режиме диалога с ЭВМ.

Продолжительность непрерывной работы с ВДТ и ПЭВМ без регламентированного перерыва не должна превышать двух часов.

При восьмичасовом рабочем дне (смене) и работе на ВДТ, ЭВМ и ПЭВМ регламентированные перерывы следует устанавливать:

* для I категории работ – через 2 часа от начала рабочего дня (смены) и через 2 часа после обеденного перерыва продолжительностью 15 минут каждый;
* для II категории работ – через 2 часа от начала рабочего дня (смены) и через 1,5-2 часа после обеденного перерыва продолжительностью 15 минут каждый или продолжительностью 10 минут через каждый час работы;
* для III категории – через 1,5-2 часа от начала рабочего дня (смены) и через 1,5-2 часа после обеденного перерыва продолжительностью 20 минут каждый или продолжительностью 15 минут через каждый час работы.

При двенадцатичасовом рабочем дне (смене) регламентированные перерывы должны устанавливаться в первые 8 часов работы аналогично перерывам при восьмичасовом рабочем дне (смене), а в течении последних 4 часов работы, независимо от категории и вида работ, каждый час продолжительностью 15 минут.

# Обеспечение пожарной безопасности

Пожарная безопасность – состояние объекта, при котором с регламентирующей вероятностью исключается возможность возникновения и развития пожара, а также обеспечивается защита людей  и материальных ценностей от воздействия опасных факторов.

Воспламенившаяся электронная техника, находится под напряжением и следовательно имеет электронные компаненты, поэтому для устранения пламени используются огнетушители, предназначенные для устранения пламени на объектах, которые находятся под напряжением. Для устранения пламени используются углекислотные огнетушители типов ОУ-2, ОУ-5, ОУ-8.

Достоинством углекислотного огнетушителя, является высокая эффективность тушения пожара и сохранность электронного оборудования.

Обеспечение пожарной безопасности является обязанностью руководителей, соответствующих должностных лиц и работников, а также граждан. Эти обязанности должны быть отражены в соответствующих инструкциях [6].

Для обеспечения пожарной безопасности на рабочем месте необходимо соблюдать следующие требования:

* необходимо наличие сигнализации для звукового оповещения о пожаре, средств связи для вызова пожарной службы;
* у телефонных аппаратов на видных местах должны быть вывешены таблички с указанием номера вызова пожарных аварийно-спасательных подразделений, а также подробный план этажа здания на случай экстренной эвакуации;
* проходы, подходы к средствам пожаротушения, места включения и выключения электрооборудования не должны быть загромождены прочими вещами;
* все работники обязаны знать правила и порядок применения первичных средств пожаротушения и меры безопасности при пользовании ими;
* необходимо наличие первичных средств пожаротушения (углекислотный огнетушитель).

Каждый работник независимо от занимаемой должности обязан знать и выполнять правила пожарной безопасности, не допускать действий, которые могут привести к возникновению пожара. В связи с этим со всеми работниками должен проводиться вводный, первичные и повторные на рабочем месте инструктажи по пожарной безопасности (повторный не реже 1 раза в 6 месяцев) [10].

# Обеспечение охраны окружающей среды

Основные принципы охраны окружающей среды допускают воздействие предприятий на природную среду, исходя из требований в области охраны окружающей среды. При этом снижение негативного воздействия на окружающую среду должно достигаться на основе использования наилучших существующих технологий с учетом экономических и социальных факторов.

Технологический процесс, осуществляемый при работе с ВДТ и ПЭВМ, не должен наносить вреда окружающей среде. С этой целью при производстве работ необходимо обязательно предусматривать комплекс соответствующих мероприятий применительно к данной работе [7].

Переработка устаревшей электроники требует значительных усилий и представляет собой немалую проблему. Следует бережно относиться к оборудованию, и в случае поломки отправить на ремонт, а не заменять новым.

В число конкретных мер по улучшению охраны окружающей среды входят:

* меры по экономии энергии – замена энергоемких устройств, использование энергоэффективных компьютеров и мониторов или оптимизированного освещения в административном помещении, как, собственно, и в производственном;
* правильная сортировка отходов с соблюдением строгих норм, превышающих требования законодательства и утилизация устаревшей сложной техники;
* создание производственно-материальной базы для такой работы;
* повышение уровня охвата населения системой экологического просвещения, образования, подготовки кадров.

При выполнениие всех мероприятий окружающая среда станет более благопрятна для работника. Воздействие вредний факторов на окружающую среду уменьшиться, следовательно качество работы улучшится.

# ЗАКЛЮЧЕНИЕ

Представленная версия приложения является законченным программным продуктом, но так как программное средство имеет возможность легкого расширения, имеется возможно дополнительно разработать модули для любого потребителя.

Разработанное программное средство автоматизирует ведение коммерции в сети интернет, с его помощью можно осуществить приобрести товар или услугу, так же можно просмотреть статистику заказов пользователей, закзать популярные товары которые формируются в автоматическом режиме. Позволяет облегчить работу по ведению коммерции в интернете.

В результате выполнения проекта были реализованные поставленные задачи, среди которых адаптивный, интуитивный и приятный интерфейс, а также набор функций для удобного использования интернет приложения.

Проект «Интернет приложение для ведения электронной коммерции» отлично подходит для решения поставленных задач.

Программный продукт работает под управлением любой операционной системы, главное требование – это наличие актуальной версии веб-браузера, а также интернет соединения.

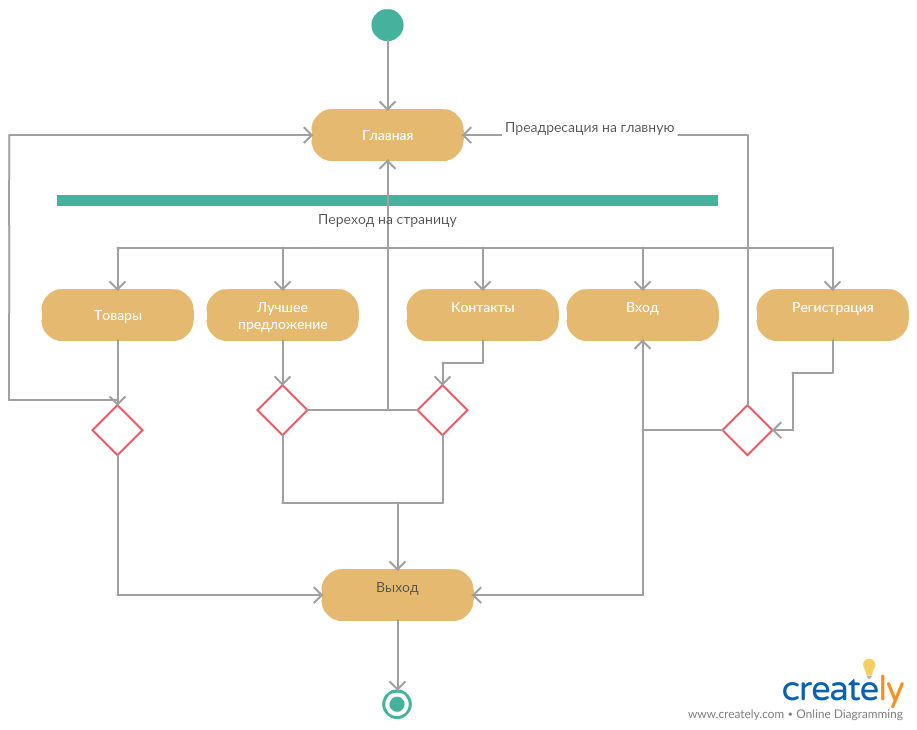
# ЛИТЕРАТУРА

* + - 1. HTML5.BY [Электронный ресурс]. Руководство по HTML. Режим доступа: http://html5.by.
      2. jquery-docs.ru [Электронный ресурс]. Руководство по JQUERY. Режим доступа: https://jquery-docs.ru.
      3. Вейнер П. – Языки программирования JAVA и JavaScript. – М: ЛОРИ, 2000.
      4. Гарнаев А. – Web-программирование на Java и JavaScript. – СПб.: БХВ Санкт-Перебург, 2002.
      5. Закон РБ «О пожарной безопасности».
      6. Закон РБ «Об охране окружающей среды».
      7. Закон РБ «Об охране труда».
      8. Лешек А. – Анализ и проектирование информационных систем с помощью UML 2.0. 3-е издание – Москва, Санкт-Петербург, Киев, 2008.
      9. Нормы пожарной безопасности РБ. Категорирование помещений, зданий и наружных установок по взрывопожарной и пожарной опасности.
      10. Постановление Министерства здравоохранения РБ №59 от 28 июня 2013 Санитарные нормы и правила «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами».
      11. Робин Никсон. – Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 4-е издание – БХВ-Петербург, 2016.
      12. Хабрахабр [Электронный ресурс]. Электронные данные. Режим доступа: https://habrahabr.ru.

# Приложение А

**Диаграмма деятельности**

*(обязательное)*

****

# Приложение Б

**Диаграмма последовательности**

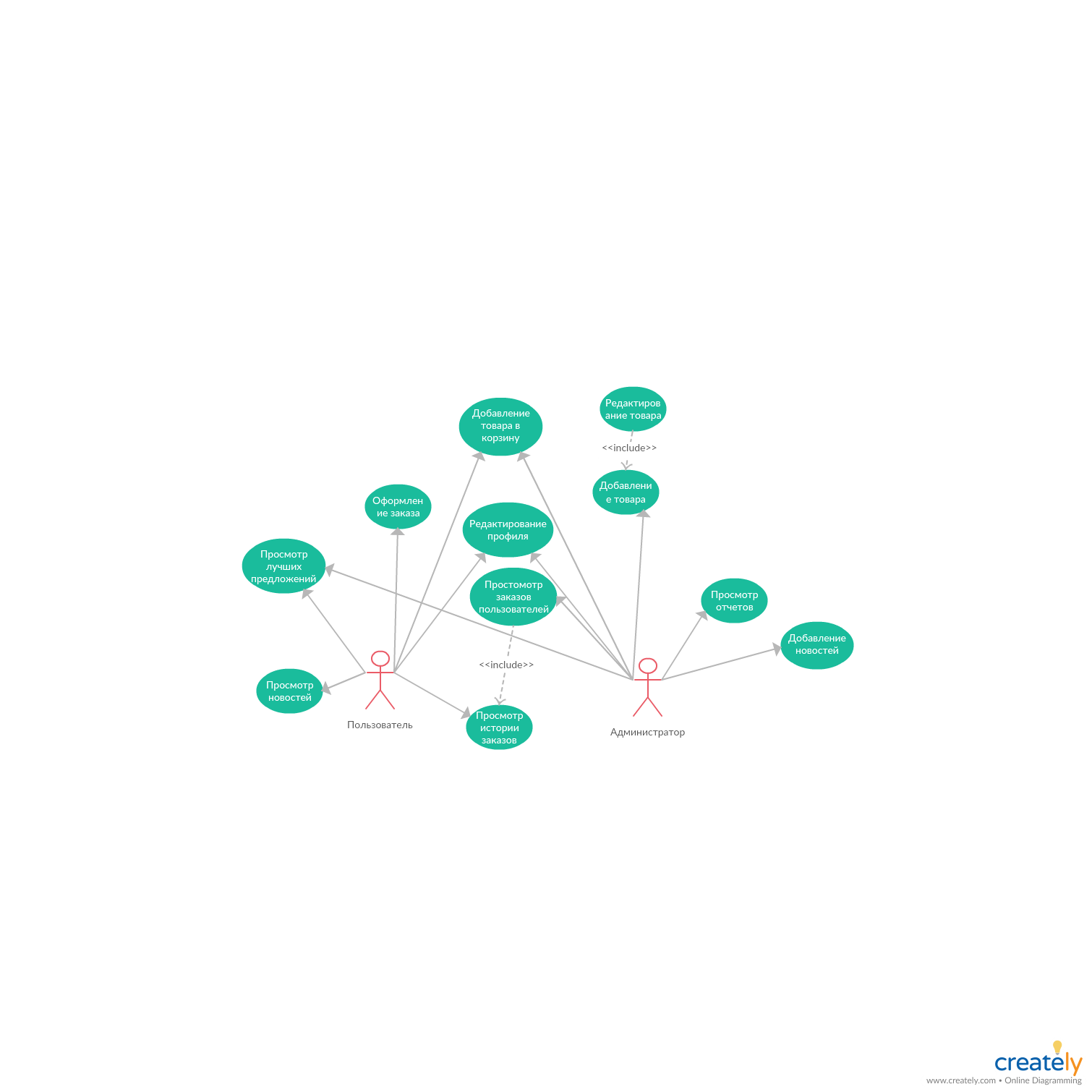
*(обязательное)*

# C:\Users\Администратор\Downloads\Электронная коммерция (3).png

# Приложение В

**Диаграмма вариантов использования**

*(обязательное)*

****

# Приложение Г

**Листинг программы**

*(обязательное)*

package org.solteh.service;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.stereotype.Service;  
  
@Service  
public class SecurityServiceImpl implements SecurityService {  
 private static final Logger logger = LoggerFactory.getLogger(SecurityServiceImpl.class);  
 private final AuthenticationManager authenticationManager;  
 private final UserDetailsService userDetailsService;  
  
 @Autowired  
 public SecurityServiceImpl(AuthenticationManager authenticationManager, UserDetailsService userDetailsService) {  
 this.authenticationManager = authenticationManager;  
 this.userDetailsService = userDetailsService;  
 }  
  
 @Override  
 public String findLoggedInUsername() {  
 Object userDetails = SecurityContextHolder.getContext().getAuthentication().getDetails();  
 if (userDetails instanceof UserDetails) {  
 return ((UserDetails) userDetails).getUsername();  
 }  
 return null;  
 }  
  
