Разработка информационной системы интернет магазина

Введение

В современном обществе каждый из нас сталкивается с тем, что ему приходится что – то покупать или же наоборот продавать и оказывать услуги. Товарно-денежные отношения являются наиболее распространенной формой взаимодействия между людьми.

2020 год очень негативно сказывается на субъектах предпринимательской и государственной деятельности из – за COVID-19.

Выбранная мною тема считается актуальной на сегодняшний день т.к. миллионы людей, не выходя из дома совершают покупки в интернет магазинах. Целью работы является разработка информационной системы интернет магазина.

Для достижения цели в работе необходимо решить следующие задачи:

* провести анализ средств разработки и выбрать инструментальную систему для разработки интернет магазина,
* проанализировать предметную область и представить ее в виде структурной схемы UML продемонстрировать при этом все сущности предметной области, атрибуты, функции и связи.
* разработать базу данных средствами сервера баз данных MongoDB express,
* реализовать вэб-интерфейс интернет-магазина.

Актуальность темы обусловлена тем что в мире огромными темпами растет количество интернет магазинов и как следствие количество потенциальных покупателей интернет магазинов так же растет.

Интернет магазины существенно уменьшают издержки их владельца на содержании обычного магазина. Из – за того что интернет магазин можно посетить из любой точки земного шара, существенно расширяется рынок сбыта.

Разработка и запуск интернет магазина в целом обходится во много раз дешевле, чем открытие обычного магазина. Если владелец обычного магазина тратится на аренду, коммунальные платежи, оплату налогов на имущество, страхование. То обладатель интернет магазина может вложить данные средства в качественную проработку и раскрутку интернет магазина.

Интернет магазин отличается большей гибкостью. К примеру ассортимент магазина можно обновлять мгновенно настолько часто, насколько это необходимо. На главной странице размещать акции на товары без необходимости в размещении офлайн рекламы, а так же в изготовлении и распространении печатных материалов. Интернет магазин не имеет ограничения по месту, поэтому предлагаемый ассортимент может приближаться к бесконечности. Здесь нет никакой привязки к сезону. Тем самым владельцу отпадает потребность менять ассортимент в зависимости от сезона. Даже не сезонные товары доступны покупателям в любое время года.

Так же, расширяются покупательские возможности. Покупатель может совершить покупку через сайт или приложение в любой стране, в любом городе, 24 часа в сутки 7 дней в неделю.

В конце концов интернет магазин это безопасно. В обычных магазинах происходят вооруженные ограбления, пьяные дебоши и т.д.

Интернет магазины не ограничиваются тем ассортиментом который при доставке вы можете визуализировать, осязать и даже попробовать на вкус. Покупатель может приобрести различный медиа контент.

На данный момент большой популярностью пользуются различные онлайн площадки интернет магазинов которые предоставляют обучающий контент. К ним относятся различные обучающие курсы начиная от естественных, технических и даже гуманитарных видов наук. Так же есть курсы по изучению иностранных языков.

Набирают популярность курсы по обучению и переобучению кадров различных специальностей.

Данный контент интернет магазинов предоставляет возможность проходить обучение удаленно путем просмотра контента как на интернет площадках данных интернет магазинов, так и с возможностью отсылки выбранного медиа контента покупателю по электронной почте либо предоставления возможности скачивания покупаемого медиа ресурса.

Это дает интернет магазинам неоспариваемое преимущество перед другими магазинами. Тем самым интернет магазины, медленно но уверенно захватывают рынок становясь монополистами в товарно-денежных отношениях. Это является ключевым моментом при переходе производителей с обычной торговли на торговлю в интернет магазинах.

Находясь в поиске своего дела, многие приходят к идее создания собственного интернет-магазина. Это достаточно удобный и перспективный вид бизнеса на развивающимся рынке.

Главным плюсом интернет магазина то что все действия контролируются через интернет. Интернет дает мобильность и позволяет работать там, где в определенный момент находится владелец бизнеса, а не наоборот.

Сайт – это один из инструментов ведения бизнеса. Ключевой концепцией любого интернет магазина является пользовательский интерфейс, через который покупатель может изучить информационную систему интернет магазина.

В первую очередь с помощью информационной системы интернет магазин ознакамливает аудиторию пользователей с структурой интернет магазина.

На главной странице магазина главной задачей является познакомить пользователя с компанией.

Карточка товаров. Каждая карточка товаров должна иметь фотографию товара, его название, цену, описание и кнопку купить. Однако это еще не все. В карточке товаров должна присутствовать кнопка редактировать, которая скрыта от обычных пользователей. Данная кнопка доступна как администратору сайта. Либо, если пользователи имеют право создавать свои карточки товаров, то данная возможность редактирования должна быть доступной только владельцу который их создал.

Главной задачей карточки товара является то, что пользователь может не только почитать описание того или иного товара, но и купить его.

Подавляющее большинство пользователей так называемые «Визуалы» т.е. те кто выбирает товар по внешнему виду.

Каждый интернет магазин должен содержать в себе корректную информацию об оплате и способе доставки товара.

Корзина и оформление заказа.

В каждой карточке товара должна присутствовать кнопка добавления товара в корзину. Иногда в некоторых карточках добавляют кнопку Купить. Однако действие у данных кнопок должно быть одинаковым

(Это запихни в описание используемых технологий а не введение)

Наступила эпоха JavaScript. Скромно начав свой путь в качестве языка сценариев, выполняемых на стороне клиента, JavaScript не только стал распространенным повсеместно на стороне клиента – его использование в качестве языка программирования серверных приложений резко возросло благодаря NodeJS.

Перспективы данного стека технологий очевидны. Теперь не нужно переключаться с JavaScript на PHP, C#, Ruby или любой другой серверный язык программирования. Это позволяет разработчикам клиентской части перейти к программированию серверных приложений.

Однако помимо знаний языка необходимо немало изучить дополнительно.

NodeJS это программная платформа основанная на движке v8 (компилирующая JavaScript в машинный код) превращая JavaScript из узконаправленного языка в язык общего назначения.

Раин Даль в 2009 году, после нескольких лет экспериментов разработал очень эффективный серверный язык, который отличается по парадигме от старых подходов т.е. были однопоточные, многопоточные среды разработки. Он разработал серверный язык на событиях. JavaScript событийно ориентированый язык. По сути, он вытащил этот язык из браузера и написал пару плагинов на С++. В итоге он получил движок, который транслирует JS отдельно от браузера. При этом он добавил различные устройства ввода вывода. Добавил определенный функционал при работе с файлами, для создания различных серверов и прочего.

NodeJS – это просто движок V8 который транслирует JS. Поверх него написана оболочка на С++. Это абсолютно отдельная сущность которая способна понимать JS, причем делает это настолько эффективно, что она может работать где угодно т.е. не ограничивая себя возможностями браузерного JS.

В NodeJS используется модульный подход благодаря которому появилась возможность размещать весь JS в разных файлах. Причем делается это очень просто и нативно.

Событийно-ориентированный подход очень оптимизирует данную платформу потому что выполняются только те куски кода которые нужны. Происходит это благодаря концепту EventLoop.

JS монополист на front-end т.к. ему нет равных. Плюс заключается в том что зная один язык я могу реализовать приложение на сервере.

Так же NodeJS позволяет использовать различные сборщики Webpack, Babel которые я использую в своем приложении. А так же пакетный менеджер npm.

1. Анализ объекта

Самая популярная тема в сайтостроении на данный момент это интернет магазины. С одной стороны есть много готовых движков, но они подходят не всем. А с другой стороны можно написать интернет магазин с нуля и под конкретные нужды, но для этого необходимо обладать хорошими знаниями различных технологий.

Любой вэб-сайт состоит из двух основопологающих вещей. Это серверная часть, и соответственно клиентская. На серверной части происходят вычисления, построение логических алгоритмов и т.д. Происходит связка с БД. В принципе этим и ограничивается серверная часть. Нужно вдаться в детали.

Клиентская часть это то что отображает браузер пользователю.

Серверная часть – это back-end.

Клиентская часть – это front-end.

А в связке они дают full-stack приложение в котором данные технологии front-end и back-end работают синхронно.

Интернет магазин должен представлять собой информационную систему. Которая в свою очередь даст понять пользователю какие услуги предоставляет интернет магазин.

Требования к интернет магазину:

* Интернет магазин должен работать on-line.
* Вся информация должна быть актуальной на текущий момент времени.
* Все изменения, сделанные пользователем, а именно сформированные или измененные заказы и т.д должны немедленно отображаться в учетной системе и быть доступными для дальнейшей обработки.

Каждый зарегистрированный пользователь, в данном интернет магазине, может добавлять свои собственные курсы для продажи. Каждый пользователь, который добавляет курсы для продажи, является администратором, владельцем данных курсов. Тем самым владельцу курсов предоставляется доступ к редактированию, удалению а так же, просмотру сделанных заказов другими зарегистрированными пользователями.

Редактировать и удалять курсы может только владелец данных курсов.

Если пользователь не является владельцем курса, то он не может произвести редактирование курса другого пользователя т.к. в целях безопасности данный функционал редактирования скрыт от таких пользователей.

Однако просматривать список курсов и описание данных курсов, которые добавили множество других пользователей, могут и не зарегистрированные пользователи. Однако их права ограничены. Они не могут редактировать курсы т.к. не являются их владельцами. А так же не могут совершить покупку. Для этого необходимо пройти регистрацию и авторизацию пользователя.

Для того что бы иметь представление о будущей структуре информационной системе, достаточно лишь обратить внимание на уже существующие информационные системы.

* 1. Описание информационной системы интернет магазина.

Удачно спроектированная информационная система интернет магазина – это в высшей степени эффективный инструмент торговли т.к. он способен захватывать внимание аудитории. Прежде всего она должна заинтриговывать потенциального покупателя.

Для того что бы представить структуру будущей информационной системы можно взглянуть на уже существующие системы интернет магазинов. Рассмотрим информационную систему интернет магазина на примере интернет магазина курсов.

Первым делом при посещении любого интернет магазина пользователь попадает на главную страницу. Внешний вид и дизайн, сруктура и наполнение главной страницы влияет на общее мнение целевой аудитории. Главная страница это лицо интернет магазина от которой в первую очередь завися продажи.

Главная страница состоит из следующих блоков

* Блок навигации (обычно рассполагается сверху или сбоку страницы)
* Главный блок
* Footer (Подвал сайта)

Блок навигации.

Довольно часто одним из базовых элементов навигации является логотип сайта. Это своего рода визитная карточка интернет магазина. Однако главным элементом навигации являются ссылки. Ссылки являются своего рода путеводителями доставляющие каждая по своиму маршруту к определенной странице интернет магазина. Каждая ссылка отвечает за свой аспект перенаправления пользователя. Исходя из этого меню интернет магазина курсов может перенаправлять пользователя к примеру на следующие страницы.

