СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Обеспечение производства продуктов питания в количестве и ассортименте, достаточных для устойчивого продовольственного снабжения населения, — такая задача на ближайшие годы стоит в Беларуси перед приоритетным национальным проектом в области сельского хозяйства.

Однако достаток и даже изобилие пищевых продуктов еще не означает автоматического внедрения принципов рационального, правильного, сбалансированного питания в повседневную личную жизнь людей.

В эпоху научно-технического прогресса в связи сизменившимися условиями труда и быта возникла проблема предупреждения заболеваний, связанных с избыточным и нерациональным потреблением пищи и малоподвижным образом жизни или мышечной ненагруженностью (гиподинамией). Все чаще встречаются болезни, возникающие вследствие нарушения обмена веществ (ожирение, сахарный диабет и др.).

Вместе с тем, за последние 15 лет, доходы значительного количества населения страны резко снизились, что также сказалось на качестве и количестве потребления пищи. В результате увеличилось количество болезней, связанных с недостаточностью и низкой калорийностью питания.

В связи с этим в настоящее время актуальной становится проблема повышения культуры питания, с тем чтобы рацион питания соответствовал энергетическим затратам и физиологическим потребностям организма. Рациональное использование пищевых продуктов каждым человеком, исключение переедания и недоедания, поможет многим укрепить здоровье.

Целью данного курсового проекта является повышение уровня комфорта обслуживания, благодаря созданию удобной системы заказа и доставки еды.

Для того, чтобы достичь поставленной цели необходимо найти решение следующих задач:

- исследовать процессы сервиса заказа и доставки еды;
- описать исследованные процессы, построить на основе описанного функциональную модель;
- осуществить проектирование системы с использованием UML-диаграмм;
- создать базу данных;
- разработать сервис для заказа и доставки еды;
- протестировать полученный программный продукт;
- описать руководство пользователя по работе с данным сервисом.

- поддерживать программный продукт во время эксплуатации.

В результате решения всех поставленных задач, мы получим готовое программное обеспечение, способное удовлетворить среднестатистического пользователя. Приложение позволит позволит оптимизировать заказ еды, благодаря простому и удобнму интерфейсу. Кроме этого, приложение будет обладать системой оценок, благодаря чему пользователь всегда будет иметь представление о заказываемом продукте.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 История появления доставки еды

Услугой доставки еды на дом, в офис или на удаленные объекты сегодня сложно кого-то удивить. Более того, в некоторых странах, особенно развитых, она не просто составляет конкуренцию домашнему и традиционному общественному питанию, но и активно вытесняет их. Люди все чаще предпочитают заказать блюда по телефону или в Интернете, чем готовить самостоятельно или посещать заведения общепита. Однако так было не всегда, и история доставки готовых блюд как массового сегмента рынка питания началась лишь в XX и полностью сформировалась в XXI веках.

Массовое распространение этой услуги стало возможным только после того, как в обществе сформировались определенные условия, наиболее важными из которых являются:

1. Массовая урбанизация и повышение занятости населения – жители крупных городов зачастую проводят большую часть жизни на работе, в результате чего у них не остается времени на частое приготовление пищи или посещение заведений общепита;
2. Развитие транспортной инфраструктуры – для быстрой доставки еды необходимы обустроенные дороги и, в определенной степени, наличие мобильного транспорта;
3. Появление эффективных средств связи – услуга доставки питания не будет иметь смысла, если клиент не сможет связаться с поставщиком непосредственно из своего дома или офиса;
4. Промышленное производство продуктов питания – традиционное сельское хозяйство огородного или фермерского типа не способно покрыть потребности миллионов людей в пище.

Перед тем, как появилась доставка питания в современном ее понимании, возникновение подобных услуг происходило редко, хотя и нельзя отрицать наличие их аналогов даже в древние эпохи. Известно, что на всех крупных стройках, от пирамид Египта до грандиозных готических соборов Европы, помимо полевых кухонь имелись и люди, разносившие домашнюю еду для строительных бригад.

Как ни странно, но история доставки еды как отдельного и массового сегмента рынка началась не на Западе. Первопроходцем в этой области стала Индия, где в конце 19 века появилась профессия «даббавала» – пешего разносчика пищи, помещенной в специальные металлические судки «дабба».

Этому способствовало несколько факторов:

– интенсивная урбанизация региона, появление крупных городов – прежде всего Мумбаи, Дели;

– запутанная транспортная структура городов – офисным клеркам, рабочим было трудно добираться из удаленного офиса домой для приема пищи;

– популярность домашнего питания – в Индии традиционно не слишком любят общепит за низкое качество и санитарную безопасность пищи.