 @Override  
 public void autologin(String username, String password) {  
 UserDetails userDetails = userDetailsService.loadUserByUsername(username);  
 UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken = new UsernamePasswordAuthenticationToken(userDetails, username, userDetails.getAuthorities());  
 authenticationManager.authenticate(usernamePasswordAuthenticationToken);  
 if (usernamePasswordAuthenticationToken.isAuthenticated()) {  
 SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);  
 logger.debug(String.format("Auto login %s successfully!", username));  
 }  
 }  
}

package org.solteh.service;  
  
import org.solteh.model.User;  
import org.solteh.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Primary;  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.HashSet;  
import java.util.Set;  
  
@Service  
@Primary  
public class UserDetailsServiceImpl implements UserDetailsService {  
 private final UserRepository userRepository;  
  
 @Autowired  
 public UserDetailsServiceImpl(UserRepository userRepository) {  
 this.userRepository = userRepository;  
 }  
  
 @Override  
 @Transactional(readOnly = true)  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
 User user = userRepository.findByUserName(username);  
  
 Set<GrantedAuthority> grantedAuthorities = new HashSet<>();  
 grantedAuthorities.add(new SimpleGrantedAuthority(user.getUserState().name()));  
  
 return new org.springframework.security.core.userdetails.User(user.getUserName(), user.getEncrytedPassword(), grantedAuthorities);  
 }  
}

package org.solteh.service;  
  
import org.solteh.model.User;  
import org.solteh.model.UserState;  
import org.solteh.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.stereotype.Service;  
  
@Service  
public class UserServiceImpl implements UserService {  
 private final UserRepository userRepository;  
 private final BCryptPasswordEncoder bCryptPasswordEncoder;  
  
 @Autowired  
 public UserServiceImpl(UserRepository userRepository, BCryptPasswordEncoder bCryptPasswordEncoder) {  
 this.userRepository = userRepository;  
 this.bCryptPasswordEncoder = bCryptPasswordEncoder;  
 }  
  
 @Override  
 public void save(User user) {  
 user.setEncrytedPassword(bCryptPasswordEncoder.encode(user.getEncrytedPassword()));  
 user.setUserState(UserState.ROLE\_EMPLOYEE);  
 userRepository.save(user);  
 }  
  
 @Override  
 public User findByUsername(String username) {  
 return userRepository.findByUserName(username);  
 }  
}

package org.solteh.utils;  
  
import org.solteh.model.CartInfo;  
  
import javax.servlet.http.HttpServletRequest;  
  
public class Utils {  
 // Products in the cart, stored in Session.  
 public static CartInfo getCartInSession(HttpServletRequest request) {  
  
 CartInfo cartInfo = (CartInfo) request.getSession().getAttribute("myCart");  
  
  
 if (cartInfo == null) {  
 cartInfo = new CartInfo();  
  
 request.getSession().setAttribute("myCart", cartInfo);  
 }  
  
 return cartInfo;  
 }  
  
 public static void removeCartInSession(HttpServletRequest request) {  
 request.getSession().removeAttribute("myCart");  
 }  
  
 public static void storeLastOrderedCartInSession(HttpServletRequest request, CartInfo cartInfo) {  
 request.getSession().setAttribute("lastOrderedCart", cartInfo);  
 }  
  
 public static CartInfo getLastOrderedCartInSession(HttpServletRequest request) {  
 return (CartInfo) request.getSession().getAttribute("lastOrderedCart");  
 }  
  
}

package org.solteh.web.controller;  
  
import org.solteh.model.User;  
import org.solteh.service.SecurityService;  
import org.solteh.service.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
@Controller  
public class AuthorizationController {  
 private final UserService userService;  
 private final SecurityService securityService;  
  
  
 @Autowired  
 public AuthorizationController(UserService userService, SecurityService securityService) {  
 this.userService = userService;  
 this.securityService = securityService;  
 }  
  
 @GetMapping(value = "/registration")  
 public String registration(Model model) {  
 model.addAttribute("userForm", new User());  
 return "registration";  
 }  
  
 @PostMapping(value = "/registration")  
 public String registration(@ModelAttribute("userForm") User userForm) {  
 userService.save(userForm);  
 securityService.autologin(userForm.getUserName(), userForm.getEncrytedPassword());  
 return "redirect:/";  
 }  
  
 @RequestMapping(value = "/login")  
 public String login(Model model, String error, String logout) {  
 if (error != null)  
 model.addAttribute("error", "Введен не правильный логин или пароль!");  
  
 if (logout != null)  
 model.addAttribute("message", "Вы успешно вышли из системы.");  
 return "login";  
 }  
}

package org.solteh.web.controller;  
  
import org.solteh.model.News;  
import org.solteh.model.Product;  
import org.solteh.repository.NewsRepository;  
import org.solteh.repository.ProductRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.PageImpl;  
import org.springframework.data.domain.PageRequest;  
import org.springframework.data.domain.Sort;  
import org.springframework.data.domain.Sort.Direction;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestParam;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.math.BigDecimal;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
@Controller  
public class DefaultController {  
  
 private final ProductRepository productRepository;  
 private final NewsRepository newsRepository;  
  
 @Autowired  
 public DefaultController(ProductRepository productRepository, NewsRepository newsRepository) {  
 this.productRepository = productRepository;  
 this.newsRepository = newsRepository;  
 }  
  
 @GetMapping("/403")  
 public String accessDenied() {  
 return "/403";  
 }  
  
 @GetMapping({"/topsales", "/topsales/{pageNumber}"})  
 public String getTopSales(@PathVariable(required = false) Integer pageNumber, Model model) {  
 Page<Product> page = null;  
 if (pageNumber == null || pageNumber == 0) {  
 pageNumber = 0;  
 page = productRepository.findTopSales(gotoCustomPage(0, 6, Direction.DESC, "p2.products"));  
 } else if (pageNumber < 0) {  
 page = productRepository.findTopSales(gotoCustomPage(0, 6, Direction.DESC, "p2.products"));  
 } else {  
 pageNumber = pageNumber - 1;  
 page = productRepository.findTopSales(gotoCustomPage(pageNumber, 6, Direction.DESC, "p2.products"));  
 }  
 if (page != null && page.getTotalPages() < pageNumber) {  
 page = productRepository.findTopSales(gotoCustomPage(page.getTotalPages(), 6, Direction.DESC, "p2.products"));  
 }  
 List<BigDecimal> maxRating = productRepository.getMaxAmountProductFromOrders();  
 if (maxRating.get(0) != null) {  
 model.addAttribute("maxRating", maxRating.get(0));  
 Map<String, Integer> ratings = getRatingsForProducts(page.getContent(), maxRating.get(0).intValue());  
 model.addAttribute("ratings", ratings);  
 }  
 model.addAttribute("parameterUrl", "topsales");  
 model.addAttribute("pageProducts", page);  
 model.addAttribute("products", page.getContent());  
 return "topsales";  
 }  
  
 private Map<String, Integer> getRatingsForProducts(List<Product> content, int i) {  
 Map<String, Integer> ratings = new HashMap<>();  
 for (Product pr : content) {  
 List<BigDecimal> returnedValue = productRepository.getSalesCountFromProduct(pr);  
 for (BigDecimal val : returnedValue) {  
 ratings.putIfAbsent(pr.getCode(), (int) (val.doubleValue() / i \* 100));  
 }  
 }  
 return ratings;  
 }  
  
 private PageRequest gotoPage(int page) {  
 return PageRequest.of(page, 6, Sort.Direction.ASC, "createDate");  
 }  
  
 private PageRequest gotoCustomPage(int page, int size, Sort.Direction sort, String... properties) {  
 return PageRequest.of(page, size, sort, properties);  
 }  
  
 @GetMapping("/")  
 public String home(Model model) {  
 PageImpl<Product> page = (PageImpl<Product>) productRepository.findAll(gotoCustomPage(0, 3, Direction.ASC, "createDate"));  
 List<Product> topSales = productRepository.find();  
 List<News> news = newsRepository.findTopNews();  
 model.addAttribute("products", topSales);  
 model.addAttribute("news", news);  
 return "index";  
 }  
  
 // Product List  
 @GetMapping({"/products", "/products/{pageNumber}"})  
 public String listProductHandler(@PathVariable(required = false) Integer pageNumber, Model model) {  
 PageImpl<Product> page = null;  
 if (pageNumber == null || pageNumber == 0) {  
 pageNumber = 0;  
 page = (PageImpl<Product>) productRepository.findAll(gotoPage(0));  
 } else if (pageNumber < 0) {  
 page = (PageImpl<Product>) productRepository.findAll(gotoPage(0));  
 } else {  
 pageNumber = pageNumber - 1;  
 page = (PageImpl<Product>) productRepository.findAll(gotoPage(pageNumber));  
 }  
 if (page != null && page.getTotalPages() < pageNumber) {  
 page = (PageImpl<Product>) productRepository.findAll(gotoPage(page.getTotalPages()));  
 }  
 model.addAttribute("parameterUrl", "products");  
 model.addAttribute("pageProducts", page);  
 model.addAttribute("products", page.getContent());  
 return "productList";  
 }  
  
 @GetMapping("/contacts")  
 public String getContactsPage() {  
 return "contact";  
 }  
  
 @GetMapping(value = {"/productImage"})  
 public void productImage(HttpServletRequest request, HttpServletResponse response, Model model,  
 @RequestParam("code") String code) throws IOException {  
 Product product = null;  
 if (code != null) {  
 product = this.productRepository.findByCode(code);  
 }  
 if (product != null && product.getImage() != null) {  
 response.setContentType("image/jpeg, image/jpg, image/png, image/gif");  
 response.getOutputStream().write(product.getImage());  
 }  
 response.getOutputStream().close();  
 }  
  
}

package org.solteh.web.controller;  
  
import org.solteh.model.SubscribeEmails;  
import org.solteh.repository.EmailsRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.mail.javamail.JavaMailSender;  
import org.springframework.mail.javamail.MimeMessageHelper;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
  
import javax.mail.internet.MimeMessage;  
import java.util.logging.Logger;  
  
@Controller  
public class EmailController {  
  
 private static final Logger logger = Logger.getLogger(EmailController.class.getName());  
 private final JavaMailSender javaMailSender;  
  
 private final EmailsRepository emailsRepository;  
  
 @Autowired  
 public EmailController(JavaMailSender javaMailSender, EmailsRepository emailsRepository) {  
 this.javaMailSender = javaMailSender;  
 this.emailsRepository = emailsRepository;  
 }  
  
 @PostMapping("/subscribe")  
 public String addEmailToSubscribe(@ModelAttribute("email") String email) {  
 try {  
 emailsRepository.save(new SubscribeEmails(email));  
 sendEmail(email);  
 logger.info("Email successfully subscribe and send!");  
 } catch (Exception ex) {  
 logger.warning("Error in sending email: " + ex);  
 }  
 return "redirect:/";  
 }  
  
  
 private void sendEmail(String email) throws Exception {  
 MimeMessage message = javaMailSender.createMimeMessage();  
 MimeMessageHelper helper = new MimeMessageHelper(message);  
  
 helper.setTo(email);  
 helper.setText("Вы успешно оформили подписку новостей, на портале электронной коммерции. ");  
 helper.setSubject("Подписка портал электронной коммерции");  
  
 javaMailSender.send(message);  
 }  
}

package org.solteh.web.controller;  
  
import org.solteh.model.News;  
import org.solteh.model.SubscribeEmails;  
import org.solteh.repository.EmailsRepository;  
import org.solteh.repository.NewsRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.mail.javamail.JavaMailSender;  
import org.springframework.mail.javamail.MimeMessageHelper;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
  
import javax.mail.internet.MimeMessage;  
import java.util.Date;  
import java.util.List;  
  
@Controller  
public class NewsController {  
  
 private final NewsRepository newsRepository;  
 private final EmailsRepository emailsRepository;  
 private final JavaMailSender javaMailSender;  
  
 @Autowired  
 public NewsController(NewsRepository newsRepository, EmailsRepository emailsRepository, JavaMailSender javaMailSender) {  
 this.newsRepository = newsRepository;  
 this.emailsRepository = emailsRepository;  
 this.javaMailSender = javaMailSender;  
 }  
  
 @GetMapping("/news/{id}")  
 public String getFullNews(@PathVariable long id, Model model) {  
 model.addAttribute("news", newsRepository.getOne(id));  
 return "fullnews";  
 }  
  
 @GetMapping("/news/remove/{id}")  
 public String removeNews(@PathVariable long id, Model model) {  
 newsRepository.deleteById(id);  
 return "redirect:/";  
 }  
  
 @GetMapping("/admin/news")  
 public String addNews(Model model) {  
 model.addAttribute("news", new News());  
 return "addnews";  
 }  
  
 @PostMapping("/admin/news")  
 public String saveNews(@ModelAttribute("news") News saveNews) throws Exception {  
 saveNews.setCreateDate(new Date());  
 newsRepository.save(saveNews);  
 sendEmail(emailsRepository.findAll(), saveNews);  
 return "redirect:/";  
 }  
  
 private void sendEmail(List<SubscribeEmails> emails, News news) throws Exception {  
 MimeMessage message = javaMailSender.createMimeMessage();  
 MimeMessageHelper helper = new MimeMessageHelper(message);  
 for (SubscribeEmails email : emails) {  
 helper.setTo(email.getEmail());  
 helper.setText(news.getText());  
 helper.setSubject("Новости на портале:" + news.getTitle());  
  
 javaMailSender.send(message);  
 }  
  
 }  
}

package org.solteh.web.controller;  
  
import org.solteh.repository.\*;  
import org.springframework.beans.factory.annotation.\*;  
import org.springframework.stereotype.\*;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.\*;  
  