* Главная
* Курсы
* Добавить курс
* Корзина
* Заказы
* Профиль пользователя
* Войти(Выйти)

Так же является плюсом если навигация интернет магазина имеет область ввода для поска в которую так же включен фильтр по поиску тех или иных товаров, в данном случае курсов.

Однако в большинстве случаев владельцы интернет магазинов ограничивают область видимости навигации для не зарегестрированных пользователей лишь показывая им главную информацию, но не позволяя производить каких либо действий.

Исходя из этого меню для не зарегестрированных пользователей может выглядеть следующим образом:

* Главная
* Курсы
* Войти

Главная страница может быть разбита на несколько блоков каждый из которых несет свой смыл.

Считается хорошей практикой размещать блок который будет рассказывать о виде деятельности предоставляемых услуг. Обычно это своего рода само реклама которая рассписывается в двух словах для того что бы потенциальный покупатель понимал какой товар представлен в данном интернет магазине.

Часто в магазинах курсов, на главной странице, размещают блок с информацией о самых популярных и часто покупаемых курсах. Блок который предлагает курсы для дальнейшего изучения.

Так же размещают блок с информацией о акциях, скидках, распродажах, новинках и т.д

Страница «Курсы»

На странице курсов выводится весь список доступных для покупки курсов. На данной странице должна находится боковая навигация которая разбивает курсы на отдельные категории. Что в свою очередь предоставляет пользователю ознакомится с ассортиментом курсов предлагаемых интернет магазином.

Каждый курс, который предлагает интернет магазин, представляет собой карточку товаров целью которой является сфокусироватьвнимание потребителя.

Карточка товаров состоит из: фото, картинки или логотипа, названия, цены и ссылок.

Фото, картинка или логотип являются главным изображением продукта. Они являются своего рода презентацией товара. Таким образом наглядно можно донести до потребителя его выгоду и пользу от покупки данного продукта. Именно по – этому правильно подобранные картинки значительно повышают конверсию.

Следом внизу под картинкой расположено название приобретаемого курса.

Далее цена в национальной валюте.

Под ценой размещаются ссылки для просмотра подробного описания курса “Открыть курс”. Ссылка “Купить” выполняет функцию добавления курса в корзину. И еще одна ссылка, которая обычно скрыта от посторонних глаз, это ссылка “Редактировать”. Данная ссылка доступна лишь владельцу, который в свою очередь является администратором созданных им карточек товаров.

Данная ссылка позволяет владельцу курса перейти на страницу редактирования курса и внести как корректирующие изменения, так и полностью поменять содержимое карточки товаров. Изменить название курса, выставить новую цену, поменять картинку карточки товаров, изменить или полностью заменить описание продаваемого курса. И с помощью кнопки “Редактировать” внести соответствующие изменения.

А если же карточка товара потеряла свою актуальность и перестала отвечать ожидаемым от нее требованиям, то с помощью кнопки “Удалить курс” произвести удаление карточки товара.

Страница «Добавить курс»

Следующей ссылкой в области навигации является ссылка «Добавить курс». Данная страница позволяет пользователям создавать собственные карточки товаров. Тем самым в последующем производить продажу своих собственных курсов.

Страница «Корзина».

В карточке товаров при клике на кнопку «Купить» происходит добавление курса и перенаправление пользователя на страницу корзины.

Покупатель не должен быть ограничен тем, что он может заказать лишь один курс, либо одну единицу курса. Каждый покупатель, при желании, должен иметь возможность заказывать как несколько видов курсов, например из разных категорий, так и несколько единиц одного и того же курса.

При этом в корзине должен присутствовать счетчик, который должен информировать пользователя как о списке покупаемых курсов, так и количестве покупаемых единиц того или иного курса.

В списке на против каждого курса должна присутствовать кнопка удалить на тот случай если потенциальный покупатель передумал покупать курс или хочет уменьшить количество покупаемых курсов. При клике на данную кнопку счетчик должен производить перерасчет количества покупаемых курсов и информировать покупателя путем вывода новых значений элементов списка в который входят курсы и их приобретаемое количество.

Так же при любом срабатывании счетчика, которое происходит при добавлении либо удалении товара из корзины, должен происходить перерасчет общей стоимости, которая в свою очередь будет информировать покупателя о новой общей стоимости.

При окончательном решении пользователя приобрести все то что он добавил в корзину, он производит клик по кнопке «Сделать заказ». После этого происходит формирование карточки заказа на странице «Заказы».

Страница «Заказы».

Карточка заказов содержит в себе:

* идентификационный номер заказа,
* дату и время заказа,
* Имя и email сделавшего заказ,
* Список курсов и их количество,
* Общую стоимость заказа.

И далее у пользователя имеется выбор либо нажать на кнопку «Оформить заказ» либо нажать кнопку «Удалить».

Для окончательного совершения покупки и списания денег с карточки покупателя используются сторонние сервисы оплаты. Так как данное приложение создавалось не в коммерческих, то данный функционал оплаты отсутствует.

Страница «Профиль».

Каждый зарегистрированный и авторизованный пользователь имеет право на создание личного профиля.

При создании профиля из базы данных подтягиваются данные указанные при регистрации пользователя а именно email и имя пользователя. Для полноты создания профиля остается лишь загрузить фотографию для аватара.

Загрузка фотографии происходит путем клика на кнопку загрузить. Происходит обращение к файловой системе где пользователь обязан выбрать фотографию для аватара.

После выбора фотографии она отображается в профиле, но для того что бы изменения вступили в силу пользователь должен подтвердить загрузку данной фотографии путем клика на кнопку Изменить.

Навигиция, Главная страница, Footer подвал сайта, страница курсов, страница добавить курс, корзина, заказы, профиль, выйти.

Footer подвал сайта обычно содержит блоки необходимые для коммуникации с реальными и потенциальными заказчиками.

* Физические адреса,
* Телефонные номера
* Форму обратной связи
* Адрес электронной почты
* Ссылки на мессенджеры и социальные сети.

Для современных магазинов – сайт это не основа, но важный и нужный инструмент.

Большинство современных предпринимателей, при решении создать интернет магазин, стремятся получить сервис, который транслирует нужную информацию на пользовательском экране, предоставляя покупателю нужные возможности, а владельцу предоставляет инструменты по контролю над сайтом и обслуживанию клиентов т.е. структурированную информационную систему интернет магазина которая позволит без каких либо технических навыков программирования эффективно управлять интернет магазином владельцу, а покупателю, без киках либо трудностей, путешествовать по страницам интернет магазина.

Любой человек решившийся создать интернет магазин встает перед выбором движка/программы/платформы – называть это можно как угодно.

Обычно подавляющее большинство начинают использовать отдельно стоящие системы управления сайтом и его оффлайновой составляющей – например 1С. А так же систему CRM (Customer Relationship Management или «управление взаимоотношением с клиентами»). В этом случае все системы должны сообщаться между собой, они должны постоянно передавать друг другу данные о клиентах, заказах, отгрузках, движениях денег, товаров

и так далее.

Однако такую систему необходимо поддерживать и для этого необходим программист. Постоянный, приходящий, или компания к которой придется периодически обращаться.

Чем больше отдельных систем, тем больше проблем взаимодействия между ними. Особенно в современных условиях, когда практически любая IT-система находится в состоянии «вечной беты», то есть когда программный продукт непрерывно совершенствуется и обновляется, но при этом не имеет состояния, в котором он, условно, абсолютно стабилен. В любой момент любого обновления любой из систем может перестать работать стыковка между ними. Эти моменты нужно постоянно контролировать, в противном случае рассинхронизация тут же остановит какие-либо процессы. Клиенту

не позвонят в нужный момент, посылку не отправят, принятые деньги не отнесутся на заказ, а отгрузка не попадёт в отчёты. Да мало ли чего может произойти.

Есть интегрированные решения когда платформа интернет магазина – это сразу и ERP(управление ресурсами предприятия) и CRM, CMS(Content Management System – система управления контентом сайта). Но подобных систем на рынке практически нет.

Единственный и возможно самый лучший вариант для успешно развивающегося среднего или крупного проекта является «самописный» движок. Создание такого проекта отнимает огромное количество времени. Однако если все хорошо продумать, то магазин отработает многие годы, по крайней мере пока движок не потеряет своей актуальности.

* 1. Построение концептуальной модели информационной системы интернет магазина.

Концептуальная модель – это модель исследуемой предметной области построенаая на основе “Унифицированного языка моделирования UML”.

С помощью данного языка отображается вся логика системы а так же ее поведение. Нотации визуального проектирования применяются довольно долго. На мой взгляд они играют основную роль для взаимопонимания. Хорошая диаграмма часто позволяет обмениваться идеями о проекте, особенно когда вы хотите избежать излишне подробного объяснения. Диаграммы также помогают понять и программную систему, и бизнес план.

Когда группа разработчиков пытается в чем-то разобраться, диаграммы помогают установлению взаимопонимания и распространению такого понимания в команде. Диаграммы, по крайней мере пока, не заменяют языки программирования, но способны оказать существенную помощь.

Рассмотрим диаграмму последовательности на примере «Интернет магазина курсов».