Система даббавал, существующая и сегодня, имеет несколько отличий от западных служб доставки еды. Главная особенность – приготовлением пищи является не отдельная организация, а обычное население. Каждое утро разносчик на велосипеде, скутере или пешком собирает судки с готовой едой у домовладельцев, с которыми заключен контракт, и начинают развозить их по клиентам. Так как в Индии даббавала традиционно формируются из малограмотных слоев общества, для координации своих действий разносчики разработали уникальную, не имеющую аналогов в мире, систему обозначений, основанную на визуальных символах и цвете. Она оказалась настолько совершенной, что вероятность ошибочной доставки составляет не более 1 раза на 6-8 миллионов заказов.

На Западе же данная услуга как массовое и распространенное явление появилась несколько позже, в начале XX века. Связывают ее с массовым строительством небоскребов в США – грандиозные на тот момент сооружения требовали труда сотен рабочих и их полной занятости. Наряду с традиционными в таких случаях полевыми кухнями некоторые учреждения общественного питания стали предлагать услугу по доставке холодных блюд, закусок и напитков. Ее популярности способствовало развитие эффективных и удобных средств связи – телефонов. Заказать еду стало возможным, просто набрав номер кафе или ресторана.

Но особенно сильное развитие рынка доставки еды произошло в 50-х годах XX века, захватив не только США и Европу, но и бурно развивающиеся государства Юго-Восточной Азии.

В Америке итальянские эмигранты и, позже, вернувшиеся из Европы американские солдаты стали основывать пиццерии, в том числе предлагавшие привоз пиццы на дом или в офис, такую же услугу стали предлагать и сети фаст-фуда.

В восточно - азиатских капиталистических странах (Японии, Южной Корее, Тайланде, на Тайване, позже – в Китае) появились предприятия,

изготавливавшие упрощенные вариации традиционных блюд – суши, лапши, разнообразных салатов и так далее.

В Европе особенно быстро развивалась индустрия доставки пиццы и разнообразных закусок (бутербродов, сэндвичей и т. д.).

Сильный толчок рынку доставки питания дало появление и развитие Интернета. Теперь клиенту не нужно предварительно узнавать меню конкретного поставщика – оно в наглядной форме стало размещаться прямо на сайте, на нем же можно и заказать обед с помощью онлайн-заявки или по телефону. Это значительно упростило и ускорило процесс, сделав заказ еды проще, чем ее собственноручное приготовление. С увеличением числа интернет-пользователей и совершенствованием самой технологии появились специализированные сервисы, занимающиеся исключительно доставкой блюд.

1.2 Актуальность способов заказа еды

Мобильные устройства постепенно завоевывают лидерство, становясь основным каналом заказа доставки готовой еды (Рисунок 1.2.1).