@Controller  
public class ReportController {  
 @Autowired  
 private OrderRepository orderRepository;  
  
 @GetMapping("/admin/report")  
 public String showReport() {  
 return "report";  
 }  
  
 @GetMapping("/JsonDataOrderReport")  
 @ResponseBody  
 public List<Object[]> getOrderJsonData() {  
 List<Object[]> objects = orderRepository.findOrdersWithGroupByDate();  
 return objects;  
 }  
}

package org.solteh.web.controller;  
  
import org.solteh.model.CartInfo;  
import org.solteh.model.CartLineInfo;  
import org.solteh.model.Order;  
import org.solteh.model.OrderDetail;  
import org.solteh.model.Product;  
import org.solteh.model.User;  
import org.solteh.repository.OrderDetailRepository;  
import org.solteh.repository.OrderRepository;  
import org.solteh.repository.ProductRepository;  
import org.solteh.repository.UserRepository;  
import org.solteh.utils.Utils;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
  
import javax.servlet.http.HttpServletRequest;  
import java.util.Date;  
  
@Controller  
public class ShoppingCartController {  
 private final ProductRepository productRepository;  
 private final OrderRepository orderRepository;  
 private final UserRepository userRepository;  
 private final OrderDetailRepository orderDetailRepository;  
  
 @Autowired  
 public ShoppingCartController(ProductRepository productRepository, OrderRepository orderRepository, UserRepository userRepository, OrderDetailRepository orderDetailRepository) {  
 this.productRepository = productRepository;  
 this.orderRepository = orderRepository;  
 this.userRepository = userRepository;  
 this.orderDetailRepository = orderDetailRepository;  
 }  
  
 @GetMapping({"/buyProduct"})  
 public String listProductHandler(HttpServletRequest request,  
 @RequestParam(value = "code", defaultValue = "") String code) {  
  
 Product product = null;  
 if (code != null && code.length() > 0) {  
 product = productRepository.findByCode(code);  
 if (product != null) {  
 CartInfo cartInfo = Utils.getCartInSession(request);  
 cartInfo.addProduct(product, 1);  
 }  
 }  
  
 return "redirect:/shoppingCart";  
 }  
  
 @GetMapping({"/shoppingCartRemoveProduct"})  
 public String removeProductHandler(HttpServletRequest request, Model model, //  
 @RequestParam(value = "code", defaultValue = "") String code) {  
 Product product = null;  
 if (code != null && code.length() > 0) {  
 product = productRepository.findByCode(code);  
 }  
 if (product != null) {  
  
 CartInfo cartInfo = Utils.getCartInSession(request);  
  
 cartInfo.removeProduct(product);  
  
 }  
  
 return "redirect:/shoppingCart";  
 }  
  
  
 // GET: Show cart.  
 @GetMapping(value = {"/shoppingCart"})  
 public String shoppingCartHandler(HttpServletRequest request, Model model) {  
 CartInfo myCart = Utils.getCartInSession(request);  
  
 model.addAttribute("cartForm", myCart);  
 return "shoppingCart";  
 }  
  
 // POST: Update quantity for product in cart  
 @PostMapping(value = {"/shoppingCart"})  
 public String shoppingCartUpdateQty(HttpServletRequest request,  
 @ModelAttribute("cartForm") CartInfo cartForm) {  
  
 CartInfo cartInfo = Utils.getCartInSession(request);  
 if (cartInfo.getCartLines().isEmpty()) {  
 return "redirect:/shoppingCart";  
 }  
 cartInfo.updateQuantity(cartForm);  
  
 return "redirect:/shoppingCart";  
 }  
  
 // GET: Show information to confirm.  
 @GetMapping(value = {"/shoppingCartConfirmation"})  
 public String shoppingCartConfirmationReview(HttpServletRequest request, Model model) {  
 CartInfo cartInfo = Utils.getCartInSession(request);  
 //getUser and redirect to profile if not personal info  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 User user = userRepository.findByUserName(auth.getName());  
 if (user == null) {  
 return "redirect:/login";  
 }  
 if (!user.isCorrectPersonalInfo()) {  
 model.addAttribute("error", true);  
 return "redirect:/profile";  
 }  
 if (cartInfo == null || cartInfo.isEmpty()) {  
  
 return "redirect:/shoppingCart";  
 }  
 cartInfo.setEmail(user.getEmail());  
 cartInfo.setAddress(user.getAddress());  
 cartInfo.setName(user.getFio());  
 cartInfo.setPhone(user.getPhone());  
 model.addAttribute("cartInfo", cartInfo);  
 return "shoppingCartConfirmation";  
 }  
  
 // POST: Submit Cart (Save)  
 @PostMapping(value = {"/shoppingCartConfirmation"})  
 public String shoppingCartConfirmationSave(HttpServletRequest request) {  
 CartInfo cart = Utils.getCartInSession(request);  
  
 if (cart.isEmpty()) {  
  
 return "redirect:/shoppingCart";  
 }  
 try {  
 Order order = new Order();  
 OrderDetail detail = new OrderDetail();  
 for (CartLineInfo line : cart.getCartLines()) {  
 detail.addProduct(line.getProduct(), line.getQuantity());  
 }  
 detail.setAmount(cart.getAmountTotal());  
 orderDetailRepository.save(detail);  
 order.setDetails(detail);  
 order.setOrderDate(new Date());  
 order.setAmount(cart.getAmountTotal());  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 String name = auth.getName(); //get logged in username  
 order.setUser(userRepository.findByUserName(name));  
 order.setCustomerAddress(order.getUser().getAddress());  
 order.setCustomerEmail(order.getUser().getEmail());  
 order.setCustomerName(order.getUser().getFio());  
 order.setCustomerPhone(order.getUser().getPhone());  
 orderRepository.save(order);  
 cart.setOrderNum(order.getId());  
 } catch (Exception e) {  
  
 return "shoppingCartConfirmation";  
 }  
 // Remove Cart from Session.  
 Utils.removeCartInSession(request);  
  
 // Store last cart.  
 Utils.storeLastOrderedCartInSession(request, cart);  
  
 return "redirect:/shoppingCartFinalize";  
 }  
  
 @GetMapping(value = {"/shoppingCartFinalize"})  
 public String shoppingCartFinalize(HttpServletRequest request, Model model) {  
  
 CartInfo lastOrderedCart = Utils.getLastOrderedCartInSession(request);  
  
 if (lastOrderedCart == null) {  
 return "redirect:/shoppingCart";  
 }  
 model.addAttribute("lastOrderedCart", lastOrderedCart);  
 return "shoppingCartFinalize";  
 }  
}

package org.solteh.web.controller;  
  
import org.apache.commons.lang.exception.ExceptionUtils;  
import org.solteh.model.Order;  
import org.solteh.model.Product;  
import org.solteh.model.User;  
import org.solteh.repository.OrderRepository;  
import org.solteh.repository.ProductRepository;  
import org.solteh.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.validation.BindingResult;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.servlet.mvc.support.RedirectAttributes;  
  
import java.util.Date;  
import java.util.List;  
  
@Controller  
public class UserController {  
  
 private final OrderRepository orderRepository;  
  
 private final ProductRepository productRepository;  
  
 private final UserRepository userRepository;  
  
 @Autowired  
 public UserController(OrderRepository orderRepository, ProductRepository productRepository, UserRepository userRepository) {  
 this.orderRepository = orderRepository;  
 this.productRepository = productRepository;  
 this.userRepository = userRepository;  
 }  
  
 @GetMapping("/userOrder/{userId}")  
 public String getOrdersByUserId(@PathVariable int userId, Model model) {  
 List<Order> orders = orderRepository.findOrdersByUserId(userId);  
 model.addAttribute("orders", orders);  
 return "orderList";  
 }  
  
 @GetMapping(value = {"/profile"})  
 public String accountInfo(Model model, @RequestParam(value = "error", defaultValue = "false") boolean error) {  
 UserDetails userDetails = (UserDetails) SecurityContextHolder.getContext().getAuthentication().getPrincipal();  
 model.addAttribute("user", userRepository.findByUserName(userDetails.getUsername()));  
 model.addAttribute("error", error);  
 return "accountInfo";  
 }  
  
 @PostMapping(value = "/profile")  
 public String saveUser(@ModelAttribute("user") User remoteUser) {  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 String name = auth.getName(); //get logged in username  
 User user = userRepository.findByUserName(name);  
 user.setFio(remoteUser.getFio());  
 user.setAddress(remoteUser.getAddress());  
 user.setEmail(remoteUser.getEmail());  
 user.setPhone(remoteUser.getPhone());  
 userRepository.save(user);  
 return "redirect:/shoppingCart";  
 }  
  
 @GetMapping(value = {"/user/orders"})  
 public String orderList(Model model, //  
 @RequestParam(value = "page", defaultValue = "1") String pageStr) {  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 String name = auth.getName(); //get logged in username  
 User user = userRepository.findByUserName(name);  
 if (user == null) {  
 return "/403";  
 }  
 List<Order> orders = orderRepository.findOrdersByUserId(user.getId());  
 model.addAttribute("orders", orders);  
 return "orderList";  
 }  
  
 // GET: Show product.  
 @GetMapping(value = {"/admin/product"})  
 public String product(Model model, @RequestParam(value = "code", defaultValue = "") String code) {  
 Product product = null;  
 if (code != null && code.length() > 0) {  
 product = productRepository.findByCode(code);  
 }  
 if (product != null) {  
 model.addAttribute("productForm", product);  
 } else {  
 model.addAttribute("productForm", new Product());  
 }  
 return "product";  
 }  
  
 @GetMapping("/admin/users")  
 public String findAllUsers(Model model) {  
 model.addAttribute("users", userRepository.findAll());  
 return "users";  
 }  
  
 // POST: Save product  
 @PostMapping(value = {"/admin/product"})  
 public String productSave(Model model, //  
 @ModelAttribute("productForm") Product product, //  
 BindingResult result, //  
 final RedirectAttributes redirectAttributes) {  
  
 try {  
 Product newProduct = new Product();  
 newProduct.setCode(product.getCode());  
 newProduct.setDescription(product.getDescription());  
 newProduct.setImage(product.getFileData().getBytes());  
 newProduct.setName(product.getName());  
 newProduct.setPrice(product.getPrice());  
 newProduct.setCreateDate(new Date());  
 productRepository.save(newProduct);  
 } catch (Exception e) {  
 Throwable rootCause = ExceptionUtils.getRootCause(e);  
 String message = rootCause.getMessage();  
 model.addAttribute("errorMessage", message);  
 // Show product form.  
 return "product";  
 }  
  
 return "redirect:/products";  
 }  
  
 @GetMapping(value = {"/user/order"})  
 public String orderView(Model model, @RequestParam("orderId") Long orderId) {  
 Order order = null;  
 if (orderId != null) {  
 order = this.orderRepository.getOne(orderId);  
 }  
 if (order == null) {  
 return "redirect:/user/orderList";  
 }  
  
 model.addAttribute("order", order);  
  
 return "order";  
 }  
}

package org.solteh;  
  
import com.zaxxer.hikari.HikariDataSource;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.boot.autoconfigure.jdbc.DataSourceProperties;  
import org.springframework.boot.context.properties.ConfigurationProperties;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.context.annotation.Primary;  
import org.springframework.core.env.Environment;  
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;  
import org.springframework.orm.jpa.JpaVendorAdapter;  
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;  
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;  
import org.springframework.transaction.annotation.EnableTransactionManagement;  
  
import javax.sql.DataSource;  
import java.util.Properties;  
  
@Configuration  
@EnableJpaRepositories  
@EnableTransactionManagement  
public class JpaConfiguration {  
  
 private final Environment environment;  
  
 @Value("${datasource.sampleapp.maxPoolSize:100}")  
 private int maxPoolSize;  
  
 @Autowired  
 public JpaConfiguration(Environment environment) {  
 this.environment = environment;  
 }  
  
 /\*  
 \* Populate SpringBoot DataSourceProperties object directly from  
 application.yml  
 \* based on prefix.Thanks to .yml, Hierachical data is mapped out of  
 the box with matching-name  
 \* properties of DataSourceProperties object].  
 \*/  
 @Bean  
 @Primary  
 @ConfigurationProperties(prefix = "spring.datasource")  
 public DataSourceProperties dataSourceProperties() {  
 return new DataSourceProperties();  
 }  
  
 /\*  
 \* Configure HikariCP pooled DataSource.  
 \*/  
 @Bean  
 public DataSource dataSource() {  
 DataSourceProperties dataSourceProperties = dataSourceProperties();  
 HikariDataSource dataSource = org.springframework.boot.jdbc.DataSourceBuilder  
 .create(dataSourceProperties.getClassLoader())  
  
 .driverClassName(dataSourceProperties.getDriverClassName())  
 .url(dataSourceProperties.getUrl())  
 .username(dataSourceProperties.getUsername())  
 .password(dataSourceProperties.getPassword())  
 .type(HikariDataSource.class)  
 .build();  
 dataSource.setMaximumPoolSize(maxPoolSize);  
 return dataSource;  
 }  
  
 /\*  
 \* Entity Manager Factory setup.  
 \*/  
 @Bean  
 @Primary  
 public LocalContainerEntityManagerFactoryBean entityManagerFactory() {  
 LocalContainerEntityManagerFactoryBean factoryBean = new  
 LocalContainerEntityManagerFactoryBean();  
 factoryBean.setDataSource(dataSource());  
 factoryBean.setPackagesToScan("org.solteh");  
 factoryBean.setJpaVendorAdapter(jpaVendorAdapter());  
 factoryBean.setJpaProperties(jpaProperties());  
 return factoryBean;  
 }  
  