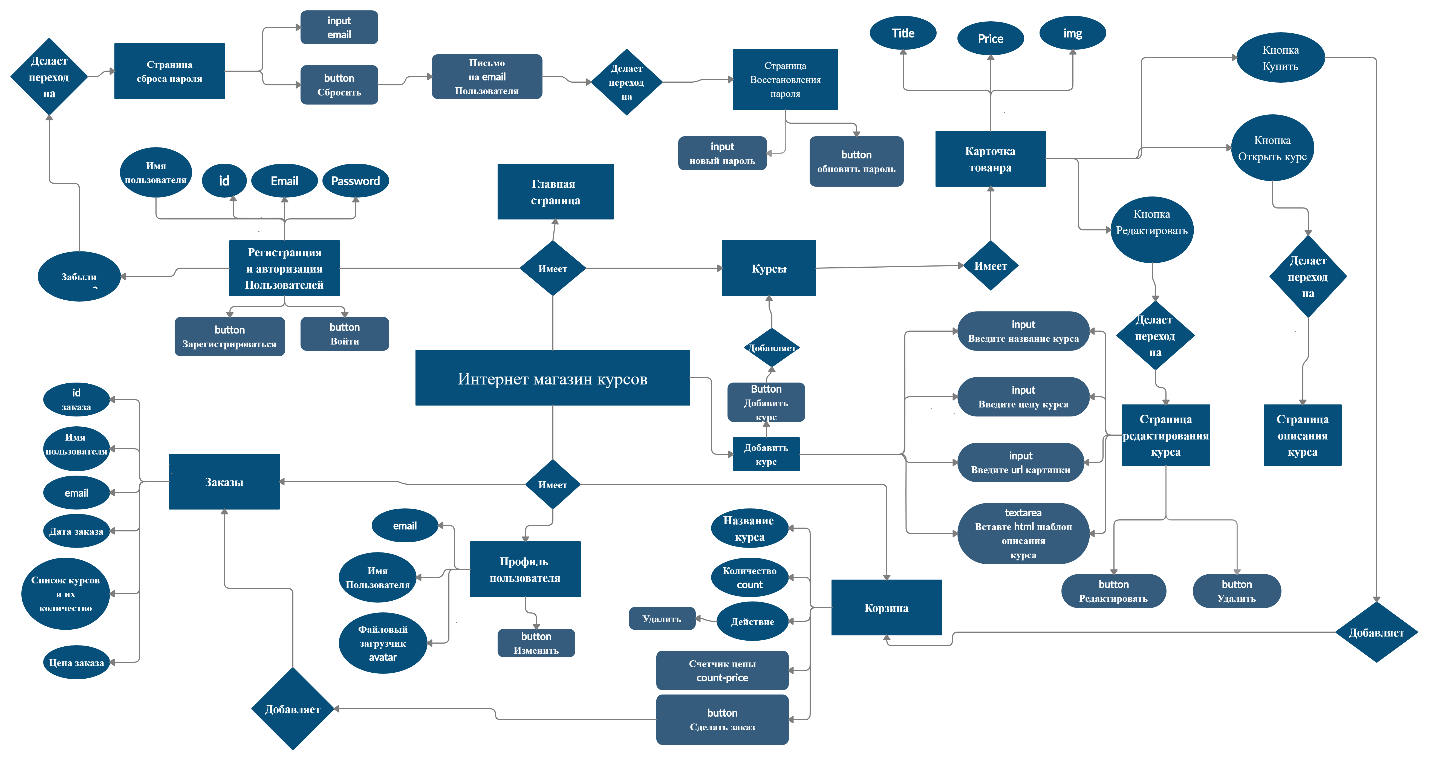


Рисунок 1.1 – Концептуальная модель интернет магазина курсов

Рассмотрим базовые компоненты которые используются для построения концептуальной модели.

* Сущность – обозначается в виде прямоугольника
* Атрибуты – обозначаются в виде овала
* Связь – обозначается в виде ромба которая объединяет все сущности

Самой главной сущностью является сам интернет магазин. У него имеется шесть главных сущностей в виде страниц интернет магазина:

* Регистрация и авторизация пользователей
* Главная страница
* Добавить курс
* Курсы
* Корзина
* Заказы
* Профиль пользователя

Рассмотрим каждую сущность в отдельности.

Регистрация и авторизация пользователей. В данной сущности имеются два главных атрибута, кнопки “Зарегистрироваться” и “Войти”.

Сущность регистрация имеет четыре атрибута. Три области ввода: имя пользователя, email пользователя, и пароль(password) пользователя.

Так же имеется автоматически генерируемый id пользователя при удачной регистрации пользователя.

Кнопка “ Войти” имеет всего два атрибута email и password.

Так же страница “Регистрации и авторизации” имеет ссылку “Забыли пароль”. Которая ведет на второстепенную сущность “Страница сброса пароля.” У нее имеются атрибуты область ввода email, и кнопка “Сбросить” при нажатии на которую отправляется письмо на email пользователю. В письме пользователь подтверждает что хочет изменить пароль нажатием на ссылку и попадает еще на одну второстепенную сущность “Страница восстановления пароля”. Данная сущность состоит из двух атрибутов: области ввода нового пароля и кнопки “Обновить пароль”.

Главная страница атрибутов не имеет т.к. она используется лишь для описания интернет магазина.

Страница “Добавить курс” состоит из следующих атрибутов: Области ввода названия курса, области ввода цены курса, области ввода url адреса картинки, и текстовой области для добавления html кода а так же кнопки добавить курс с помощью которой происходит формирование карточки товара и редирект на страницу курсов.

Страница “Курсы” имеет сущность “Карточка товара” которая в свою очередь имеет следующие атрибуты.

Title – название товара,

Price – цена товара,

Img – картинка товара

Так же имеются кнопки “Купить курс”, “Открыть курс”, “Редактировать курс”.

При нажатии на кнопку “Редактировать курс” происходит переход на второстепенную сущность “Страница редактирования курса” которая состоит из следующих атрибутов: Области ввода названия курса, области ввода цены курса, области ввода url адреса картинки, и текстовой области для добавления html кода. Так же имеются два атрибута кнопок “Редактировать” и “Удалить”.

Кнопка “Открыть курс” делает редирект на “Страница описания курса”.

Кнопка “Купить курс” делает редирект на страницу “Корзина” Где формируется список заказов.

Страница “Корзина” имеет следующие атрибуты. “Название курса”, “Количество(count)”, “Счетчик цены(count-price)”, а так же кнопки “Удалить” и “Сделать заказ”.

Страница “Заказы” имеет следующие атрибуты: id заказа, Имя пользователя, email, Дата заказа, список курсов и их количество, цена.

Страница “Профиль пользователя” имеет следующие атрибуты: email, Имя пользователя, файловый загрузчик, кнопку “Изменить”.

Если присмотреться к данной концептуальной модели, то можно обнаружить, что хоть данная модель максимально отображает архитектуру ПО, однако она не дает полную картину того что будет происходить при взаимодействии пользователя с данным программным продуктом.

1. Постановка задачи
   1. Определение требований к информационной системе интернет магазина курсов.

Пользовательский сценарий Use-case помогает описывать систему какой она должна быть с точки зрения использования. Данный подход позволяет продумать каким должен быть пользовательский интерфейс.

Use-case диаграмма представляет собой последовательность действий выполненных системой в ответ на события инициируемые действующим лицом. Она описывает взаимодействие между пользователем и системой и отражает поведение системы с точки зрения пользователя.

Действующее лицо это роль которую играет пользователь по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей и наименование работы.

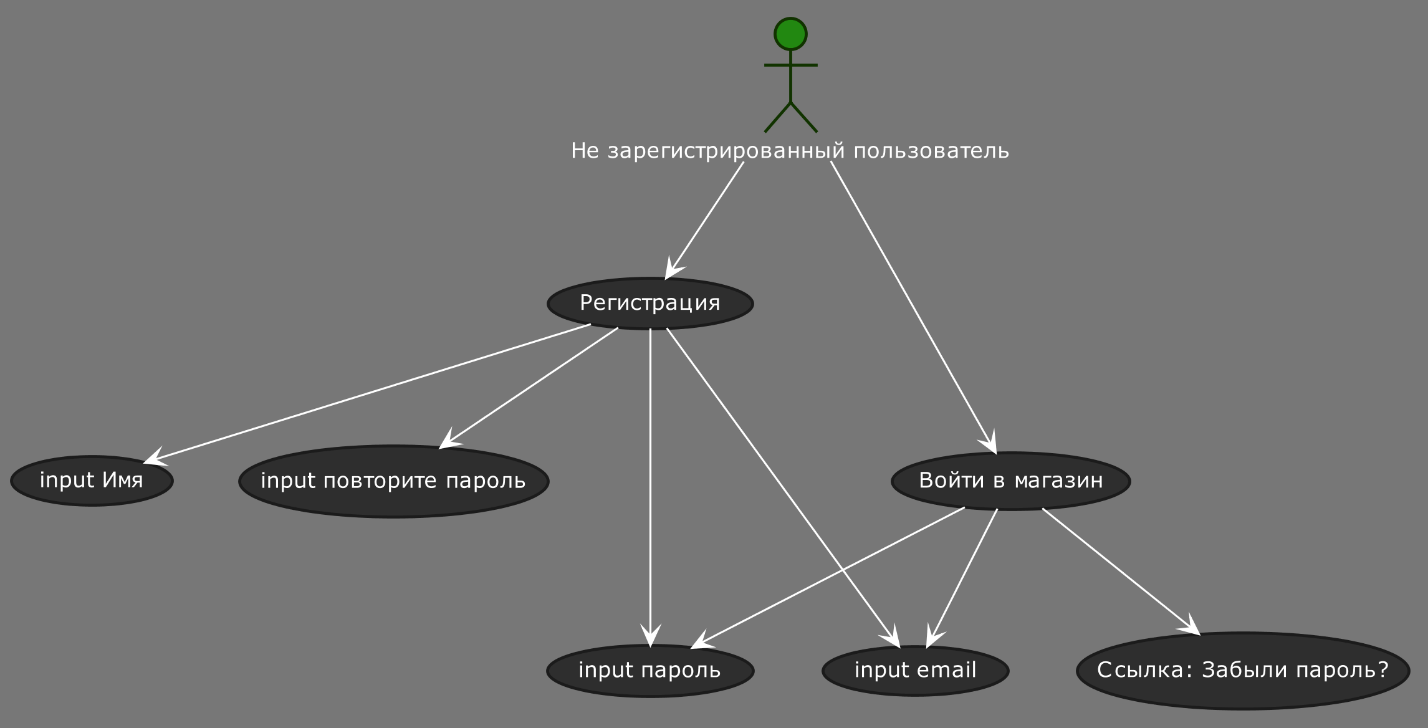
Типы действующих лиц:

- Пользователи системы

- другие системы, взаимодействующие с данными

- время.

Рассмотрим информационную систему интернет-магазина на примере Use-case диаграмм.

Рисунок 2.1 – UC-1 Регистрация и авторизация пользователя.