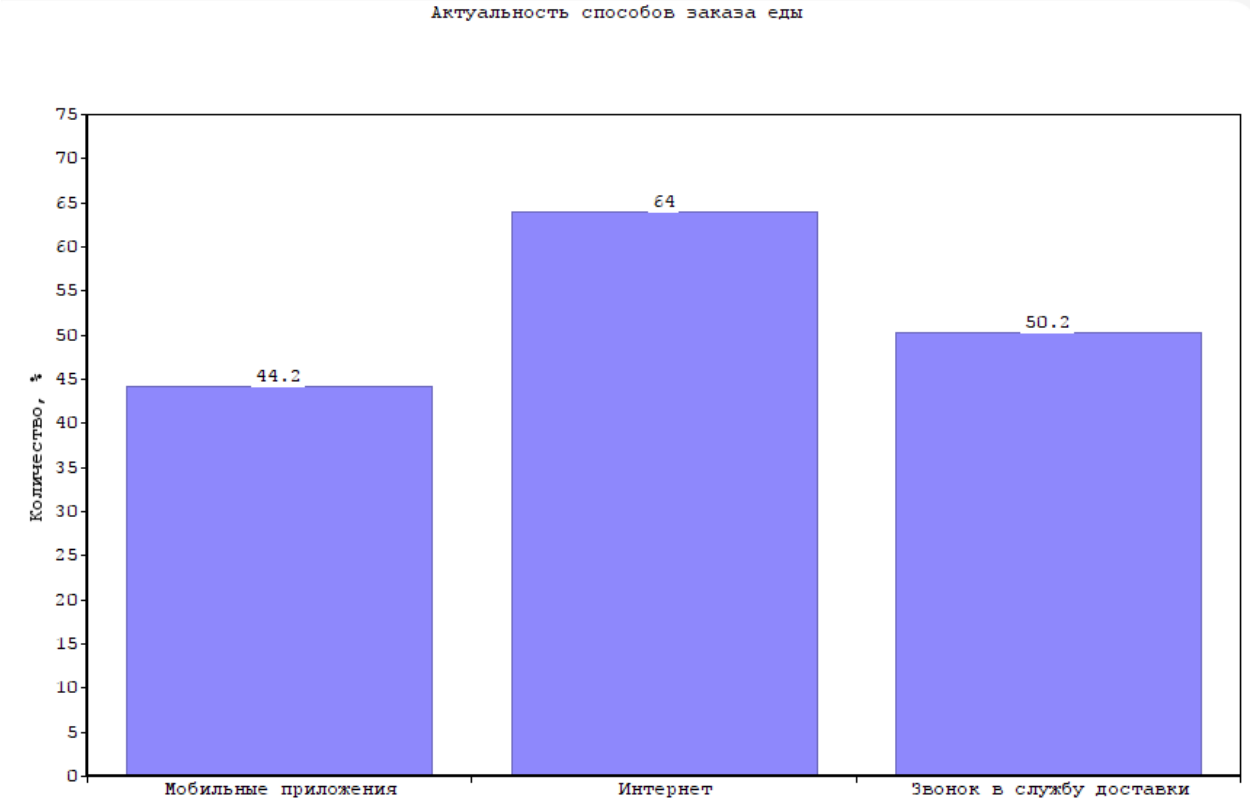

Рисунок 1.2.1 – Диаграмма актуальности способов заказа еды

На сегодняшний день уже 44,2% людей используют мобильные приложения ресторанов или служб доставки для совершения заказа. За последний год популярность подобного способа выросла на 6,8 %

Смена предпочтений с точки зрения канала общения бизнеса и потребителя произошла в 2017 году, когда доля заказов готовой еды, совершенных через интернет, впервые за 6 лет наблюдений превысила долю, сделанных по звонку.

Сегодня наблюдается уже переключение пользователей с web-страниц на мобильные приложения служб доставки. Сайты ресторанов и сервисов-агрегаторов теряют свою аудиторию. За последний год доля людей, заказывающих доставку готовой еды через интернет, снизилась с 68% до 64%.

Продолжают терять актуальность и звонки в службы доставки: за последний год отказались от привычки диктовать заказ по телефону 7%. В настоящее время к общению с диспетчерами и курьерами прибегают лишь 50,2% респондентов, в то время как годом ранее – данный показатель был на уровне 57,2%.

1.3 Нюансы запуска сервиса заказа и доставки еды

Построение собственной системы доставки еды лучше всего начинать постепенно. Зона охвата включает ближайшие улицы, легко досягаемые за 10 –15 минут. Первоначальные инвестиции в систему доставки еды на дом минимальны и включают затраты на рекламу, а также упаковочные материалы.

Сначала нужно установить ресторану «испытательный срок», в процессе которого ресторан начнет доставлять еду в пределах района своего местонахождения. За это время команда ресторана получит ценный опыт и протестирует свои фактические и потенциальные возможности по доставке еды. В процессе решения возникающих проблем ресторан научится оперативно и безошибочно осуществлять доставку еды на дом. После того, как ресторан отработает идеальную систему доставки еды в своем районе можно будет приступить к расширению и более широкому охвату новых районов города.

Построение собственной системы доставки еды на дом включает подготовку каналов по приему заказов:

1) телефонная линия по приему заказов и персонал, который будет принимать заказы;
2) СМС – канал по приему заказов еды на дом;
3) онлайн – система по приему заказов на сайте ресторана и т.д.

Как правило, служба доставки состоит из операторов, курьеров и менеджеров. Первые принимают заказы, вторые их развозят, а третьи организуют процесс, прорабатывают меню и отвечают за финансовые показатели работы службы. Иногда в штат приглашают упаковщиков и логистов.

Количество операторов зависит от интенсивности работы. Чем больше оборот доставки, тем больше операторов требуется.

Обычно устанавливается минимальная сумма заказа, с которой выгодно работать. Чем ближе адрес доставки, тем ниже минимальная стоимость. Если гость живет в соседнем доме, то официант или курьер может потратить 10 минут и отнести ему кофе. Но ради одного салата ехать на другой конец города не стоит.