 /\*  
 \* Provider specific adapter.  
 \*/  
 @Bean  
 public JpaVendorAdapter jpaVendorAdapter() {  
 return new HibernateJpaVendorAdapter();  
 }  
  
 /\*  
 \* Here you can specify any provider specific properties.  
 \*/  
 private Properties jpaProperties() {  
 Properties properties = new Properties();  
 properties.put("hibernate.dialect",  
 environment.getRequiredProperty  
 ("spring.jpa.properties.hibernate.dialect")  
 );  
 return properties;  
 }  
}

package org.solteh;  
  
import org.slf4j.\*;  
import org.springframework.security.access.\*;  
import org.springframework.security.core.\*;  
import org.springframework.security.core.context.\*;  
import org.springframework.security.web.access.\*;  
import org.springframework.stereotype.\*;  
  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import java.io.\*;  
  
@Component  
public class MyAccessDeniedHandler implements AccessDeniedHandler {  
 private static Logger logger = LoggerFactory.getLogger(MyAccessDeniedHandler.class);  
  
 @Override  
 public void handle(HttpServletRequest request, HttpServletResponse response, AccessDeniedException accessDeniedException) throws IOException, ServletException {  
 Authentication auth  
 = SecurityContextHolder.getContext().getAuthentication();  
  
 if (auth != null) {  
 logger.info("Пользователь '" + auth.getName()  
 + "' попытался получить доступ к защищенному URL-адресу: "  
 + request.getRequestURI());  
 }  
  
 response.sendRedirect(request.getContextPath() + "/403");  
 }  
}

package org.solteh;  
  
import org.hibernate.SessionFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.Bean;  
import org.springframework.core.env.Environment;  
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;  
import org.springframework.jdbc.datasource.DriverManagerDataSource;  
import org.springframework.orm.hibernate5.HibernateTransactionManager;  
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
  
import javax.sql.DataSource;  
import java.util.Properties;  
  
@SpringBootApplication  
@EnableWebMvc  
@EnableJpaRepositories(basePackages = "org.solteh")  
public class SpringBootWebApplication {  
 private final Environment env;  
  
 @Autowired  
 public SpringBootWebApplication(Environment env) {  
 this.env = env;  
 }  
  
  
 public static void main(String[] args) {  
 SpringApplication.run(SpringBootWebApplication.class, args);  
 }  
  
 @Bean(name = "dataSource")  
 public DataSource getDataSource() {  
 DriverManagerDataSource dataSource = new DriverManagerDataSource();  
  
 // See: application.properties  
 dataSource.setDriverClassName(env.getProperty("spring.datasource.driver-class-name"));  
 dataSource.setUrl(env.getProperty("spring.datasource.url"));  
 dataSource.setUsername(env.getProperty("spring.datasource.username"));  
 dataSource.setPassword(env.getProperty("spring.datasource.password"));  
  
 System.out.println("## getDataSource: " + dataSource);  
  
 return dataSource;  
 }  
  
 @Autowired  
 @Bean(name = "sessionFactory")  
 public SessionFactory getSessionFactory(DataSource dataSource) throws Exception {  
 LocalSessionFactoryBean factoryBean = new LocalSessionFactoryBean();  
 // Package contain entity classes  
 factoryBean.setPackagesToScan("");  
 factoryBean.setDataSource(dataSource);  
 factoryBean.setHibernateProperties(getProperties());  
 factoryBean.afterPropertiesSet();  
 //  
 SessionFactory sf = factoryBean.getObject();  
 System.out.println("## getSessionFactory: " + sf);  
 return sf;  
 }  
  
 private Properties getProperties() {  
 Properties properties = new Properties();  
  
 // See: application.properties  
 properties.put("hibernate.dialect", env.getProperty("spring.jpa.properties.hibernate.dialect"));  
 properties.put("hibernate.show\_sql", env.getProperty("spring.jpa.show-sql"));  
 properties.put("hibernate.generate-ddl", env.getProperty("spring.jpa.generate-ddl"));  
 properties.put("hibernate.hbm2ddl.auto", env.getProperty("spring.jpa.hibernate.ddl-auto"));  
 properties.put("current\_session\_context\_class", //  
 env.getProperty("spring.jpa.properties.hibernate.current\_session\_context\_class"));  
 return properties;  
 }  
  
 @Autowired  
 @Bean(name = "transactionManager")  
 public HibernateTransactionManager getTransactionManager(SessionFactory sessionFactory) {  
 return new HibernateTransactionManager(sessionFactory);  
 }  
}

package org.solteh;  
  
import org.springframework.context.MessageSource;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.context.support.ReloadableResourceBundleMessageSource;  
import org.springframework.mail.javamail.JavaMailSender;  
import org.springframework.mail.javamail.JavaMailSenderImpl;  
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
  
import java.util.Properties;  
  
@Configuration  
public class SpringBootWebConfiguration implements WebMvcConfigurer {  
 private static final String[] CLASSPATH\_RESOURCE\_LOCATIONS = {  
 "classpath:/META-INF/resources/", "classpath:/resources/",  
 "classpath:/static/", "classpath:/public/"};  
  
 @Override  
 public void addResourceHandlers(ResourceHandlerRegistry registry) {  
 if (!registry.hasMappingForPattern("/webjars/\*\*")) {  
 registry.addResourceHandler("/webjars/\*\*").addResourceLocations(  
 "classpath:/META-INF/resources/webjars/");  
 }  
 if (!registry.hasMappingForPattern("/static/\*\*")) {  
 registry.addResourceHandler("/static/\*\*").addResourceLocations(  
 CLASSPATH\_RESOURCE\_LOCATIONS);  
 }  
 }  
  
 @Bean  
 public MessageSource messageSource() {  
 ReloadableResourceBundleMessageSource messageSource = new ReloadableResourceBundleMessageSource();  
 // Load file: validation.properties  
 messageSource.setBasename("classpath:validation");  
 messageSource.setDefaultEncoding("UTF-8");  
 return messageSource;  
 }  
  
 @Bean  
 public JavaMailSender getJavaMailSender() {  
 JavaMailSenderImpl mailSender = new JavaMailSenderImpl();  
 mailSender.setHost("smtp.gmail.com");  
 mailSender.setPort(587);  
  
 mailSender.setUsername("username@gmail.com");  
 mailSender.setPassword("emailpassword");  
  
 Properties props = mailSender.getJavaMailProperties();  
 props.put("mail.transport.protocol", "smtp");  
 props.put("mail.smtp.auth", "true");  
 props.put("mail.smtp.starttls.enable", "true");  
 props.put("mail.debug", "true");  
  
 return mailSender;  
 }  
  
}

package org.solteh;  
  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.ApplicationContextAware;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.ViewResolver;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
import org.thymeleaf.spring5.ISpringTemplateEngine;  
import org.thymeleaf.spring5.SpringTemplateEngine;  
import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;  
import org.thymeleaf.spring5.view.ThymeleafViewResolver;  
import org.thymeleaf.templatemode.TemplateMode;  
import org.thymeleaf.templateresolver.ITemplateResolver;  
  
@Configuration  
@ComponentScan("org.solteh")  
public class ThymeleafConfig implements ApplicationContextAware, WebMvcConfigurer {  
  
 private ApplicationContext applicationContext;  
  
 public void setApplicationContext(ApplicationContext applicationContext) {  
 this.applicationContext = applicationContext;  
 }  
  
 @Bean  
 public ViewResolver viewResolver() {  
 ThymeleafViewResolver resolver = new ThymeleafViewResolver();  
 resolver.setTemplateEngine(templateEngine());  
 resolver.setCharacterEncoding("UTF-8");  
 return resolver;  
 }  
  
 @Bean  
 public ISpringTemplateEngine templateEngine() {  
 SpringTemplateEngine engine = new SpringTemplateEngine();  
 engine.setEnableSpringELCompiler(true);  
 engine.setTemplateResolver(templateResolver());  
 return engine;  
 }  
  
 private ITemplateResolver templateResolver() {  
 SpringResourceTemplateResolver resolver = new SpringResourceTemplateResolver();  
 resolver.setApplicationContext(applicationContext);  
 resolver.setPrefix("/WEB-INF/templates/");  
 resolver.setTemplateMode(TemplateMode.HTML);  
 return resolver;  
 }  
}

package org.solteh;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
  
@Configuration  
@EnableWebSecurity  
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {  
 private final UserDetailsService userDetailsService;  
  
 @Autowired  
 public WebSecurityConfig(UserDetailsService userDetailsService) {  
 this.userDetailsService = userDetailsService;  
 }  
  
 @Bean  
 public BCryptPasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public AuthenticationManager customAuthenticationManager() throws Exception {  
 return authenticationManager();  
 }  
  
 @Autowired  
 public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {  
 // Setting Service to find User in the database.  
 // And Setting PassswordEncoder  
 auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());  
  
 }  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.authorizeRequests().antMatchers("/webjars/\*\*", "/static/\*\*").permitAll();  
  
 http.csrf().disable();  
 // Requires login with role ROLE\_EMPLOYEE or ROLE\_MANAGER.  
 // If not, it will redirect to /admin/login.  
 http.authorizeRequests().antMatchers("/user/orders", "/user/order", "/profile")//  
 .access("hasAnyRole('ROLE\_EMPLOYEE', 'ROLE\_MANAGER')");  
  
 // Pages only for MANAGER  
 http.authorizeRequests().antMatchers("/admin/product").access("hasRole('ROLE\_MANAGER')");  
  
 // When user login, role XX.  
 // But access to the page requires the YY role,  
 // An AccessDeniedException will be thrown.  
 http.authorizeRequests().and().exceptionHandling().accessDeniedPage("/403");  
  
 // Configuration for Login Form.  
 http.authorizeRequests().and().formLogin()//  
 .loginProcessingUrl("/j\_spring\_security\_check") // Submit URL  
 .loginPage("/login")//  
 .defaultSuccessUrl("/")//  
 .failureUrl("/login?error=true")//  
 .usernameParameter("userName")//  
 .passwordParameter("password")  
  
 // Configuration for the Logout page.  
 // (After logout, go to home page)  
 .and().logout().logoutUrl("/logout").logoutSuccessUrl("/");  
  
 }  
}

package org.solteh.model;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class CartInfo {  
  
 private final List<CartLineInfo> cartLines = new ArrayList<>();  
 private long orderNum;  
  
 private String name;  
 private String phone;  
 private String email;  
 private String address;  
  
 public List<CartLineInfo> getCartLines() {  
 return this.cartLines;  
 }  
  
 private CartLineInfo findLineByCode(String code) {  
 for (CartLineInfo line : this.cartLines) {  
 if (line.getProduct().getCode().equals(code)) {  
 return line;  
 }  
 }  
 return null;  
 }  
  
 public void addProduct(Product product, int quantity) {  
 CartLineInfo line = this.findLineByCode(product.getCode());  
  
 if (line == null) {  
 line = new CartLineInfo();  
 line.setQuantity(0);  
 line.setProduct(product);  
 this.cartLines.add(line);  
 }  
 int newQuantity = line.getQuantity() + quantity;  
 if (newQuantity <= 0) {  
 this.cartLines.remove(line);  
 } else {  
 line.setQuantity(newQuantity);  
 }  
 }  
  
 public void updateProduct(String code, int quantity) {  
 CartLineInfo line = this.findLineByCode(code);  
  
 if (line != null) {  
 if (quantity <= 0) {  
 this.cartLines.remove(line);  
 } else {  
 line.setQuantity(quantity);  
 }  
 }  
 }  
  
 public void removeProduct(Product product) {  
 CartLineInfo line = this.findLineByCode(product.getCode());  
 if (line != null) {  
 this.cartLines.remove(line);  
 }  
 }  
  
 public boolean isEmpty() {  
 return this.cartLines.isEmpty();  
 }  
  
 public int getQuantityTotal() {  
 int quantity = 0;  
 for (CartLineInfo line : this.cartLines) {  
 quantity += line.getQuantity();  
 }  
 return quantity;  
 }  
  
 public double getAmountTotal() {  
 double total = 0;  
 for (CartLineInfo line : this.cartLines) {  
 total += line.getAmount();  
 }  
 return total;  
 }  
  
 public void updateQuantity(CartInfo cartForm) {  
 if (cartForm != null) {  
 List<CartLineInfo> lines = cartForm.getCartLines();  
 for (CartLineInfo line : lines) {  
 this.updateProduct(line.getProduct().getCode(), line.getQuantity());  
 }  
 }  
  
 }  
  
 public long getOrderNum() {  
 return orderNum;  
 }  
  
 public void setOrderNum(long orderNum) {  
 this.orderNum = orderNum;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String lastname) {  
 this.name = lastname;  
 }  
  
 public String getPhone() {  
 return phone;  
 }  
  
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
}

package org.solteh.model;  
  
public class CartLineInfo {  
 private Product product;  
 private int quantity;  
  
 public CartLineInfo() {  
 this.quantity = 0;  
 }  
  
  
 public int getQuantity() {  
 return quantity;  
 }  
  
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 }  
  
 public double getAmount() {  
 return this.product.getPrice() \* this.quantity;  
 }  
  
 public Product getProduct() {  
 return product;  
 }  
  
 public void setProduct(Product product) {  
 this.product = product;  
 }  
}

package org.solteh.model;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.Lob;  
import javax.persistence.Table;  
import java.util.Date;  
  
@Entity  
@Table(name = "news")  
public class News {  
 @Id  
 @Column  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private long id;  
 @Lob  
 @Column  
 private String title;  
 @Lob  
 @Column  
 private String text;  
 @Column  
 private Date createDate;  
  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long id) {  
 this.id = id;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getText() {  
 return text;  
 }  
  
 public void setText(String text) {  
 this.text = text;  
 }  
  
 public Date getCreateDate() {  
 return createDate;  
 }  
  
 public void setCreateDate(Date createDate) {  
 this.createDate = createDate;  
 }  
}

package org.solteh.model;  
  