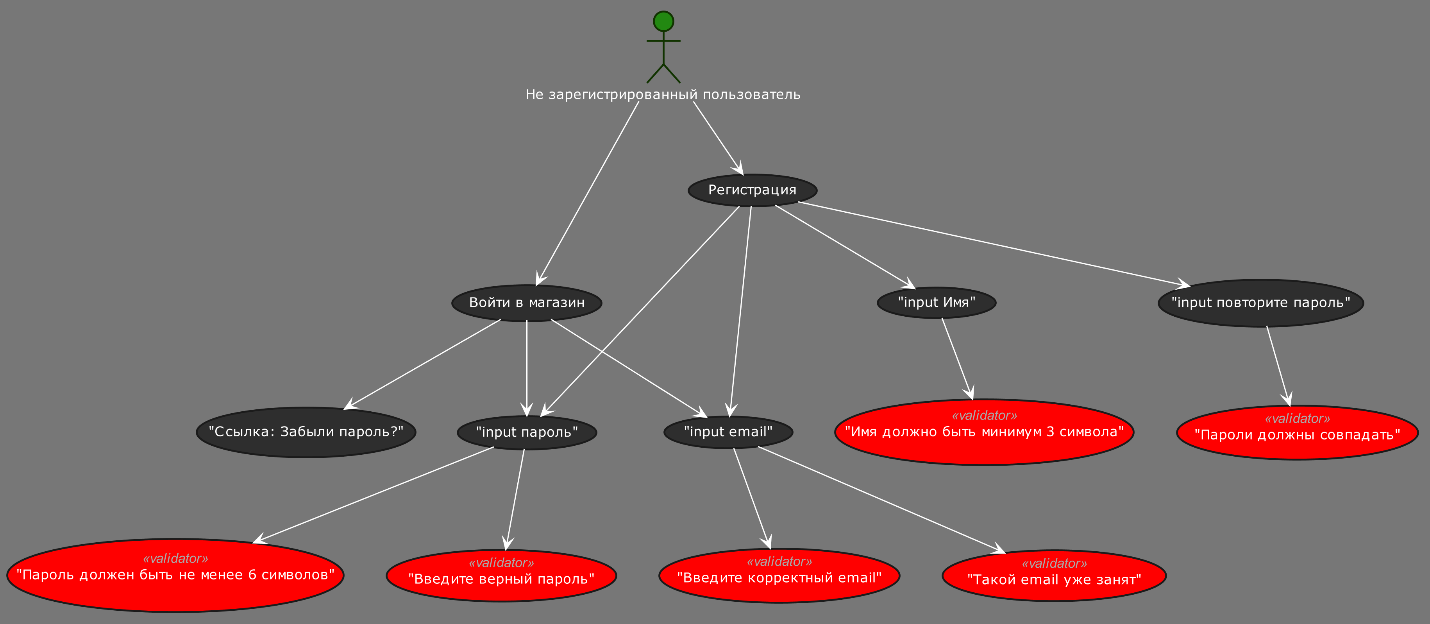


Рисунок 2.2 – UC-2 Валидация форм регистрации и авторизации пользователя.



Рисунок 2.3 – UC-3 Навигация не зарегистрированного пользователя.

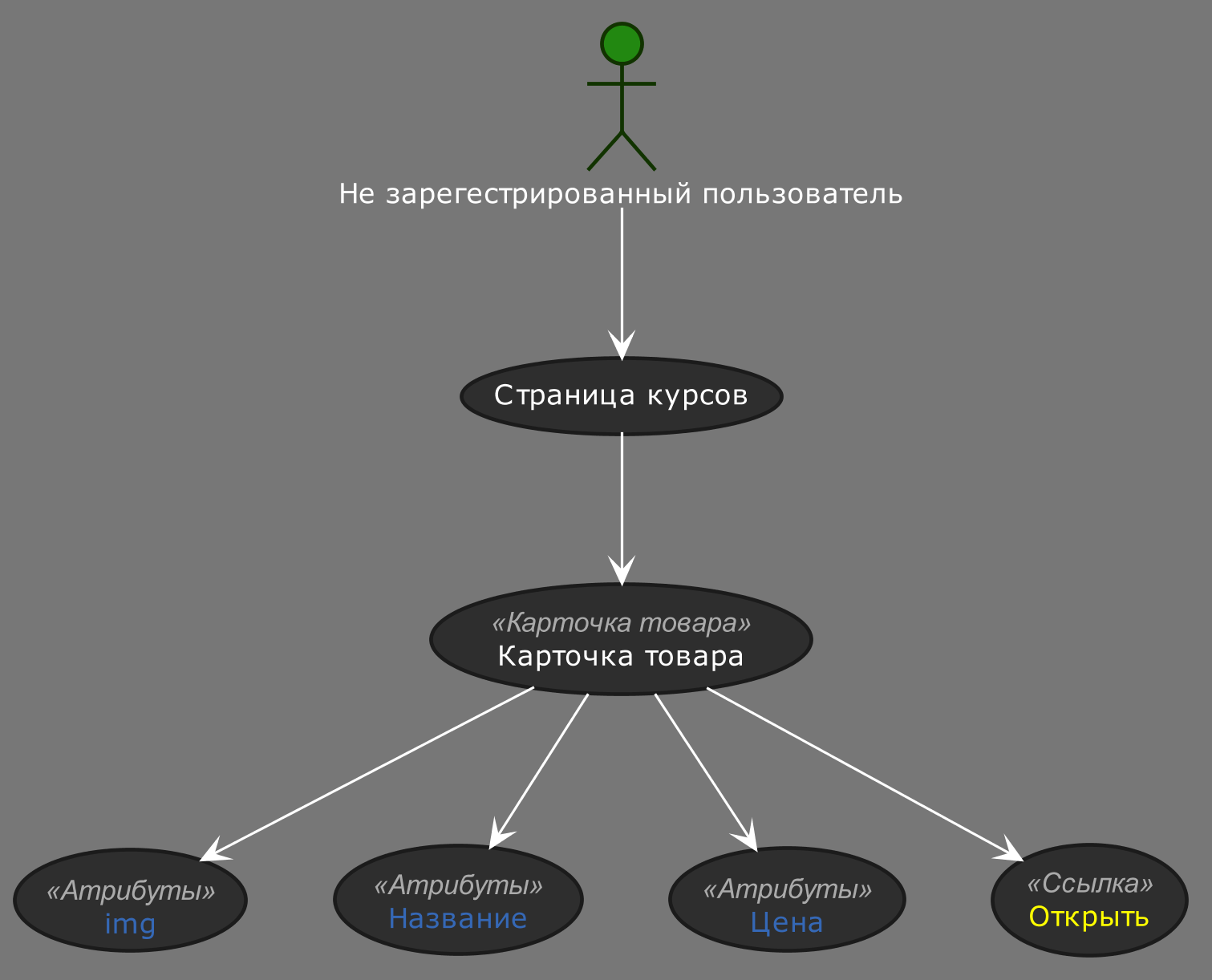


Рисунок 2.4 -UC4 Страница «Курсы» для не зарегистрированного пользователя.



Рисунок 2.4 – UC-5 Зарегистрированный пользователь «Главная».

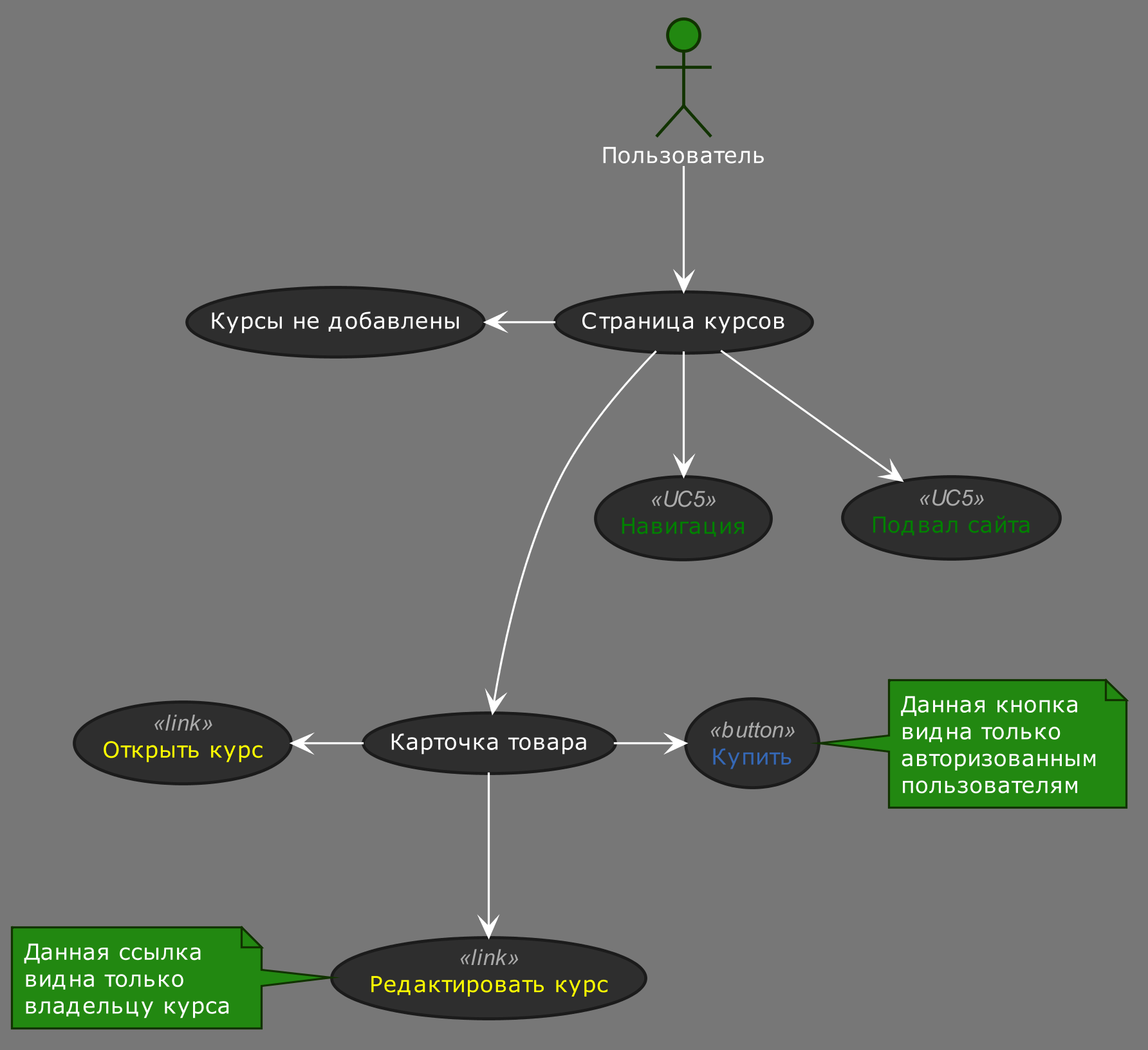


Рисунок 2.4 – UC-6 Страница «Курсы» авторизованного пользователя.