Традиционно особенным успехом пользуются суши, роллы и пицца, однако их теснят иные меню, которые повары вместе с менеджерами адаптируют специально для доставки. Многие блюда, которые предлагают рестораны, практически нереально транспортировать куда-либо. В зону риска попадают десерты с мороженым или сложными украшениями, десерты, состоящие из нескольких частей. Некоторые блюда в силу технологических особенностей не могут повторно разогреваться, их тоже лучше исключить из меню доставки.

Для того, чтобы ваши потенциальные клиенты могли узнавать о том, что у вас есть доставка на дом, необходима реклама.

Напечатайте рекламные плакаты и повесьте их в метро, на улицах разместите билборды и раздавайте листовки. И не забывайте указывать на них адрес сайта, где можно будет посмотреть меню доставки, и номера телефонов, по которым можно сделать заказ. Продвигайте свою услугу доставки в социальных сетях.

Скорость доставки является одним из основных конкурентных преимуществ наряду с самим продуктом. Обеспечить эффективную работу на высоких скоростях возможно только с помощью современных программных технологий, позволяющих полностью автоматизировать процессы. Кроме того, это поможет сократить часть операционных расходов в дальнейшем, а также обеспечить необходимый контроль на каждом этапе прохождения заказа.

Услуги доставки на рынке общепита оказывают не только отдельные рестораны, но и агенты. Их преимущество – онлайн-заказ из разнообразных ресторанов. Как правило, ресторан сам платит агенту комиссионные, составляющие до 30% от стоимости заказа. Вариант удобен для тех, кому

невыгодно организовывать свою доставку, или для тех, кому хочется расширить рынок сбыта.

Помимо вышесказанного следует также учесть следующие нюансы:

1. Приготовьтесь к пиковым нагрузкам.

Настраивайте логистику с учетом пиковых нагрузок – например, выводите на эту смену больше поваров и курьеров или сделайте им гибкий график. Убедитесь, что в пиковые часы их достаточно, чтобы справляться с потоком заказов. Доставка еды всегда работает в режиме пиковых нагрузок.

2. Работа операторов.

На приеме заказов должен работать человек грамотный, вежливый, приветливый и способный улаживать возникающие конфликты. Приятная беседа с оператором call-центра – один из залогов того, что у клиента останется положительное впечатление от заведения.

3. Упаковка блюд.

Если неправильно упаковать еду, то все усилия по ее приготовлению уже не будут иметь значения, ведь она просто потеряет по дороге товарный вид. Упаковывать блюда можно в пластиковые контейнеры, но из прочного материала. Слишком мягкий пластик может помяться по дороге. Горячие блюда и горячие напитки должны доставляться в термо упаковках.

Горячие и холодные блюда не рекомендуется держать рядом во время доставки. Также нельзя «смешивать» очень ярко выраженные запахи разных типов блюд. Если курьер повезет клиенту в одном пакете мороженое и жареную рыбу, то клиент рискует попробовать десерт с рыбным запахом.

4. Мотивируйте курьеров.

Есть разные способы сотрудничества с курьерами: выплата им фиксированной суммы, выплата процента от заказа, выплата за выезд в определенную зону. От первого способа стоит максимально быстро отказаться, поскольку только последние два мотивируют курьера вернуться как можно быстрее. Есть множество сложных схем мотивации курьеров, один из них — создание «курьерских каст». Лучшие курьеры работали только в центре, и их было строго лимитированное число. Они зарабатывали больше остальных, но могли вылететь из этой касты, если доставляли заказ слишком долго, были невежливы и т. д.

5. Время доставки.

Быстрая доставка – это то, засчет чего можно победить всех конкурентов. Разумные сроки доставки – 1,5-2 часа с момента заказа. Но не больше. Более долгий период ожидания уже может доставить клиенту

дискомфорт. Именно по этой причине в доставку на дом редко включают блюда, в приготовлении которых невозможно использовать заготовки.

Если при запуске сервиса учесть все перечисленные факторы, то можно быть уверенным в том, что сервис по заказу и доставке еды будет успешным.

Непосредственно весь процесс подготовки и запуска такого сервиса представляет собой кропотливую работу, в которой задействовано много лиц. Им необходимо принять к сведению все возможные нюансы и разработать план действий в любой ситуации.