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.FetchType;  
import javax.persistence.ForeignKey;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.JoinColumn;  
import javax.persistence.ManyToOne;  
import javax.persistence.Table;  
import java.io.Serializable;  
import java.util.Date;  
  
  
/\*\*  
 \* When the ShoppingCart with the LineItems  
 \* are confirmed by the WebUser, an Order is created.&nbsp;  
 \*/  
@Entity  
@Table(name = "Orders")  
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})  
public class Order implements Serializable {  
  
 private static final long serialVersionUID = -1053096276411165964L;  
 @Id  
 @Column  
 @GeneratedValue(strategy = GenerationType.AUTO)  
 private Long id;  
  
 @Column(name = "Order\_Date", nullable = false)  
 private Date orderDate;  
  
 @Column(name = "Amount", nullable = false)  
 private double amount;  
  
 @Column(name = "Customer\_Name")  
 private String customerName;  
  
 @Column(name = "Customer\_Address")  
 private String customerAddress;  
  
 @Column(name = "Customer\_Email", length = 128)  
 private String customerEmail;  
  
 @Column(name = "Customer\_Phone", length = 128)  
 private String customerPhone;  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "ORDER\_DETAILS\_ID", nullable = false, //  
 foreignKey = @ForeignKey(name = "ORDER\_DETAILS\_FK"))  
 private OrderDetail details;  
  
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "USER\_ID", nullable = false, //  
 foreignKey = @ForeignKey(name = "ORDER\_USR\_FK"))  
 private User user;  
  
 public Order() {  
 details = new OrderDetail();  
 }  
  
 public Date getOrderDate() {  
 return orderDate;  
 }  
  
 public void setOrderDate(Date orderDate) {  
 this.orderDate = orderDate;  
 }  
  
 public double getAmount() {  
 return amount;  
 }  
  
 public void setAmount(double amount) {  
 this.amount = amount;  
 }  
  
 public String getCustomerName() {  
 return customerName;  
 }  
  
 public void setCustomerName(String customerName) {  
 this.customerName = customerName;  
 }  
  
 public String getCustomerAddress() {  
 return customerAddress;  
 }  
  
 public void setCustomerAddress(String customerAddress) {  
 this.customerAddress = customerAddress;  
 }  
  
 public String getCustomerEmail() {  
 return customerEmail;  
 }  
  
 public void setCustomerEmail(String customerEmail) {  
 this.customerEmail = customerEmail;  
 }  
  
 public String getCustomerPhone() {  
 return customerPhone;  
 }  
  
 public void setCustomerPhone(String customerPhone) {  
 this.customerPhone = customerPhone;  
 }  
  
 public OrderDetail getDetails() {  
 return details;  
 }  
  
 public void setDetails(OrderDetail details) {  
 this.details = details;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public User getUser() {  
 return user;  
 }  
  
 public void setUser(User user) {  
 this.user = user;  
 }  
}

package org.solteh.model;  
  
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;  
  
import javax.persistence.Column;  
import javax.persistence.ElementCollection;  
import javax.persistence.Entity;  
import javax.persistence.ForeignKey;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.JoinColumn;  
import javax.persistence.Table;  
import java.io.Serializable;  
import java.util.HashMap;  
import java.util.Map;  
  
@Entity  
@Table(name = "Order\_Details")  
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})  
  
public class OrderDetail implements Serializable {  
 private static final long serialVersionUID = -589747686685428188L;  
 @Id  
 @Column  
 @GeneratedValue(strategy = GenerationType.AUTO)  
 private long id;  
  
 @JoinColumn(name = "PRODUCT\_ID",  
 foreignKey = @ForeignKey(name = "ORDER\_DETAIL\_PROD\_FK"))  
 @ElementCollection  
 private Map<Product, Long> products = new HashMap<>();  
  
 @Column(name = "Amount")  
 private double amount;  
  
  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long id) {  
 this.id = id;  
 }  
  
 public double getAmount() {  
 return amount;  
 }  
  
 public void setAmount(double amount) {  
 this.amount = amount;  
 }  
  
 public Map<Product, Long> getProducts() {  
 return products;  
 }  
  
 public void setProducts(Map<Product, Long> products) {  
 this.products = products;  
 }  
  
 public void addProduct(Product product, long quantity) {  
 this.products.put(product, quantity);  
 }  
}

package org.solteh.model;  
  
import org.springframework.web.multipart.MultipartFile;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.Lob;  
import javax.persistence.Table;  
import javax.persistence.Temporal;  
import javax.persistence.TemporalType;  
import javax.persistence.Transient;  
import java.io.Serializable;  
import java.util.Date;  
  
@Entity  
@Table(name = "Products")  
public class Product implements Serializable {  
  
 private static final long serialVersionUID = -1000119078147252957L;  
  
 @Id  
 @Column(name = "Code", length = 20, nullable = false)  
 private String code;  
  
 @Column(name = "Name", length = 255, nullable = false)  
 private String name;  
  
 @Column(name = "Description")  
 private String description;  
  
 @Column(name = "Price", nullable = false)  
 private double price;  
  
 @Lob  
 @Column(name = "Image", length = Integer.MAX\_VALUE, nullable = true)  
 private byte[] image;  
  
 @Temporal(TemporalType.TIMESTAMP)  
 @Column(name = "Create\_Date", nullable = false)  
 private Date createDate;  
  
 @Transient  
 private MultipartFile fileData;  
  
 public static long getSerialVersionUID() {  
 return serialVersionUID;  
 }  
  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 public byte[] getImage() {  
 return image;  
 }  
  
 public void setImage(byte[] image) {  
 this.image = image;  
 }  
  
 public Date getCreateDate() {  
 return createDate;  
 }  
  
 public void setCreateDate(Date createDate) {  
 this.createDate = createDate;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String description) {  
 this.description = description;  
 }  
  
 public MultipartFile getFileData() {  
 return fileData;  
 }  
  
 public void setFileData(MultipartFile fileData) {  
 this.fileData = fileData;  
 }  
}

package org.solteh.model;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
@Entity  
@Table(name = "emails")  
public class SubscribeEmails {  
 @Id  
 @Column  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 @Column  
 private String email;  
  
 public SubscribeEmails(String email) {  
 this.email = email;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
}

package org.solteh.model;  
  
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.Enumerated;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.Table;  
import java.io.Serializable;  
  
@Entity  
@Table(name = "Users")  
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})  
public class User implements Serializable {  
 private static final long serialVersionUID = -812013033671051986L;  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 @Column(name = "User\_Name", length = 20, unique = true, nullable = false)  
 private String userName;  
  
 @Column(name = "Encryted\_Password", length = 128, nullable = false)  
 private String encrytedPassword;  
 @Column(name = "email", unique = true)  
 private String email;  
  
 @Enumerated  
 @Column(name = "User\_State")  
 private UserState userState;  
  
 @Column  
 private String fio;  
 @Column  
 private String address;  
 @Column  
 private String phone;  
  
 public String getUserName() {  
 return userName;  
 }  
  
 public void setUserName(String userName) {  
 this.userName = userName;  
 }  
  
 public String getEncrytedPassword() {  
 return encrytedPassword;  
 }  
  
 public void setEncrytedPassword(String encrytedPassword) {  
 this.encrytedPassword = encrytedPassword;  
 }  
  
 public UserState getUserState() {  
 return userState;  
 }  
  
 public void setUserState(UserState userState) {  
 this.userState = userState;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getFio() {  
 return fio;  
 }  
  
 public void setFio(String fio) {  
 this.fio = fio;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public String getPhone() {  
 return phone;  
 }  
  
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
  
 public boolean isCorrectPersonalInfo() {  
 return getAddress() != null && !getAddress().isEmpty() &&  
 getFio() != null && !getFio().isEmpty() &&  
 getEmail() != null && !getEmail().isEmpty() &&  
 getPhone() != null && !getPhone().isEmpty();  
 }  
}

package org.solteh.model;  
  
public enum UserState{  
 ROLE\_MANAGER,  
 ROLE\_EMPLOYEE  
}

# ===============================  
# = DATA SOURCE  
# ===============================  
spring.datasource.driver-class-name=com.mysql.jdbc.Driver  
spring.datasource.url=jdbc:mysql://localhost:3306/ecommerce?useSSL=false  
spring.datasource.username=root  
spring.datasource.password=root  
spring.datasource.initialization-mode=always  
# ===============================  
# = JPA / HIBERNATE  
# ===============================  
spring.jpa.show-sql=true  
spring.jpa.generate-ddl=true  
spring.jpa.hibernate.ddl-auto=create-drop  
spring.jpa.properties.hibernate.current\_session\_context\_class=org.springframework.orm.hibernate5.SpringSessionContext  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect  
spring.jpa.properties.hibernate.id.new\_generator\_mappings=false  
spring.jpa.properties.hibernate.format\_sql=true  
spring.jackson.serialization.fail-on-empty-beans=false  
# ===============================  
# = Thymeleaf configurations  
# ===============================  
spring.thymeleaf.mode=HTML  
spring.thymeleaf.cache=false  
# ===============================  
# = Other configurations  
# ===============================  
spring.servlet.multipart.max-file-size=10mb  
spring.servlet.multipart.max-request-size=10mb  
logging.level.org.hibernate.SQL=DEBUG  
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org"  
>  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
  
<div th:replace="'fragments/indexContent'"></div>  
  
<br><br>  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<html xmlns:th="http://www.thymeleaf.org" xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
  
<div class="container">  
 <div class="row">  
 <div id="postlist">  
 <div class="panel">  
 <div class="panel-heading">  
 <div class="text-center">  
 <div class="row">  
 <div class="col-sm-9">  
 <h3 class="pull-left"><span th:utext="${news.getTitle()}"></span>  
 </h3>  
 </div>  
 <div class="col-sm-3">  
 <h4 class="pull-right">  
 <small><em th:text="${#dates.format(news.getCreateDate(), 'dd MMMM yyyy HH:mm')}">2014-07-30<br>18:30:00</em>  
 </small>  
 <th:block sec:authorize="hasRole('ROLE\_MANAGER')">  
 <br/>  
 <small><a th:href="@{|/news/remove/${news.getId()}|}"><em class="alert-danger">удалить  
 запись</em></a></small>  
 </th:block>  
  
 </h4>  
 </div>  
 </div>  
 </div>  
 </div>  
  
 <div class="panel-body" th:utext="${news.getText()}">  
 </div>  
  
 </div>  
 </div>  
 </div>  
</div>  
<br><br>  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
  
<div class="container">  
 <h3 class="page-title">Страница заказов пользователя</h3>  
 <div class="table-responsive">  
 <table class="table">  
 <caption>Кол-во заказов: <span th:utext="${orders.size()}"></span></caption>  
 <thead>  
 <tr>  
 <th scope="col">Номер</th>  
 <th scope="col">Дата</th>  
 <th scope="col">ФИО</th>  
 <th scope="col">Адрес</th>  
 <th scope="col">Email</th>  
 <th scope="col">Сумма</th>  
 <th scope="col">Просмотр</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="order : ${orders}">  
 <td scope="row" th:utext="${order.id}"></td>  
 <td th:utext="${#dates.format(order.orderDate,'dd-MM-yyyy HH:mm')}"></td>  
 <td th:utext="${order.getCustomerName()}"></td>  
 <td th:utext="${order.customerAddress}"></td>  
 <td th:utext="${order.customerEmail}"></td>  
 <td style="color:red;" th:utext="${#numbers.formatDecimal(order.amount,3,2,'COMMA')}"></td>  
 <td><a th:href="@{|/user/order?orderId=${order.id}|}">Просмотр</a></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
 <!-- <div class="page-navigator" th:if="${paginationResult.totalPages > 1}">  
 <th:block th:each="page: ${paginationResult.navigationPages}">  
 <a th:if="${page != -1}" class="nav-item"  
 th:href="@{|/admin/orderList?page=${page}|}" th:utext="${page}"></a>  
  
 <span th:if="${page == -1}" class="nav-item"> ... </span>  
 </th:block>  
 </div>-->  
  
</div>  
<br><br>  
  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
  
<div class="container">  
 <h2>Профиль пользователя</h2>  
 <div class="user-container">  
 <h3>Имя пользователя: <span th:utext="${#request.userPrincipal.name}"></span></h3>  
 <h4>  
 <div th:if="${user.userState == T(org.solteh.model.UserState).ROLE\_EMPLOYEE}">  
 Статус:<span class="text-success">Пользователь</span></div>  
 <div th:if="${user.userState == T(org.solteh.model.UserState).ROLE\_MANAGER}">  
 Статус:<span class="text-danger">Администратор</span>  
 </div>  
 </h4>  
 </div>  
 <div th:if="${#request.getParameter('error') == 'true'}" style="color: red; margin: 10px 0px;">  
 Заполните данные пользователя.  
 </div>  
 <h4>Персональная информация:</h4>  
 <form method="post" th:action="@{/profile}" th:object="${user}">  
 <ul class="list-group">  
 <li class="list-group-item"><input type="text"  
 th:field="${user.fio}"  
 placeholder="ФИО"  
 class="form-control"  
 required/></li>  
 <li class="list-group-item"><input type="text"  
 th:field="${user.email}"  
 placeholder="Адрес электронной почты"  
 class="form-control"  
 required/></li>  
 <li class="list-group-item"><input type="text"  
 th:field="${user.address}"  
 placeholder="Адрес доставки"  
 class="form-control"  
 required/></li>  
 <li class="list-group-item"><input type="text"  
 th:field="${user.phone}"  
 placeholder="Контактный телефон"  
 class="form-control"  
 required/></li>  
 </ul>  
 <button type="submit" class="btn btn-primary block full-width m-b">Сохранить</button>  
 </form>  
</div>  
<br><br>  
  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
  