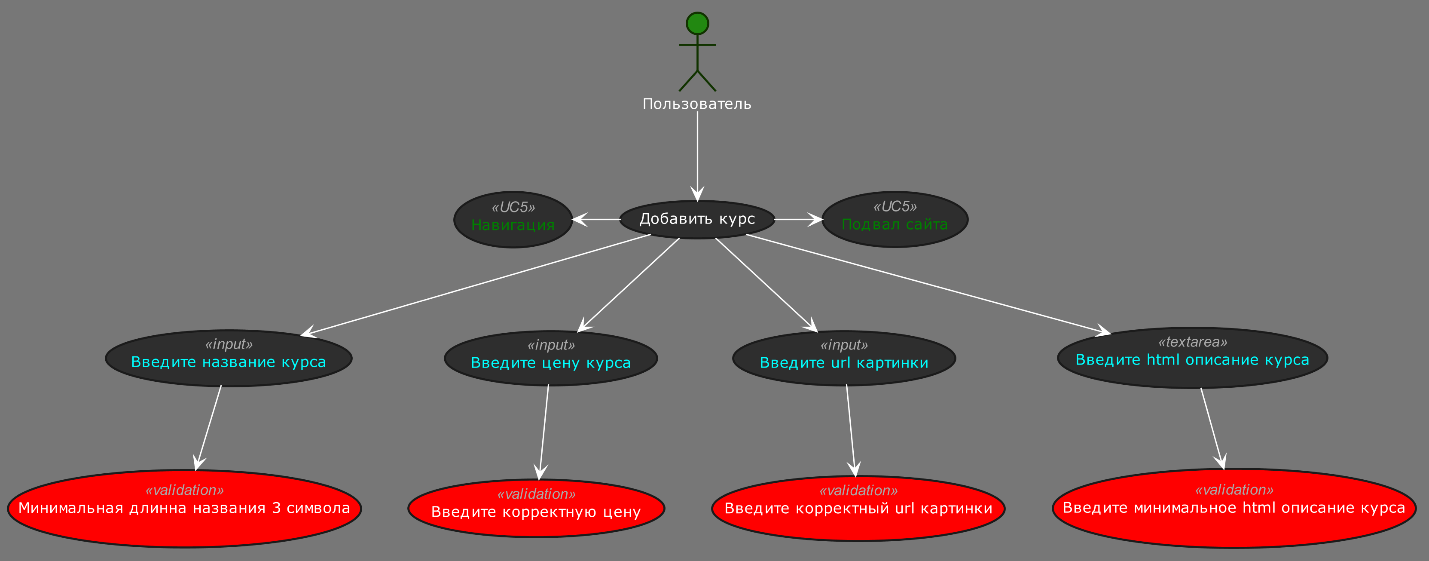


Рисунок 2.5 – UC-7 Страница «Добавить курс».

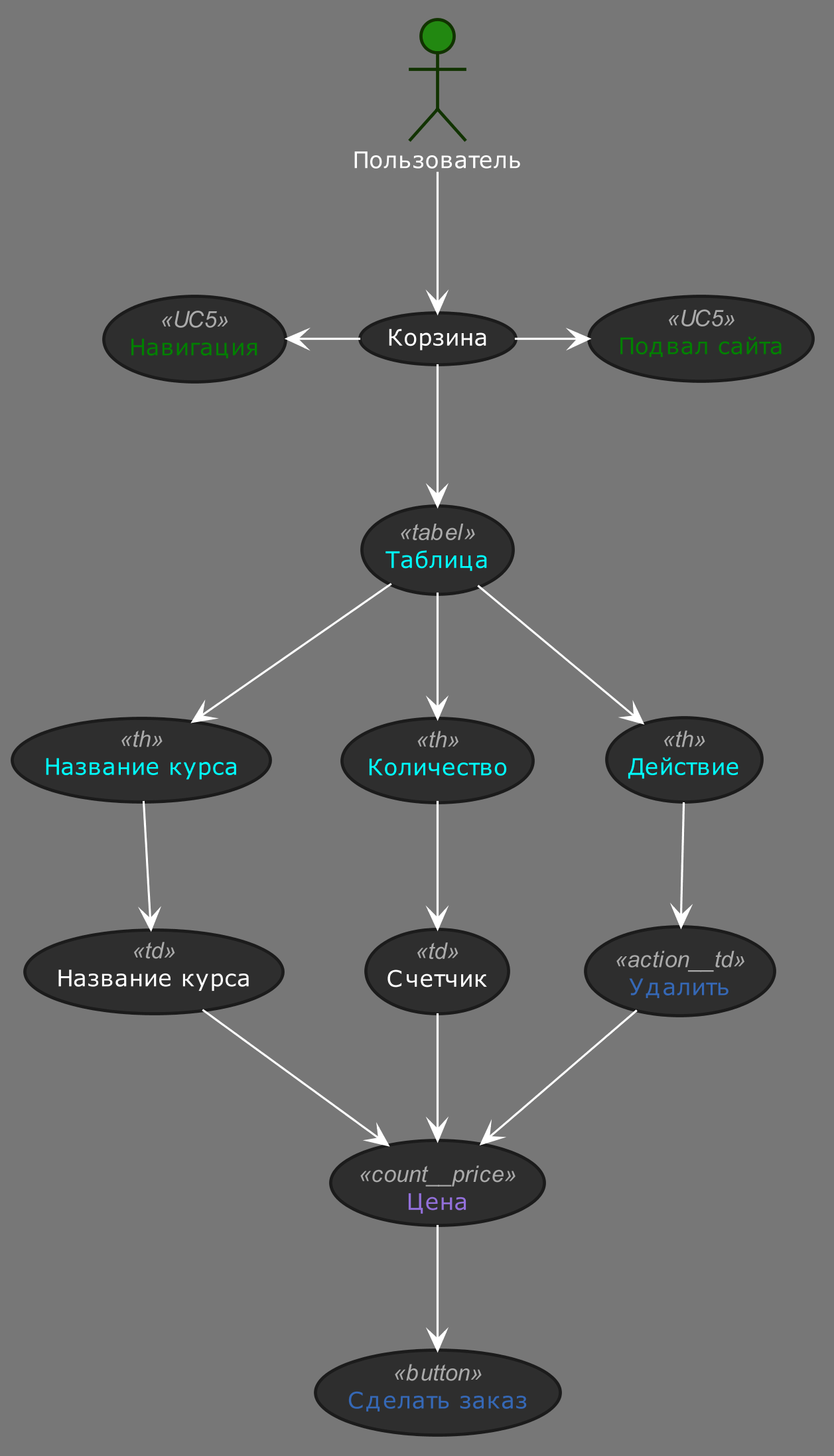


Рисунок 2.6 – UC-8 Страница «Корзина».

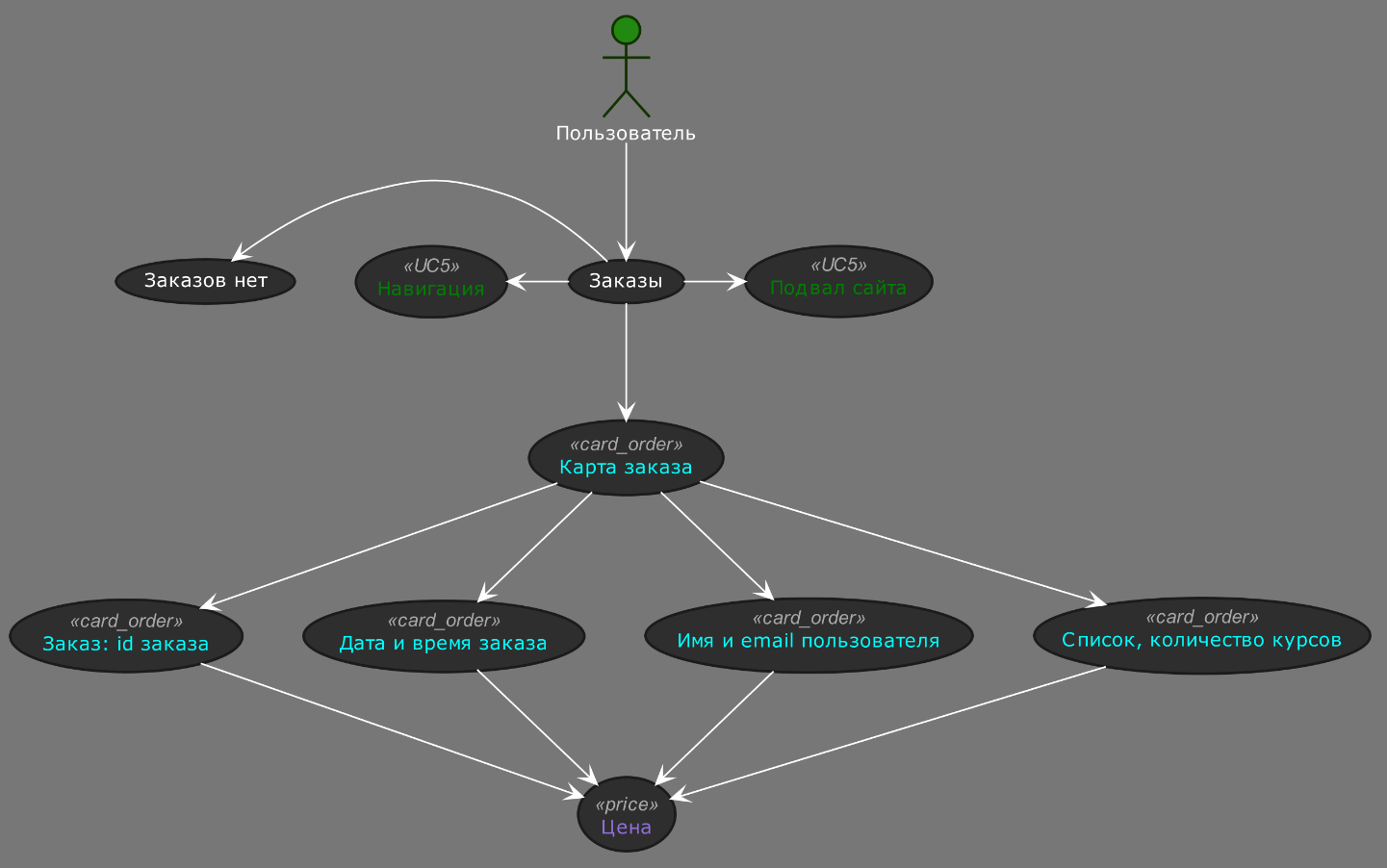


Рисунок 2.7 – UC-9 Страница «Заказы».

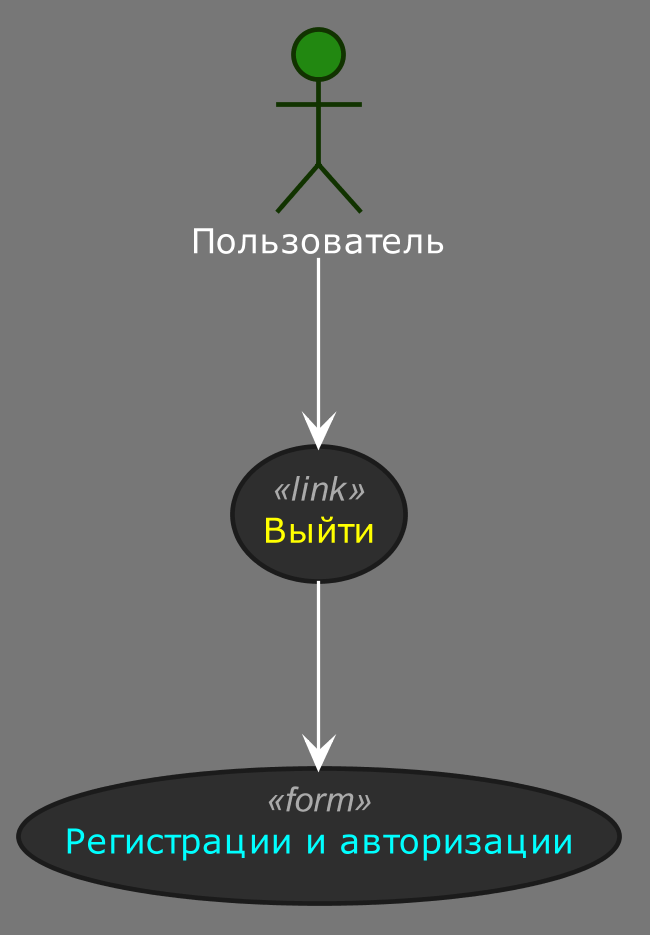


Рисунок 2.8 – UC-10.

При получении технического задания от заказчика, лучшим способом что бы ничего не пропустить, это составить спецификации требований.

Делается это для того, что бы избежать огромной массы не точностей и в первую очередь для того что бы избежать недопониманий с заказчиком. т.к. зачастую даже сам заказчик не видит конечного результата. Спецификации требований позволяют составить четко структурированный план, который предъявляется заказчику. В свою очередь, если заказчик захочет внести изменения, то это произойдет на этапе формирования пользовательского сценария, а не на этапе разработки программного обеспечения.

Пример UC1 форма регистрации и авторизации пользователя:

1. На странице присутствует две кнопки «Войти в магазин» и «Регистрация» которые позволяют переключатся между формами. А также имеется навигация не зарегистрированного пользователя и подвал сайта(footer) которые рассмотрены в UC2.
2. Форма «Регистрация» представляет собой четыре области ввода (input). А так же, над каждой областью ввода имеется соответствующее ему название.
   1. Имя
   2. Email
   3. Пароль
   4. Пароль еще раз
3. «Кнопка Зарегистрироваться» при которой должно происходить шифрование сессии с занесением пользовательских данных в hooks и в MongoDB для того, чтобы уменьшить шанс взлома приложения. После этого должно происходить перенаправление на страницу «Войти в магазин», где пользователь должен авторизоваться.
4. Форма «Войти в магазин» представляет собой две области ввода(input). ). А так же над каждой областью ввода имеется соответствующее ему название.
   1. Email
   2. Пароль
5. При нажатии «Войти» должна производится сверка Email и пароля в БД. После удачной авторизации должно происходить перенаправление на главную страницу интернет-магазина.
6. Ссылка «Забыли пароль?» При нажатии на которую должно происходить перенаправление на страницу «Сброса пароля».
7. Страница «Сброса пароля» сброса пароля представляет собой
   1. Label с названием Email
   2. Форму для ввода Email
8. Кнопку «Сбросить» при нажатии на которую должно отправляться письмо со следующим содержимым.
9. Тема письма: Восстановление доступа.
   1. Заголовок: Вы забыли пароль?
   2. Параграф: Если нет то проигнорируйте данное письмо. Иначе перейдите по ссылке.
   3. Ссылка содержит в себе первичный ключ, путь до страницы и сгенерированный в БД resToken и текст ссылки «Восстановление доступа».
   4. Вторая ссылка «Магазин курсов»
10. В письме при подтверждении пользователем о восстановлении доступа, путем нажатия на ссылку, должно происходить сравнение resToken отправленного в письме пользователю с resToken в БД. После чего должно происходить перенаправление на страницу ввода нового пароля.
11. Страница ввода нового пароля состоит из формы, которая представляет собой:
    1. Label Новый пароль
    2. Input Новый пароль
    3. Кнопка «Обновить пароль» при нажатии на которую должно происходить редактирование пользовательских данных в сессии, а именно замена существующего пароля на новый и перенаправление на страницу логина где с помощью ввода новых реквизитов пользователя должна происходить авторизация пользователя и перенаправление на главную страницу интернет магазина.

Довольна часто возникают проблемы при регистрации или при авторизации пользователя. Этот сценарий так же нужно предусмотреть.

При описании альтернативного сценария, в данном случае валидации формы, используют те же пункты и подпункты только добавляется символьное или числовое значение.

* + 1. Имя должно быть минимум 3 символа.
    2. Введите корректный email.
    3. Такой email уже занят
    4. Пароль должен быть не менее 6 символов
    5. Пароли должны совпадать.

Пример UC2 не зарегистрированный пользователь:

1. Не зарегистрированный пользователь при переходе по ссылке в интернет магазин попадает на форму регистрации и авторизации пользовательский сценарий которой расписан в UC1.
2. Область навигации состоит из ссылок
   1. Первая ссылка состоит из SVG картинки и «Магазин курсов». При клике на картинку или на ссылку происходит переход на главную страницу сценарий которой расписан в UC3.
   2. «Главная страница»
   3. «Курсы»
      1. Страница курсов для не зарегистрированного пользователя состоит из карточки товаров которая в свою очередь состоит:
      2. Картинки или логотипа товара.
      3. Название курса
      4. Цена курса
      5. Ссылки «Открыть курс» которая веден на страницу детального описания курса.
   4. «Войти»
3. Подвала сайта(footer) который состоит из трех основных блоков:
   1. Первый блок состоит из SVG планеты и ссылки «Магазин курсов». При клике на картинку или на ссылку происходит переход на главную страницу.
      1. Следующих реквизитов: свидетельство 0129174, 24 января 2020г. УНП 192761153, дата регистрации в торговом реестре РБ: 08.03.2020.
      2. SVG карты оплаты Visa, MasterCard, Maestro.
   2. Блок с контактной информацией.
      1. +375298918971
      2. +375448918971
      3. 210040 г. Витебск ул Чкалова 41/1 каб 312
   3. Блок со списком в который входят SVG ссылки социальных иконок.
   4. Альтернативный сценарий перехода по ссылкам ведет на страницу ошибки 404 которая состоит:
      1. Области навигации не зарегистрированного пользователя
      2. Заголовка «404 Ошибка»
      3. Параграфа «Страница не найдена»
      4. Ссылки «Перейдите по ссылке на главную страницу»

Пример UC3 главная страница:

1. Главная страница состоит из трех главных блоков
   1. Блок навигации
      1. Не зарегистрированный пользовател, «Курсы», «Войти».
      2. Зарегистрированный пользователь: «Главная», «Курсы», «Добавить курс», «Корзина», «Заказы», «Профиль», «Войти».
   2. Главный блок.
      1. Данное приложения является дипломным проектом слушателя группы ИС-17 Факультета переподготовки кадров Атрощенко Константина Анатольевича.
      2. Угловатый блок не правильной формы. О приложении: Данное приложение не является коммерческим. Это лишь демонстрация изученного материала. «Информационная система интернет магазина курсов» позволяет каждому зарегистрированному пользователю продавать свои курсы. Продавец свяжется с вами через email для уточнения и обсуждения оплаты. Более подробную информацию можно узнать при клике на сертификат.
      3. SVG ссылка сертификат.
   3. Подвал сайта пользовательское поведение которого описано в UC2.
   4. Альтернативный сценарий перехода по ссылкам ведет на страницу 404 пользовательский сценарий которой описан в UC2.

Пример UC4 Страница курсов:

1. Блок навигации описанный в UC3 блок footer описанный в UC2.
2. Блок курсов состоящий из карточек товаров.
   1. Карточка товара владельца курса состоит:
      1. Картинка или логотип.
      2. Название курса
      3. Цена курса
      4. Ссылки «Открыть курс»
         1. Происходит перенаправление на страницу детального описания курса.
      5. Ссылки «Редактировать курс»
         1. Происходит перенаправление на страницу редактирования курса.
         2. Состоит из области навигации описанной в UC3, подвала сайта описанного в UC2, формы редактирования курса.
         3. Форма редактирования курса состоит:
            1. Input placeholder «Введите название курса». Название курса должно подставляться автоматически в соответствии с редактируемым курсом т.е. placeholder должен заменяться на значение имени редактируемого курса.
            2. Валидация. Минимальная длинна названия 3 символа.
            3. Input placeholder «Введите цену». Значение цены должно подставляться автоматически в соответствии с редактируемым курсом т.е. placeholder должен заменяться на значение цены редактируемого курса.
            4. Валидация. Введите корректную цену.
            5. Input placeholder «Введите url картинки». URL картинки должен подставляться автоматически в соответствии с редактируемым курсом т.е. placeholder должен заменяться на значение url пути картинки редактируемого курса.
            6. Валидация. Введите корректный url адрес.
            7. Текстовая не должна выводить значение редактируемого курса. Вместо этого должен отображаться placeholder «Вставьте html шаблон описания курса».
      6. Валидация. Введите минимальное HTML описание курса. Однако значение выставлено в ноль для того что бы пользователь мог внести описание позже.
         1. Кнопки «Редактировать курс» при клике на которую должен происходить анализ редактируемых параметров и в случае введенных каких-либо новых реквизитов карточки товара, старые данные должны обновится на новые, а не затронутые параметры должны оставаться прежними.
         2. Кнопка «Удалить» производит полное удаление карточки товара. Перенаправляет на страницу курсов где происходит новый поиск всех курсов по id с последующей их отрисовкой.
      7. Кнопка «Купить» добавляет курс в корзину.

Пример UC5 Добавить курс:

1. Блок навигации описанный в UC3 блок footer описанный в UC2.
2. Форма добавления курса состоит:
   1. Input placeholder «Введите название курса»
      1. Валидация. Минимальная длинна названия 3 символа.
   2. Input placeholder «Введите цену курса»
      1. Валидация. Введите корректную цену
   3. Input placeholder «Введите url картинки»
      1. Валидация. Введите корректный url картинки.
   4. Textarea placeholder «Вставьте html описание курса»
      1. Валидация. Введите минимальное HTML описание курса. Однако значение выставлено в ноль для, того что бы, пользователь мог внести описание позже.
   5. Кнопка «Добавить курс». После нажатия должен сгенерироваться id данной карточки товара и перенаправить на страницу курсов, где происходит отрисовка всех курсов.

Пример UC6 Корзина:

1. Блок навигации описанный в UC3 блок footer описанный в UC2.
2. Блок корзины состоит:
   1. Если в корзине нет добавленных курсов то выводится параграф «Корзина пуста».
   2. При добавлении курса в корзину формируется таблица которая состоит из:
      1. Заголовков:
         1. Название курса
         2. Количество
         3. Действие
      2. Строк:
         1. Добавляется название курса
         2. Пользователь имеет право заказать несколько единиц одного и того же курса. Счетчик должен проанализировать количество добавленных единиц и отобразить их в графе количество.
         3. В графе действие находится кнопка «Удалить» при нажатии на которую должен срабатывать счетчик и пересчитать количество добавленных курсов и их единиц (одноименных курсов) в корзину.
   3. Цена. При любом срабатывании счетчика цена перерассчитывается.
   4. Кнопка «Сделать заказ». При ее нажатии формируется карточка заказа и добавляется на страницу «Заказы».