<div class="container">  
 <h2>Профиль пользователя</h2>  
 <div class="user-container">  
 <h3>Имя пользователя: <span th:utext="${#request.userPrincipal.name}"></span></h3>  
 <h4>  
 <div th:if="${user.userState == T(org.solteh.model.UserState).ROLE\_EMPLOYEE}">  
 Статус:<span class="text-success">Пользователь</span></div>  
 <div th:if="${user.userState == T(org.solteh.model.UserState).ROLE\_MANAGER}">  
 Статус:<span class="text-danger">Администратор</span>  
 </div>  
 </h4>  
 </div>  
 <div th:if="${#request.getParameter('error') == 'true'}" style="color: red; margin: 10px 0px;">  
 Заполните данные пользователя.  
 </div>  
 <h4>Персональная информация:</h4>  
 <form method="post" th:action="@{/profile}" th:object="${user}">  
 <ul class="list-group">  
 <li class="list-group-item"><input type="text"  
 th:field="${user.fio}"  
 placeholder="ФИО"  
 class="form-control"  
 required/></li>  
 <li class="list-group-item"><input type="text"  
 th:field="${user.email}"  
 placeholder="Адрес электронной почты"  
 class="form-control"  
 required/></li>  
 <li class="list-group-item"><input type="text"  
 th:field="${user.address}"  
 placeholder="Адрес доставки"  
 class="form-control"  
 required/></li>  
 <li class="list-group-item"><input type="text"  
 th:field="${user.phone}"  
 placeholder="Контактный телефон"  
 class="form-control"  
 required/></li>  
 </ul>  
 <button type="submit" class="btn btn-primary block full-width m-b">Сохранить</button>  
 </form>  
</div>  
<br><br>  
  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<nav xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4"  
 th:fragment="nav"  
 class="navbar navbar-inverse">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 </div>  
 <div class="collapse navbar-collapse" id="myNavbar">  
 <ul class="nav navbar-nav">  
 <li th:class="${content}=='containerExcl'? active : ''"><a href="/">Главная</a></li>  
 <li th:class="${content}=='products'? active : ''"><a th:href="@{/products}">Товары</a></li>  
 <li th:class="${content}=='topsales'? active : ''"><a th:href="@{/topsales}">Лучшее предложение</a></li>  
 <li th:class="${content}=='contacts'? active : ''"><a th:href="@{/contacts}">Контакты</a></li>  
 </ul>  
  
 <ul class="nav navbar-nav navbar-right">  
 <th:block sec:authorize="!isAuthenticated()">  
 <li>  
 <a th:href="@{/login}">Авторизация</a>  
 </li>  
 <li>  
 <a th:href="@{/registration}">Регистрация</a>  
 </li>  
 </th:block>  
 <th:block sec:authorize="isAuthenticated()">  
  
 <li class="dropdown">  
 <a href="#"  
 class="dropdown-toggle"  
 data-toggle="dropdown"  
 role="button"  
 aria-haspopup="true"  
 aria-expanded="false">  
 <span class="glyphicon glyphicon-user"></span> Привет,  
 <span th:utext="${#request.userPrincipal.name}"></span>  
 </a>  
 <ul class="dropdown-menu">  
 <th:block sec:authorize="hasAnyRole('ROLE\_MANAGER','ROLE\_EMPLOYEE')">  
 <li><a th:href="@{/profile}">Профиль</a></li>  
 <li><a th:href="@{/user/orders}">Список заказов</a></li>  
 </th:block>  
 <th:block sec:authorize="hasRole('ROLE\_MANAGER')">  
 <li role="separator" class="divider"></li>  
 <li>  
 <a th:href="@{/admin/product}">Добавить продукт</a>  
 <a th:href="@{/admin/news}">Добавить новость</a>  
 <a th:href="@{/admin/report}">Статистика заказов</a>  
 <a th:href="@{/admin/users}">Список пользователей</a>  
 </li>  
 </th:block>  
 <li role="separator" class="divider"></li>  
 <li><a th:href="@{/logout}">Выйти</a></li>  
 </ul>  
 </li>  
 </th:block>  
 <li><a th:href="@{/shoppingCart}"><span class="glyphicon glyphicon-shopping-cart"></span> Корзина</a>  
 </li>  
 </ul>  
 </div>  
 </div>  
</nav>

<div th:fragment="contacts" class="container">  
 <div class="row">  
 <div class="col col-lg-6">  
 <iframe src="https://yandex.by/map-widget/v1/-/CBetNSxkoB" width="560" height="400"  
 frameborder="0"></iframe>  
 </div>  
 <div class="col">  
 <h1>О компании</h1><br/>  
 Своим рождением первые системы и методы электронной коммерции обязаны появлению технологий автоматизации  
 продаж и внедрению автоматизированных систем управления корпоративными ресурсами[1]. В 1960 году  
 американские компании American Airlines и IBM приступают к созданию системы автоматизации процедуры  
 резервирования мест на авиарейсы. Система SABRE (Semi-Automatic Business Research Environment) сделала  
 воздушные перелёты более доступными для рядовых пассажиров, помогая им ориентироваться в тарифах и  
 рейсах, число которых постоянно растёт. За счёт автоматизации процесса расчёта тарифов при  
 резервировании мест снизилась стоимость услуг, выросли пассажироперевозки[3]. Совместный проект American  
 Airlines и IBM — один из первых примеров электронной коммерции.  
  
 Наиболее динамично рынок электронной коммерции развивается в течение последних 20 лет, что обусловлено  
 стремительным ростом количества интернет-пользователей, увеличением влияния социальных сетей и других  
 интерактивных онлайн-платформ, динамичным развитием систем электронных платежей и переходом ведущих  
 игроков рынка к новым технологическим платформам для электронной коммерции[4] (от Web 1.0 к Web 2.0,  
 далее к Web 3.0).  
  
 По результатам исследования, суммарный оборот интернет-коммерции в 2012 году составил 350,6 млрд рублей,  
 а количество интернет-магазинов за 2012 год (по отношению к 2011) увеличилось на 30 % и составило 32,5  
 тыс.[5]. В 2011 году суммарный оборот интернет-магазинов оценивался в 258 млрд рублей. Рост объёма рынка  
 увеличился 36 %[5]. По результатам исследования РАЭК за 2012 год, объём рынка онлайн-торговли составил  
 284,96 млрд руб[6].  
 </div>  
 </div>  
</div>

<footer xmlns:th="http://www.thymeleaf.org" th:fragment="footer" class="container-fluid text-center"  
 style="margin-bottom: 50px">  
 <p>Интернет-приложение ведения электронной коммерции</p>  
 <form class="form-inline" th:method="post" th:action="@{/subscribe}">Получать уведомления  
 <input type="email" id="email" name="email" class="form-control" size="50" placeholder="Email адрес">  
 <button type="submit" class="btn btn-danger">Подписаться</button>  
 </form>  
</footer>

<head th:fragment="header">  
 <title>Интернет-приложение ведения электронной коммерции</title>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" type="text/css"  
 href="/webjars/bootstrap/3.3.7-1/css/bootstrap.min.css"/>  
 <link rel="stylesheet" type="text/css" href="/static/css/main.css">  
 <link rel="stylesheet" type="text/css" href="/static/css/j-rating.css">  
 <script type="text/javascript" src="/webjars/jquery/1.11.1/jquery.min.js"></script>  
 <script type="text/javascript" src="/webjars/bootstrap/3.3.7-1/js/bootstrap.min.js"></script>  
</head>

<div xmlns:th="http://www.thymeleaf.org" th:fragment="containerExcl" class="container">  
 <div>  
 <div class="row">  
 <div class="col-sm-3">  
 <div class="list-group">  
 <h3 class="nav-header disabled"><a>Новости: </a></h3>  
  
 <th:block th:each="item: ${news}">  
 <a th:href="@{|/news/${item.getId()}|}"  
 class="list-group-item list-group-item-action flex-column align-items-start">  
 <div class="d-flex w-100 justify-content-between">  
 <h5 class="mb-1" th:utext="${item.getTitle()}">List group item heading</h5>  
 <small th:text="${#dates.format(item.getCreateDate(), 'dd-MMM-yyyy')}">3 days ago  
 </small>  
 </div>  
 </a>  
 </th:block>  
 </div>  
 </div>  
 <div class="col-sm-9">  
 <h3 class="col-sm-12">Самые покупаемые товары:</h3>  
 <th:block th:each="product: ${products}">  
 <div class="col-sm-4">  
 <div class="panel panel-success">  
 <div class="panel-heading" th:text="${product.getName()}">BLACK FRIDAY DEAL</div>  
 <div class="panel-body"><img th:src="@{|/productImage?code=${product.getCode()}|}"  
 class="img-responsive">  
 <div class="caption">  
 <h5 th:utext="${product.getName()}"></h5>  
 <p th:utext="${product.getDescription()}"></p>  
 </div>  
 <div class="ratings">  
 <h5 class="pull-right"  
 th:utext="${#numbers.formatDecimal(product.getPrice(),1,2,'COMMA')+' руб.'} "></h5>  
 <p>  
 </p>  
 </div>  
  
  
 <th:block sec:authorize="hasAuthority('ROLE\_MANAGER')">  
  
 <a style="color:red;"  
 th:href="@{|/admin/product?code=${product.getCode()}|}">Edit Product</a>  
  
 </th:block>  
 </div>  
 <div class="panel-footer">  
 <span th:text="${product.getName()}"></span>  
 <a href="#" th:href="@{|/buyProduct?code=${product.getCode()}|}">  
 <span class="glyphicon glyphicon-shopping-cart pull-right"></span>  
 </a></div>  
 </div>  
 </div>  
 </th:block>  
 </div>  
 </div>  
 </div>  
 <br>  
</div>

<div th:fragment="webstore" class="jumbotron">  
 <div class="container text-center">  
 <h1>Интернет-приложение</h1>  
 <p>ведения электронной коммерции</p>  
 </div>  
</div>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4">  
<head th:replace="fragments/head :: head"></head>  
<body>  
<script type="text/javascript" src="/static/js/markitup/jquery.markitup.js"></script>  
<script type="text/javascript" src="/static/js/markitup/sets/default/set.js"></script>  
<link rel="stylesheet" type="text/css" href="/static/js/markitup/skins/markitup/style.css"/>  
<link rel="stylesheet" type="text/css" href="/static/js/markitup/sets/default/style.css"/>  
<script type="text/javascript">  
 jQuery(document).ready(function () {  
 jQuery("#markItUp").markItUp(mySettings);  
 });  
</script>  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
<th:block sec:authorize="hasRole('ROLE\_MANAGER')">  
  
<div class="container">  
 <form th:object="${news}" th:action="@{/admin/news}" th:method="POST">  
 <div class="form-group">  
 <label for="title" >  
 Введите заголовок </label>  
 <input id="title" class="form-control" type="text" th:field="\*{title}"/>  
 <label for="markItUp" >Введите тект новости</label>  
 <textarea id="markItUp" class="form-control" rows="10" th:field="\*{text}"></textarea>  
 <button type="submit" class="form-control">Сохарнить</button>  
 </div>  
 </form>  
  
</div>  
</th:block>  
<br><br>  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
<div th:fragment="products" class="container center-block">  
 <div class="row ">  
 <div class="col-md-4">  
 <h3>Авторизация</h3>  
 <h4>Введите имя пользователя и пароль</h4>  
 <!-- /login?error=true -->  
 <th:block th:if="${#session != null && #session.getAttribute('SPRING\_SECURITY\_LAST\_EXCEPTION') != null}">  
 <div th:if="${#request.getParameter('error') == 'true'}" style="color: red; margin: 10px 0px;">  
 Ошибка авторизации!!!<br/> Причина :  
 <span class="error-message" th:utext="${error}"></span>  
 </div>  
 </th:block>  
  
 <form method="POST" th:action="@{/j\_spring\_security\_check}">  
 <table>  
 <tr>  
 <div class="form-group input-group has-feedback">  
 <span class="input-group-addon"><span class="glyphicon glyphicon-user"></span></span>  
 <input type="text"  
 name="userName"  
 placeholder="Имя пользователя"  
 class="form-control"  
 required/>  
 <span class="glyphicon form-control-feedback" aria-hidden="true"></span>  
 </div>  
 </tr>  
 <tr>  
 <div class="form-group input-group">  
 <span class="input-group-addon">  
 <span class="glyphicon glyphicon-lock"></span>  
 </span>  
 <input name="password"  
 type="password"  
 id="password"  
 placeholder="Пароль"  
 class="form-control"  
 required/>  
 <span class="glyphicon" aria-hidden="true"></span>  
 </div>  
 </tr>  
 <tr>  
 <td>  
 <input type="submit" class="btn btn-info block full-width m-b" value="Авторизация"/>  
 <input type="reset" class="btn btn-info block full-width m-b" value="Сброс"/>  
 </td>  
 </tr>  
 </table>  
 </form>  
 </div>  
 </div>  
 <br><br>  
</div>  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
  
<div class="container">  
 <h1>Информация о заказе</h1>  
  
 <div class="customer-info-container">  
 <h3>Заказчик:</h3>  
 <ul>  
 <li>Имя: <span th:utext="${order.getUser().getFio()}"></span></li>  
 <li>Email: <span th:utext="${order.getUser().getEmail()}"></span></li>  
 <li>Телефон: <span th:utext="${order.getUser().getPhone()}"></span></li>  
 <li>Адрес: <span th:utext="${order.getUser().getAddress()}"></span></li>  
 </ul>  
  