Пример UC7 Заказы:

1. Карточка заказов состоит из:
   1. Если пользователь ничего не добавил в заказы выводится параграф «Заказов нет».
   2. Span Заказ и id заказа.
   3. Дата и время заказа.
   4. Имя и email пользователя.
   5. Список и количество единиц заказываемых курсов.
   6. Цена.

Пример UC8 Профиль:

1. Блок навигации описанный в UC3 блок footer описанный в UC2.
2. Профиль состоит из:
   1. Аватар.
      1. Параграф «Аватара нет» если пользователь не загрузил фотографию.
   2. Параграфа «Ваш email» где отображаю email который был указан при регистрации пользователя.
   3. Label Ваше имя.
   4. В input, отображаю имя указанное при регистрации пользователя.
   5. Кнопка «Выберите файл» для загрузки картинки из файловой системы пользователя.
   6. Input в котором отображается имя и расширение загружаемого файла.
   7. Кнопка «Изменить» при нажатии на которую происходят изменения которые внес пользователь а именно: изменил имя, выбрал файл для загрузки.

Пример UC9 Выйти:

1. Кнопка «Выйти» позволяет пользователю закончить сессию и выйти из системы интернет магазина.
2. Кнопка «Войти» перенаправляет пользователя на станицу логина где пользователь может вновь авторизоваться и зайти в систему интернет магазина.

Такой последовательный вариант описания требований, в виде пунктов и подпунктов сильно упрощает работу. Данный вариант я считаю лучшим т.к некоторые делали в блок схеме было бы рисовать крайне сложно.

По этим же сценариям можно проводить тестирование системы. И если на каком – то этапе возникают сложности, систему можно доработать.

* 1. Описание аналогов системы
  2. Обзор и обоснование выбора инструментальных средств

Редактор – Visual Studio Code.

Visual Studio code – это легкий и мощный редактор исходного кода. При правильной настройке его можно превратить в настоящую IDE. Он поставляется со встроенной поддержкой JavaScript, TypeScript и Node.js и имеет богатую экосистему расширений для других языков таких как С++, С#, Java, Python, PHP, Go и сред выполнения таких как .NET и Unity.

Как и многие другие редакторы кода, VS Code использует общий пользовательский интерфейс. В панели слева показывается древовидная иерархия папок и файлов – это структура проекта. Справа показано содержимое открытых файлов.

Интерфейс VS Code делится на пять основных областей.

* Редактор – основная область для редактирования файлов.
* Боковая панель – содержит различные виды что бы помочь при работе с проектом
* Строка состояния – информация об открытом проекте и файлах
* **Панель активности** - расположена в крайнем левом углу, она позволяет переключаться между представлениями и дает дополнительные индикаторы, зависящие от контекста, например, количество исходящих изменений, когда Git включен.
* **Панели** – Внизу я могу отображать различные панели под областью редактора для вывода или отладки информации, ошибок и предупреждений или встроенного терминала. Панель также можно перемещать вправо для увеличения вертикального пространства.

Для своего проекта я использую встроенный терминал для установки различных зависимостей, библиотек и т.д.

Для настройки VS code я использую следующие плагины:

1. Auto Close Tag - автоматически добавляет закрывающий тег при вводе закрывающей скобки открывающего тега. Область применения языки гипертекстовой разметки HTML/ XML, а так же для PHP, Vue, JavaScript, TypeScript, JSX и т.д.
2. Auto Complete Tag – объединяет функции автоматического закрытия тегов и автоматического переименования тегов.
3. Bracket Pair Colorizer – подсвечивает скобки разными цветами, что прекрасно помогает при навигации, визуализации вложенных функций. Можно настроить список цветов, а также определенный цвет для лишних скобок. Данная функция плагина прекрасно помогает при поиске ошибочно напечатанных скобок.
4. Code Spell Checker – данный плагин проводит базовую проверку орфографии которая хорошо работает с кодом Camel Case.
5. CSS Peek – Данный плагин настраивает навигацию между классами прописанными в HTML, JS, JSX и тд и классами прописанными в каскадной таблице. Для быстрого поиска классы в каскадной таблице нажимаю правой кнопкой мыши на классе прописанном в HTML и выбираю *Перейти к определению*, происходит переход к нужному классу в каскадной таблице стилей.
6. CSSTree validator – плагин проводит проверку CSS в соответствии со спецификациями W3C и реализациями браузера. Расширение подсвечивает неправильные свойства и их значения.
7. EditorConfig for VS Code- данный плагин в фоновом режиме проверяет кодировку каждого открытого файла. Контролирует табуляцию.
8. ESLint – статически анализирует код JavaScript. Находит ошибки. Многие проблемы, обнаруженные этим плагином, могут быть исправлены автоматически. Так же можно написать собственные правила, которые работают со встроенными правилами.
9. Git History Diff- данный плагин позволяет посмотреть детальную историю коммитов. Посмотреть какие изменения были внесены в файл, проект. Работа с ветками проекта и многое другое.
10. HTML CSS Support – этот плагин автоматически дополняет названия id классов на основе определений найденных в рабочей области или внешних файлов.
11. JavaScript (ES6) – это расширение содержит фрагменты кода для JavaScript в синтаксисе ES6.
12. JS Refactor – это инструмент автоматического рефакторинга Javascript для кода Visual Studio, созданный для упрощения и оптимизации процесса разработки. Он предоставляет обширный список автоматизированных действий, включая обычно необходимые: метод извлечения, извлечение переменной, встроенная переменная и псевдоним для переименования встроенного кода VS. JS Refactor также поддерживает множество общих фрагментов и связанных действий для упаковки существующего кода в общие выражения блоков.
13. Prettier - Code formatter - средство форматирования кода. Упорядочивает код.
14. SASS – подсвечивает синтаксис SASS, автоформатирует и заполняет код.
15. W3C Validation - Расширение для включения поддержки проверки W3C библиотекой. Это приложение требует установленной JDK. После установки нужно настроить переменные среды.
16. PlantUML – С моей точки зрения это наиболее адекватный инструмент для создания UML диаграмм. Смысл этого инструмента прост, он позволяет задавать диаграммы в виде текста, описывающего элементы и связи между ними. Графический вид создается автоматически. Плюсы PlantUML:
17. Диаграммы задаются в виде текста, а значит в любом текстовом редакторе можно найти требуемый текст в наборе файлов и заменить его.
18. Рефакторинг вполне удобен.
19. Текстовый формат дает возможность организовывать групповую работу и отслеживать изменения в системе контроля версий.

NodeJS.

Платформа NodeJS – это программная платформа, основанная на движке v8 (транслирующем JavaScript в машинный код) превращая JavaScript из узкоспециализированного языка в язык общего назначения.

Делает он это с некоторыми особенностями, которые ранее не были учтены другими разработчиками. Он очень эффективно и очень быстро транслирует JavaScript т.е. этот движок понимает какие куски кода нам не нужны, и он не тратит время на их выполнение.

Раин Даль в 2009 году, после нескольких лет экспериментов, разработал очень эффективный серверный язык, который отличается по парадигме от старых подходов т.е. были однопоточные, многопоточные среды разработки. Он разработал серверный язык на событиях. JavaScript событийно-ориентированный язык. По сути, он вытащил этот язык из браузера и написал пару плагинов на С++. В итоге, он получил движок, который транслирует JS отдельно от браузера. При этом он добавил различные устройства ввода вывода. Добавил определенный функционал при работе с файлами, для создания различных серверов и прочего.

NodeJS – это просто движок V8 который транслирует JS. Поверх него написана оболочка на C++. Это абсолютно отдельная сущность, которая способна понимать JS, причем делает это очень эффективно, и она может работать где угодно. А с учетом того, что есть C++ он может запускаться где угодно. Он не ограничен возможностями браузерного JS.

К примеру. В 2009 году, когда NodeJS только вышел, про модульность в JS не было и речи. Никто не знал про ES6 import и ничего такого не было.

В NodeJS появились модули, и разработчики смогли писать JS в разных файлах, причем делать это очень просто и нативно. Появилась возможность работать с файлами, создавать сервера.

Сейчас, по сути, из-за того, что JS использует событийно-ориентированный подход – это очень оптимизирует данную платформу, выполнение кода, потому что выполняются только те куски, которые нужны. У него есть концепт, который называется EventLoop – это определенная очередь куда складываются события и за счет этого разработчики получают очень оптимизированную платформу для работы с JS.

Плюсы использования NodeJS. Во - первых это тот же самый JS. JS во front-end – это монополист, потому что нет конкурента языку JS на front-end. Даже если сайт будет на php, все равно на front-end будет использоваться JS. Плюс в том, что, зная один язык я могу писать серверную часть. Помимо этого, NodeJS позволяет использовать различные сборщики пакетов, к примеру Webpack который я использую в своем проекте и который будет рассмотрен чуть ниже.

А также позволяет использовать менеджер пакетов npm (аббр. node package manager) – это стандартный менеджер пакетов, автоматически устанавливающийся вместе с NodeJS. Он используется для скачивания пакетов из облачного сервера npm, либо для загрузки пакетов на сервера.

Основная его задача – это менеджмент различных модулей. Все приложения NodeJS начинаются с данного инструмента. Для того что бы установить все зависимости необходимые для разработки ПС необходимо инициализировать проект. Что делается с помощью пакетного менеджера npm.

Для инициализации проекта в консоли ввожу команду:

* npm init

Далее в package name необходимо ввести название проекта. По умолчанию npm берет название от папки в которой сейчас находится. Задаю название папки diplom-VSTU. В данной папке будет храниться разрабатываемо приложение.

Следующим шагом предлагается выбрать версию данного пакета, по умолчанию она стоит в значении 1.0.0.

Далее предлагается ввести description данного приложения. Заполняется по желанию.

Следующий пункт, entry point, спрашивает какой у нас будет входной файл в наше приложение. По умолчанию это index.js. Лучше всего оставлять этот файл потому что подключаемые в будущем библиотеки используют именно этот файл как точку входа в приложение.

Далее предлагается ввести test command. Оставляю по умолчанию.

Далее имеется возможность ввести git repository. Вводится по желанию.

В поле keywords можно вписать ключевые слова которые относятся к приложению.

Далее npm спрашивает кто является автором данного пакета. Я ввожу Konstantine899 и в треугольных скобках можно добавить email данного человека.

License обычно оставляют по умолчанию.

После всего npm спрашивает согласны вы с внесенными данными?