 </div>  
 <br/>  
 <div class="row">  
 <div class="col-sm-12 col-md-10 col-md-offset-1">  
 <table class="table table-hover">  
 <thead>  
 <tr>  
 <th>Продукт</th>  
 <th>Количество</th>  
 <th class="text-center">Цена</th>  
 <th class="text-center">Сумма</th>  
 </tr>  
 </thead>  
 <tbody>  
 <span th:each="orderDetailInfo : ${order.getDetails().getProducts()}">  
 <tr>  
 <td class="col-sm-8 col-md-6">  
 <div class="media">  
 <a class="thumbnail pull-left" href="#">  
 <img class="media-object"  
 th:src="@{|/productImage?code=${orderDetailInfo.getKey().getCode()}|}"  
 style="width: 72px; height: 72px;"> </a>  
 <div class="media-body" style="padding-left: 5%;">  
 <h4 class="media-heading"  
 th:utext="${orderDetailInfo.getKey().getName()}"></h4>  
 </div>  
 </div>  
 </td>  
 <td class="col-sm-1 col-md-1" style="text-align: center">  
 <span th:utext="${orderDetailInfo.getValue()}"></span>  
 </td>  
 <td class="col-sm-1 col-md-1 text-center"><strong th:utext="${#numbers.formatDecimal(orderDetailInfo.getKey().getPrice(),3,2,'COMMA')}">$1.11</strong>  
 </td>  
 <td class="col-sm-1 col-md-1 text-center"><strong th:utext="${#numbers.formatDecimal((orderDetailInfo.getKey().getPrice() \* orderDetailInfo.getValue()),3,2,'COMMA')}">$11.11</strong>  
 </td>  
 </tr>  
 </span>  
 <tr>  
 <td></td>  
 <td></td>  
 <td><h3>Стоимость</h3></td>  
 <td class="text-right"><h3>  
 <strong th:utext="${#numbers.formatDecimal(order.getAmount(),3,2,'COMMA')}">$31.53</strong>  
 </h3></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
 </div>  
</div>  
  
<br><br>  
  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
<th:block sec:authorize="hasRole('ROLE\_MANAGER')">  
  
 <div class="container">  
 <h2>Добавление продукта</h2>  
 <div th:if="${errorMessage!= null}" class="error-message" th:utext="${errorMessage}">  
 </div>  
  
 <form th:object="${productForm}" th:action="@{/admin/product}" method="POST" enctype="multipart/form-data">  
 <table style="text-align:left;">  
 <tr>  
 <td>Код \*</td>  
 <td style="color:red;">  
 <input type="text"  
 th:field="\*{code}"/>  
 </td>  
  
 </tr>  
 <tr>  
 <td>Название продукта \*</td>  
 <td><input th:field="\*{name}"/></td>  
 </tr>  
 <tr>  
 <td>Описание продукта продукта \*</td>  
 <td><input th:field="\*{description}"/></td>  
 </tr>  
 <tr>  
 <td>Цена \*</td>  
 <td><input th:field="\*{price}"/></td>  
 </tr>  
 <tr>  
 <td>Картинка</td>  
 <td>  
 <img th:src="@{|/productImage?code=${productForm.code}|}" width="100"/>  
 </td>  
 </tr>  
 <tr>  
 <td>Загрузка картинки:</td>  
 <td><input type="file" th:field="\*{fileData}"/></td>  
 </tr>  
 <tr>  
 <td>&nbsp;</td>  
 <td>  
 <input type="submit" value="Добавить"/>  
 <input type="reset" value="Сброс"/>  
 </td>  
 </tr>  
 </table>  
 </form>  
 </div>  
 <br><br>  
</th:block>  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
<div th:fragment="products" class="container">  
 <div class="row">  
 <th:block th:each="product: ${products}">  
 <div class="col-sm-4">  
 <div class="panel panel-primary">  
 <div class="panel-heading" th:text="${product.getName()}">BLACK FRIDAY DEAL</div>  
 <div class="panel-body"><img th:src="@{|/productImage?code=${product.getCode()}|}"  
 class="img-responsive">  
 <div class="caption">  
 <h4 th:utext="${product.getName()}"></h4>  
 <p th:utext="${product.getDescription()}"></p>  
 </div>  
 <div class="ratings">  
 <h4 class="pull-right"  
 th:utext="${#numbers.formatDecimal(product.getPrice(),1,2,'COMMA')+' руб.'} "></h4>  
 <p>  
 </p>  
 </div>  
  
  
 <th:block sec:authorize="hasAuthority('ROLE\_MANAGER')">  
  
 <a style="color:red;"  
 th:href="@{|/admin/product?code=${product.getCode()}|}">Edit Product</a>  
  
 </th:block>  
 </div>  
 <div class="panel-footer">  
 <span th:text="${product.getName()}"></span>  
 <a href="#" th:href="@{|/buyProduct?code=${product.getCode()}|}">  
 <span class="glyphicon glyphicon-shopping-cart pull-right"></span>  
 </a></div>  
 </div>  
 </div>  
 </th:block>  
 </div>  
 <br><br>  
</div>  
<nav aria-label="Page navigation" class="text-center" th:if="${pageProducts.getTotalPages() > 0}">  
 <ul class="pagination">  
 <li class="page-item"><a class="page-link"  
 th:href="@{|/${parameterUrl}/${pageProducts.getPageable().getPageNumber()}|}"  
 th:if="${pageProducts.getPageable().getPageNumber()>0}">Предыдущая</a></li>  
 <th:block th:each="page:${#numbers.sequence(1,pageProducts.getTotalPages(),1)} ">  
 <li class="page-item" th:classappend="${pageProducts.getPageable().getPageNumber()+1 == page} ? active: ''">  
 <a class="page-link" th:href="@{|/${parameterUrl}/${page}|}" th:if="${page != -1}"  
 th:utext="${page}"></a></li>  
 </th:block>  
 <li class="page-item"><a class="page-link"  
 th:href="@{|/${parameterUrl}/${pageProducts.getPageable().getPageNumber()+2}|}"  
 th:if="${pageProducts.getPageable().getPageNumber()+1 != pageProducts.getTotalPages()}">Следующая</a>  
 </li>  
  
 </ul>  
</nav>  
<br><br>  
  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
<div class="container">  
 <div class="col-md-5">  
 <div>  
  
 <h3>Регистрация нового пользователя</h3>  
  
 <form autocomplete="off"  
 action="#"  
 th:action="@{/registration}"  
 th:object="${userForm}"  
 method="post"  
 class="m-t"  
 role="form"  
 data-toggle="validator">  
  
 <div th:if="${#fields.hasErrors('userName')}"  
 th:errors="\*{userName}"  
 class="validation-message alert alert-danger"  
 role="alert"></div>  
  
 <div th:if="${#fields.hasErrors('encrytedPassword')}"  
 th:errors="\*{encrytedPassword}"  
 class="validation-message alert alert-danger"  
 role="alert"></div>  
  
 <div class="form-group input-group has-feedback">  
 <label th:if="${#fields.hasErrors('userName')}"  
 th:errors="\*{userName}"  
 class="validation-message"></label>  
 <span class="input-group-addon"><span class="glyphicon glyphicon-user"></span></span>  
 <input type="text"  
 th:field="\*{userName}"  
 placeholder="Имя пользователя"  
 class="form-control"  
 required/>  
 <span class="glyphicon form-control-feedback" aria-hidden="true"></span>  
 </div>  
 <div class="form-group input-group has-feedback">  
 <label th:if="${#fields.hasErrors('email')}"  
 th:errors="\*{email}"  
 class="validation-message"></label>  
 <span class="input-group-addon"><span class="glyphicon glyphicon-envelope"></span></span>  
 <input type="email"  
 th:field="\*{email}"  
 placeholder="Адрес электронной почты"  
 class="form-control"  
 required/>  
 <span class="glyphicon form-control-feedback" aria-hidden="true"></span>  
 </div>  
  
 <div class="form-group input-group">  
 <span class="input-group-addon">  
 <span class="glyphicon glyphicon-lock"></span>  
 </span>  
 <input name="password"  
 type="password"  
 th:field="\*{encrytedPassword}"  
 id="password"  
 placeholder="Пароль"  
 class="form-control"  
 required/>  
 <span class="glyphicon" aria-hidden="true"></span>  
 </div>  
 <button type="submit" class="btn btn-primary block full-width m-b">Регистрация</button>  
  
 </form>  
  
 </div>  
 </div>  
</div>  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:fragment="header">  
 <title>Интернет-приложение ведения электронной коммерции</title>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/3.3.7-1/css/bootstrap.min.css"/>  
 <link rel="stylesheet" type="text/css" href="/static/css/main.css">  
 <script type="text/javascript" src="/webjars/jquery/1.11.1/jquery.min.js"></script>  
 <script type="text/javascript" src="/webjars/bootstrap/3.3.7-1/js/bootstrap.min.js"></script>  
 <script type="text/javascript" src="/static/js/report.js"></script>  
</head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
<th:block sec:authorize="hasRole('ROLE\_MANAGER')">  
  
 <div class="container">  
 <div id="chartContainer" style="height: 300px; width: 100%;"></div>  
  
 <script src="https://canvasjs.com/assets/script/jquery.canvasjs.min.js"></script>  
 </div>  
</th:block>  
<br><br>  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
<div class="container">  
  
 <h2>Корзина</h2>  
  
 <th:block th:if="${cartForm == null || cartForm.cartLines == null || cartForm.cartLines.empty}">  
 <h2>Нету товаров в корзине</h2>  
 <a th:href="@{/products}">Продолжить покупки</a>  
 </th:block>  
 <th:block th:if="${cartForm != null && cartForm.cartLines != null && !cartForm.cartLines.empty}">  
 <form method="post" th:object="${cartForm}" th:action="@{/shoppingCart}">  
  
 <div class="row">  
 <div class="col-sm-12 col-md-10 col-md-offset-1">  
 <table class="table table-hover">  
 <thead>  
 <tr>  
 <th>Продукт</th>  
 <th>Количество</th>  
 <th class="text-center">Цена</th>  
 <th class="text-center">Сумма</th>  
 <th> </th>  
 </tr>  
 </thead>  
 <tbody>  
 <span th:each="cartLineInfo, varStatus : ${cartForm.getCartLines()}">  
 <tr>  
 <td class="col-sm-8 col-md-6">  
 <div class="media">  
 <a class="thumbnail pull-left" href="#">  
 <img class="media-object"  
 th:src="@{|/productImage?code=${cartLineInfo.product.code}|}"  
 style="width: 72px; height: 72px;"> </a>  
 <div class="media-body" style="padding-left: 5%;">  
 <h4 class="media-heading"  
 th:utext="${cartLineInfo.product.name}"></h4>  
 <input type="hidden"  
 th:name="|cartLines[${varStatus.index}].product.code|"  
 th:value="${cartLineInfo.product.code}"/>  
 </div>  
 </div>  
 </td>  
 <td class="col-sm-1 col-md-1" style="text-align: center">  
 <input class="form-control"  
 th:name="|cartLines[${varStatus.index}].quantity|"  
 th:value="${cartLineInfo.quantity}"/>  
 </td>  
 <td class="col-sm-1 col-md-1 text-center"><strong th:utext="${#numbers.formatDecimal(cartLineInfo.product.price,3,2,'COMMA')}">$1.11</strong>  
 </td>  
 <td class="col-sm-1 col-md-1 text-center"><strong th:utext="${#numbers.formatDecimal(cartLineInfo.getAmount(),3,2,'COMMA')}">$11.11</strong>  
 </td>  
 <td class="col-sm-1 col-md-1">  
 <a class="btn btn-danger"  
 th:href="@{|/shoppingCartRemoveProduct?code=${cartLineInfo.getProduct().getCode()}|}">  
 <span class="glyphicon glyphicon-remove"></span>Удалить  
 </a>  
 </td>  
 </tr>  
 </span>  
 <tr>  
 <td>  </td>  
 <td>  </td>  
 <td>  </td>  
 <td><h3>Стоимость</h3></td>  
 <td class="text-right"><h3>  
 <strong th:utext="${#numbers.formatDecimal(cartForm.getAmountTotal(),3,2,'COMMA')}">$31.53</strong>  
 </h3></td>  
 </tr>  
 <tr>  
 <td>  </td>  
 <td>  </td>  
 <td>  
 <button type="submit" class="btn btn-info">  
 <span class="glyphicon glyphicon-refresh"></span> Обновить количество  
 </button>  
 </td>  
 <td>  
 <a class="btn btn-default" th:href="@{/products}">  
 <span class="glyphicon glyphicon-shopping-cart"></span> Продолжить покупки  
 </a>  
 </td>  
 <td>  
 <a class="btn btn-success" th:href="@{/shoppingCartConfirmation}">  
 Оформить заказ <span class="glyphicon glyphicon-play"></span>  
 </a>  
 </td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
 </div>  
 </form>  
 </th:block>  
 <br><br>  
</div>  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
<div class="container">  
 <div class="page-title">Подтверждение заказа</div>  
 <div class="customer-info-container">  
 <h3>Информация о заказе:</h3>  
 <ul><!--/\*@thymesVar id="cartInfo" type="org.solteh.web.controller.ShoppingCartController"\*/-->  
 <li>Имя: <span th:utext="${cartInfo.name}"></span></li>  
 <li>Email: <span th:utext="${cartInfo.email}"></span></li>  
 <li>Телефон: <span th:utext="${cartInfo.phone}"></span></li>  
 <li>Адрес: <span th:utext="${cartInfo.address}"></span></li>  
 </ul>  
 <h3>Обзор корзины:</h3>  
 <ul>  
 <li>Кол-во:  
 <span th:utext="${cartInfo.getQuantityTotal()}"></span></li>  
 <li>Сумма:  
 <span class="total" th:utext="${#numbers.formatDecimal(cartInfo.getAmountTotal(),3,2,'COMMA')}">  
 </span>  
 </li>  
 </ul>  
 </div>  
 <form method="POST" th:action="@{/shoppingCartConfirmation}">  
 <!-- Edit Cart -->  
 <a class="btn btn-primary block full-width m-b" th:href="@{/shoppingCart}">  
 Изменить заказ  
 </a>  
 <!-- Edit Customer Info -->  
 <a class="btn btn-primary block full-width m-b" th:href="@{/profile}">  
 Изменить информацию заказчика  
 </a>  
 <!-- Send/Save -->  
 <input type="submit" value="Сохранить" class="btn btn-primary block full-width m-b"/>  
 </form>  
 <div class="container">  
 <div class="row">  
 <div class="col-sm-12 col-md-10 col-md-offset-1">  
 <table class="table table-hover">  
 <thead>  
 <tr>  
 <th>Продукт</th>  
 <th>Количество</th>  
 <th class="text-center">Цена</th>  
 <th class="text-center">Сумма</th>  
 <th> </th>  
 </tr>  
 </thead>  
 <tbody>  
 <span th:each="cartLineInfo, varStatus : ${cartInfo.getCartLines()}">  
 <tr>  
 <td class="col-sm-8 col-md-6">  
 <div class="media">  
 <a class="thumbnail pull-left" href="#">  
 <img class="media-object"  
 th:src="@{|/productImage?code=${cartLineInfo.product.code}|}"  
 style="width: 72px; height: 72px;"> </a>  
 <div class="media-body" style="padding-left: 5%;">  
 <h4 class="media-heading" th:utext="${cartLineInfo.product.name}"></h4>  
 </div>  
 </div>  
 </td>  
 <td class="col-sm-1 col-md-1" style="text-align: center">  
 <span th:utext="${cartLineInfo.quantity}"></span>  
 </td>  
 <td class="col-sm-1 col-md-1 text-center"><strong th:utext="${#numbers.formatDecimal(cartLineInfo.product.price,3,2,'COMMA')}">$1.11</strong>  
 </td>  
 <td class="col-sm-1 col-md-1 text-center"><strong th:utext="${#numbers.formatDecimal(cartLineInfo.getAmount(),3,2,'COMMA')}">$11.11</strong>  
 </td>  
 <td class="col-sm-1 col-md-1">  
 <a class="btn btn-danger"  
 th:href="@{|/shoppingCartRemoveProduct?code=${cartLineInfo.getProduct().getCode()}|}">  
 <span class="glyphicon glyphicon-remove"></span>Удалить  
 </a>  
 </td>  
 </tr>  
 </span>  
 <tr>  
 <td>  </td>  
 <td>  </td>  
 <td>  </td>  
 <td><h3>Стоимость</h3></td>  
 <td class="text-right"><h3>  
 <strong th:utext="${#numbers.formatDecimal(cartInfo.getAmountTotal(),3,2,'COMMA')}">$31.53</strong>  
 </h3></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
 </div>  
 </div>  
</div>  
  