После соглашения в корне приложения появляется файл package.json. Этот файл хранит всю необходимую информацию о приложении. Этот файл используется для ведения списка зависимостей приложения.

Данный файл разбит на три области.

* Scripts – сюда входят скрипты запускающие зависимости.
* Dependencies – зависимости приложения.
* devDependencies – зависимости необходимые для разработки приложения.

Рассмотрим зависимости необходимые для разработки данного приложения:

1. Nodemon. Данный пакет используется для улучшения процесса разработки и каждый раз не перезагружать сервер. Сервер перезапускается автоматически каждый раз после того, как в какой-нибудь из файлов было внесено изменение.
2. Framework express.js – это один из самых популярных легковесных фреймворков, используемый при создании веб-приложений для node.

Все возможности express.js будут рассмотрены в «Главе 3.1 Разработка архитектуры программного продукта».

1. **Webpack** представляет собой **пакет**статических модулей для современных приложений JavaScript. Когда веб-пакет обрабатывает приложение, он самостоятельно строит [граф зависимостей,](https://webpack.js.org/concepts/dependency-graph/) который отображает каждый модуль, в котором нуждается проект, и генерирует один или несколько пакетов. Можно без преувеличений сказать что webpack сформировал современный шаблон (pattern) сборки web – приложений. Для того что бы опубликовать JS приложение не достаточно просто забросить код на web-server. Для этого в JS проектах используется шаг сборки и обработки кода. Webpack берет главный файл приложения(index.js), находит все модули которые используются в этом файле а затем анализирует каждый модуль, находит его зависимости и т.д. пока не будет построен полный граф зависимостей всех файлов, всех модулей. Во время сборки код каждого файла, каждого модуля можно провести через серию трансформаций и оптимизаций за которую отвечает babel который работает в связке с webpack. К примеру, можно использовать препроцессор scss для того что бы преобразовать scss синтаксис в каскадную таблицу стилей css. Когда все модули загружены webpack создает bundle (пакет) который передает браузеру.
2. Handlebars – это движок для динамического составления HTML файлов. Это один из основных движков который позволяет достичь большого количество функционала при создании и генерации HTML файлов. Он содержит в себе обычный классический HTML но благодаря некоторым конструкциям и интерполяциям он позволяет вводить некую динамику.
3. MongoDB – это NoSQL (не реляционная база данных). MongoDB в отличие от реляционных баз данных предлагает документо-ориентированную модель данных, благодаря чему работает быстрее, обладает лучшей масштабируемостью. Хранение данных в MongoDB схоже с хранением данных в формате JSON (JavaScript Object Notation). Для хранения данных в MongoDB применяется формат, который называется BSON (БиСон) или сокращение от binary JSON. BSON позволяет работать с данными быстрее: быстрее выполняется поиск и обработка.

Реляционные базы данных хранят строки, а MongoDB хранит документы. В отличие от строк документы могут хранить сложную по структуре информацию. Документы представляют собой хранилище ключей и значений.

Если в реляционной базе SQL есть таблицы, то в мире MongoDB есть коллекции. Реляционные базы данных хранят однотипные жестко структурированные объекты, то в коллекциях содержатся самые разные объекты которые имеют различную структуру и различные наборы свойств.

Система хранения в MongoDB представляет собой набор реплик. В наборе имеется основной узел, а так же может быть набор вторичных узлов. Все вторичные узлы сохраняют целостность и автоматически обновляются с обновлением главного узла. Если основной узел выходит из строя, то один из вторичных узлов становится главным.

Отсутствие жесткой схемы базы данных, при изменении концепции хранения данных, значительно упрощают жизнь разработчикам т.к. не надо думать о пересоздании базы данных и тратить время на построение сложных запросов.

1. Mongoose – это библиотека JavaScript используемая в приложении Node.js с базой данных MongoDB.

Mongoose – это ODM ( Object Document Mapper - объектно-документный отобразитель). Это означает, что Mongoose позволяет вам определять объекты со строго-типизированной схемой, соответствующей документу MongoDB.

Mongoose предоставляет огромный набор функциональных возможностей для создания и работы со схемами. На данный момент Mongoose содержит восемь SchemaTypes (\* типы данных схемы), которые может иметь свойство, сохраняемое в MongoDB. Эти типы следующие:

* String
* Number
* Date
* Buffer
* Boolean
* Mixed
* ObjectId (уникальный идентификатор объекта, первичный ключ, \_id)
* Array

Для каждого типа данных можно:

* Задавать значение по умолчанию
* Задать пользовательскую функцию проверки данных
* Задать get-функцию которая позволяет проводить манипуляцию с данными до их возвращения в вид объекта
* Задать set-функцию которая позволяет проводить манипуляцию с данными до их сохранения в базу данных
* Определить индексы для более быстрого получения данных.

Типы данных Buffer, Mixed, ObjectId и Array относятся только к Mongoose.

Buffer позволяет сохранять двоичные данные. Тип Mixed используется для превращения свойство в неразборчивое поле. В данном поле допустимы данные любого типа. Тип ObjectId используется для определения ссылки на другой документ в БД. Тип Array позволяет сохранять JavaScript-подобные массивы. Благодаря этому типу данных можно выполнять типичные JavaScript операции над массивами.

1. Сессии npm install express-session
2. Для сохранения сессии в MongoDB я использую пакет npm install connect-mongodb-session. Данный пакет синхронизирует express-session с базой данных.
3. За шифрование пароля нужно установить новый пакет npm install bcryptjs (59)
4. Для того что бы защитить приложение нужно будет генерировать определенные ключи которые будут уникальны для данного пользователя. И для этого в express существует отдельный пакет npm install csurf. Данный пакет защищает весь front – end и сравнивает с определенной переменной. (60)
5. Для того что бы в форме регистрации и авторизации передавались ошибки необходимо подключить пакет npm install connect-flash это middleware.
6. В PHP есть функция email которая позволяет отправлять email на любой почтовый адрес. Так вот в Node.js это не так. В Node.js нужно пользоваться сторонними сервисами которые предоставляют данную возможность. Скачиваются определенные пакеты которые связываются с этими сервисами и уже через сервис работать с email.

Один из таких сервисов SendGrid (<https://sendgrid.com/>). Плюс этого сервиса в том что до определенного плана он может быть бесплатен. Он крайне прост в настройке и удобен в использовании. Бесплатный план позволяет обрабатывать сто тысяч писем в месяц.

Нажимаю на **Start Free** и заведем здесь новый акаунт.

После регистрации и авторизации перехожу в левое меню Settings и во вкладку [API Keys](https://app.sendgrid.com/settings/api_keys).

Опиши настройку

Npm install nodemailer общий пакет включающий в себя другие транспортеры. И второй пакет называется npm install nodemailer-sendgrid-transport

Деплой приложения с помощью Heroku. Подключаю специальный пакет который будет добавлять определенные **header** - ры для запросов. Для того что бы избежать определенных атак и таким образом защитить приложение. Пакет называется [helmet](https://github.com/helmetjs/helmet).

Данный пакет служит для того что бы защищать express **aplication** передавая им определенные **HTTP headers**. Читай как все это использовать.

Устанавливаю пакет. npm install helmet --save

Следующий пакет который мы подключим будет служить для того что бы сжимать определенные статические файлы которые есть в приложении. [compression](https://www.npmjs.com/package/compression). Здесь можно найти различные способы конфигурации данного пакета.

Подключаю в **index.js const compression = require('compression');** Далее регистрирую данный **middleware app.use(compression());**

npm i compression

**Немного переформулирую. Проанализировав главные инструменты. Не пиши их устанавливаемы пакеты и не описываю методы их настройки. Сделай это в главе разработки.**

**Сейчас дай общее понятие следующим оиструментам:**

* **Visual Studio Code**
* Сам язык JavaScript
* Node.js
* Framework express
* Webpack и Babel
* MongoDB
* Mongoose
* SengGrid
* Heroku

@handlebars/allow-prototype-access

Bcryptjs

Compression

Connect-flash

Connect-mongodb-session

Csurf

Express-compile-sass

Express-handlebars

Express-session

Express-svg-fallback

Express-validator

Helmet

Multer

Node-sass-middleware

Nodemailer

Nodemailer-sendgrid-transport

Uuid – C помощью этой библиотеки можно генерировать различные индификаторы. npm i uuid. (26)

Winston

devDependencies

@babel/core

@babel/preset-env

Autoprefixer

Babel-loader

Babel-minify-webpack-plugin

Concurrently

Css-loader

Css-nano

Eslint

Eslint-config-airbnb

Eslint-config-node

Eslint-config-prettier

Eslint-plugin-html

Eslint-plugin-import

Eslint-plugin-jsx-ally

Eslint-plugin-node

Eslint-plugin-prettier

Eslint-plugin-react-hooks

Mini-css-extract-plugin

Node-sass

Postcss-loader

Prettier

Sass-loader

Webpack

Webpack-cli

1. Проектирование
   1. Разработка архитектуры програмного продукта
2. Nodemon. Для установки данного пакета используется команда npm install nodemon -D. Флаг -D указывает на то что данная зависимость нужна в режиме разработки приложения.
3. Framework express.js Запихни сюда webpack и babel.
4. Устанавливаю

Устанавливается с помощью команды npm install express.

1. Handlebars.Для установки данного пакета использую команду «npm i express-handlebars» (20, 21)

Если возникает ошибка с Handlebars, добавьте это решение (версия обновилась)

**Решение** — подключите allowInsecurePrototypeAccess

<https://www.npmjs.com/package/@handlebars/allow-prototype-access>

Пример использования в коде: <https://www.npmjs.com/package/@handlebars/allow-prototype-access#usage--express-handlebars-and-mongoose->

1. MongoDB
2. Mongoose npm install mongoose (37)
   1. Проектирование структур хранение данных
   2. Описание реализации вариантов использования
3. Реализация
   1. Разработка классов информационой системы
   2. Разработка интерфейса программного продукта
   3. Разработка алгоритмов реализации вариантов использования
   4. Модульное тестирование алгоритмов реализации вариантов использования
4. Системное тестирование
   1. Функциональное тестирование
   2. Оценка безопасности
   3. Тестирование производительности
5. Экономическое обоснование эффективности внедрения
6. Охрана труда(применительно к области внедрения)
7. Промышленная экология
8. Ресурсосбережение

Заключение

Список использованной литературы

Приложения

Ссылки на всякий случай так сказать

* <https://nodejs.org/ru/>
* модули <https://nodejs.org/api/>
* При установке самого NodeJS мы устанавливаем еще и встроенные модули которые позволяют работать например <https://nodejs.org/api/os.html>
* Npm <http://prgssr.ru/development/vvedenie-v-paketnyj-menedzher-npm-dlya-nachinayushih.html>