<br><br>  
  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
  
<div class="container">  
 <div class="page-title">Введите данные заказа</div>  
  
 <form method="POST" th:object="${customerForm}" th:action="@{/shoppingCartCustomer}">  
  
 <table>  
 <tr>  
 <td>Имя \*</td>  
 <td><input th:field="\*{name}"/></td>  
 <td>  
 <span class="error-message"  
 th:if="${#fields.hasErrors('name')}" th:errors="\*{name}">..</span>  
 </td>  
 </tr>  
 <tr>  
 <td>Email \*</td>  
 <td><input th:field="\*{email}"/></td>  
 <td>  
 <span class="error-message"  
 th:if="${#fields.hasErrors('email')}" th:errors="\*{email}">..</span>  
 </td>  
 </tr>  
 <tr>  
 <td>Телефон \*</td>  
 <td><input th:field="\*{phone}"/></td>  
 <td>  
 <span class="error-message"  
 th:if="${#fields.hasErrors('phone')}" th:errors="\*{phone}">..</span>  
 </td>  
 </tr>  
 <tr>  
 <td>Адрес \*</td>  
 <td><input th:field="\*{address}"/></td>  
 <td>  
 <span class="error-message"  
 th:if="${#fields.hasErrors('address')}" th:errors="\*{address}">..</span>  
 </td>  
 </tr>  
 <tr>  
 <td>&nbsp;</td>  
 <td><input type="submit" value="Submit"/> <input type="reset"  
 value="Reset"/></td>  
 </tr>  
 </table>  
  
 </form>  
</div>  
<br><br>  
  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4">  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
  
<div class="container">  
 <div class="page-title">Заказ успешно оформлен!</div>  
 <div class="container">  
 <h3>Спасибо за ваш заказ!</h3>  
 Ваш номер заказа: <span th:utext="${lastOrderedCart.orderNum}"></span>  
 </div>  
</div>  
  
<br><br>  
  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4">  
<head th:replace="fragments/head :: head"></head>  
<body>  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
<div th:fragment="products" class="container">  
 <div class="row">  
 <th:block th:if="${products.size() == 0}">  
 <h4>В данной категории пока нет товаров. Сделайте первый заказ.</h4>  
 </th:block>  
 <th:block th:each="product: ${products}">  
 <div class="col-sm-4">  
 <div class="panel panel-primary">  
 <div class="panel-heading" th:text="${product.getName()}">BLACK FRIDAY DEAL</div>  
 <div class="panel-body"><img th:src="@{|/productImage?code=${product.getCode()}|}"  
 class="img-responsive">  
 <div class="caption">  
 <h4 th:utext="${product.getName()}"></h4>  
 <p th:utext="${product.getDescription()}"></p>  
 </div>  
 <div class="ratings">  
 <h4 class="pull-right"  
 th:utext="${#numbers.formatDecimal(product.getPrice(),1,2,'COMMA')+' руб.'} "></h4>  
 <p>  
 <div class="star-ratings-css">  
 <div class="star-ratings-css-top"  
 th:style="'width:' +@{${ratings.get(product.getCode())}}+'%'">  
 <span>★</span><span>★</span><span>★</span><span>★</span><span>★</span></div>  
 <div class="star-ratings-css-bottom">  
 <span>★</span><span>★</span><span>★</span><span>★</span><span>★</span></div>  
 </div>  
 </p>  
 </div>  
  
  
 <th:block sec:authorize="hasAuthority('ROLE\_MANAGER')">  
  
 <a style="color:red;"  
 th:href="@{|/admin/product?code=${product.getCode()}|}">Edit Product</a>  
  
 </th:block>  
 </div>  
 <div class="panel-footer">  
 <span th:text="${product.getName()}"></span>  
 <a href="#" th:href="@{|/buyProduct?code=${product.getCode()}|}">  
 <span class="glyphicon glyphicon-shopping-cart pull-right"></span>  
 </a></div>  
 </div>  
 </div>  
 </th:block>  
 </div>  
 <br><br>  
</div>  
<nav aria-label="Page navigation" class="text-center" th:if="${pageProducts.getTotalPages() > 0}">  
 <ul class="pagination">  
 <li class="page-item"><a class="page-link"  
 th:href="@{|/${parameterUrl}/${pageProducts.getPageable().getPageNumber()}|}"  
 th:if="${pageProducts.getPageable().getPageNumber()>0}">Предыдущая</a></li>  
 <th:block th:each="page:${#numbers.sequence(1,pageProducts.getTotalPages(),1)} ">  
 <li class="page-item" th:classappend="${pageProducts.getPageable().getPageNumber()+1 == page} ? active: ''">  
 <a class="page-link" th:href="@{|/${parameterUrl}/${page}|}" th:if="${page != -1}"  
 th:utext="${page}"></a></li>  
 </th:block>  
 <li class="page-item"><a class="page-link"  
 th:href="@{|/${parameterUrl}/${pageProducts.getPageable().getPageNumber()+2}|}"  
 th:if="${pageProducts.getPageable().getPageNumber()+1 != pageProducts.getTotalPages()}">Следующая</a>  
 </li>  
  
 </ul>  
</nav>  
<br><br>  
  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org"  
>  
<head th:replace="fragments/head :: head"></head>  
<body>  
  
<div th:replace="fragments/webstore :: webstore"></div>  
<nav th:replace="fragments/nav :: nav"></nav>  
  
  
<th:block sec:authorize="hasRole('ROLE\_MANAGER')">  
  
 <div class="container">  
 <table class="table">  
 <thead>  
 <tr>  
 <th scope="col">id</th>  
 <th scope="col">Login</th>  
 <th scope="col">FIO</th>  
 <th scope="col">email</th>  
 <th scope="col">Link</th>  
 </tr>  
 </thead>  
 <tbody th:each="user: ${users}">  
 <tr>  
 <th scope="row" th:utext="${user.getId()}">1</th>  
 <td th:utext="${user.getUserName()}">Mark</td>  
 <td th:utext="${user.getFio()}">Mark</td>  
 <td th:utext="${user.getEmail()}">Otto</td>  
 <td><a th:href="@{|/userOrder/${user.getId()}|}">Просмотреть</a></td>  
 </tr>  
 </tbody>  
 </table>  
  
 </div>  
</th:block>  
<br><br>  
<footer th:replace="fragments/footer :: footer"></footer>  
</body>  
</html>

@import url(https://fonts.googleapis.com/css?family=Open+Sans:400,600,700);  
  
.star-ratings-css {  
 unicode-bidi: bidi-override;  
 color: #c5c5c5;  
 font-size: 25px;  
 height: 25px;  
 width: 100px;  
 position: relative;  
 text-shadow: 0px 1px 0 #a2a2a2;  
}  
  
.star-ratings-css-top {  
 color: #e7711b;  
 padding: 0;  
 position: absolute;  
 z-index: 1;  
 display: block;  
 top: 0;  
 left: 0;  
 overflow: hidden;  
}  
  
.star-ratings-css-bottom {  
 padding: 0;  
 display: block;  
 z-index: 0;  
}  
  
.star-ratings-sprite {  
 background: url("https://s3-us-west-2.amazonaws.com/s.cdpn.io/2605/star-rating-sprite.png") repeat-x;  
 font-size: 0;  
 height: 21px;  
 line-height: 0;  
 overflow: hidden;  
 text-indent: -999em;  
 width: 110px;  
 margin: 0 auto;  
}  
  
.star-ratings-sprite-rating {  
 background: url("https://s3-us-west-2.amazonaws.com/s.cdpn.io/2605/star-rating-sprite.png") repeat-x;  
 background-position: 0 100%;  
 float: left;  
 height: 21px;  
 display: block;  
}  
  
@import url(http://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.min.css);  
.col-item {  
 border: 1px solid #E1E1E1;  
 border-radius: 5px;  
 background: #FFF;  
}  
  
.col-item .photo img {  
 margin: 0 auto;  
 width: 100%;  
}  
  
.col-item .info {  
 padding: 10px;  
 border-radius: 0 0 5px 5px;  
 margin-top: 1px;  
}  
  
.col-item:hover .info {  
 background-color: #F5F5DC;  
}  
  
.col-item .price {  
 /\*width: 50%;\*/  
 float: left;  
 margin-top: 5px;  
}  
  
.col-item .price h5 {  
 line-height: 20px;  
 margin: 0;  
}  
  
.price-text-color {  
 color: #219FD1;  
}  
  
.col-item .info .rating {  
 color: #777;  
}  
  
.col-item .rating {  
 /\*width: 50%;\*/  
 float: left;  
 font-size: 17px;  
 text-align: right;  
 line-height: 52px;  
 margin-bottom: 10px;  
 height: 52px;  
}  
  
.col-item .separator {  
 border-top: 1px solid #E1E1E1;  
}  
  
.clear-left {  
 clear: left;  
}  
  
.col-item .separator p {  
 line-height: 20px;  
 margin-bottom: 0;  
 margin-top: 10px;  
 text-align: center;  
}  
  
.col-item .separator p i {  
 margin-right: 5px;  
}  
  
.col-item .btn-add {  
 width: 50%;  
 float: left;  
}  
  
.col-item .btn-add {  
 border-right: 1px solid #E1E1E1;  
}  
  
.col-item .btn-details {  
 width: 50%;  
 float: left;  
 padding-left: 10px;  
}  
  
.controls {  
 margin-top: 20px;  
}  
  
[data-slide="prev"] {  
 margin-right: 10px;  
}

.jumbotron {  
 margin-bottom: 0 !important;  
}  
.button-update-sc {  
 color: red;  
 margin: 5px 5px 5px 20px;  
}  
  
.button-send-sc {  
 color: red;  
 margin: 5px 5px 5px 20px;  
}  
  
.error-message {  
 font-size: 90%;  
 color: red;  
 font-style: italic;  
}  
  
.price {  
 color: blue;  
 font-weight: bold;  
}  
  
.subtotal {  
 color: red;  
 font-weight: bold;  
}  
  
.total {  
 color: red;  
 font-weight: bold;  
 font-size: 120%;  
}  
  
table td {  
 padding: 5px;  
}  
  
.product-image{  
 width: 150px;  
}  
  
.custom-hr {  
 border: 1px;  
 height: 1px;  
 background-image: -webkit-linear-gradient(left, #f0f0f0, #8c8b8b, #f0f0f0);  
 background-image: -moz-linear-gradient(left, #f0f0f0, #8c8b8b, #f0f0f0);  
 background-image: -ms-linear-gradient(left, #f0f0f0, #8c8b8b, #f0f0f0);  
 background-image: -o-linear-gradient(left, #f0f0f0, #8c8b8b, #f0f0f0);  
}