2 ОПИСАНИЕ ОСНОВНЫХ ПРОЦЕССОВ СЕРВИСА ЗАКАЗА И ДОСТАВКИ ЕДЫ

IDEF (I-CAM DEFinition или Integrated DEFinition) — методологии семейства ICAM (Integrated Computer-Aided Manufacturing) для решения задач моделирование сложных систем, позволяют отображать и анализировать модели деятельности широкого спектра сложных систем в различных разрезах. При этом широта и глубина обследования процессов в системе определяется самим разработчиком, что позволяет не перегружать создаваемую модель излишними данными. Одной из частных методологий моделирования IDEF является IDEF0.

IDEF0 - методология функционального моделирования. С помощью наглядного графического языка IDEF0, изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков - в терминах IDEF0). Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы.

В курсовом проекте изучаемой системой является деятельность сервиса заказа и доставки еды.

Реализация была выполнена с помощью программного средства All Fusion Process Modeler (Рисунок 2.1).

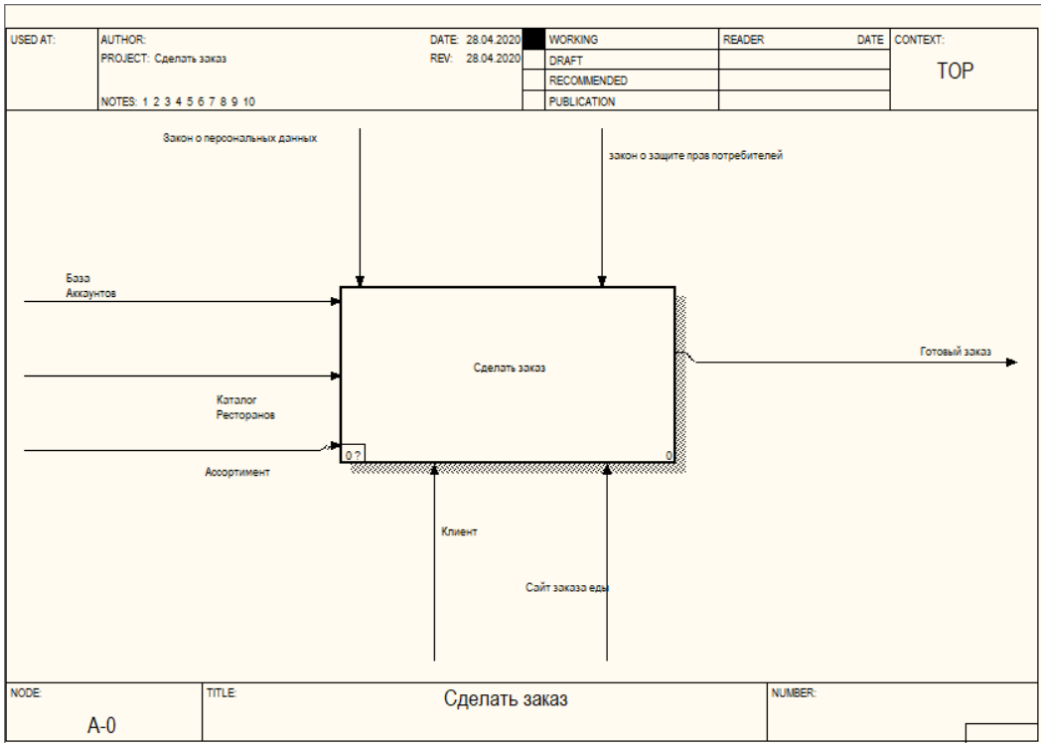

Рисунок 2.1 – Контекстная диаграмма верхнего уровня «Сделать заказ»

Блок «Сделать заказ» делится на 5 блоков: «Авторизоваться», «Выбрать заведение», «Выбрать позиции», «Указать адрес», «Подтвердить заказ» (Рисунок 2.2).

Блок «Авторизоваться» подразумеавет под собой вход пользователя в аккаунт и имеет следующие входные данные: закон о персональных данных, база аккаунтов.

Блок «Выбрать заведение» включает в себя просмотр всех заведений и выбор наиболее предпочтительного для клиента. Входные данные: каталог ресторанов, закон о персональных данных, аккаунт. На выходе получаем выбранное пользователем заведение.

Компонент «Выбрать позиции» состоит из просмотра всех позиций и выбора наиболее предпочтительных для клиента. В него входят следующие компоненты: ассортимент, выбранное завдение. На выходе получаем конкректные позиции в выбранных заведениях.

Блок «Указать адрес» подразумаевает выбор уже сохранненого адреса или добавление нового. Входные данные: конкректные позиции в выбранных заведениях. В результате получаем скопонованный заказ.

Компонент «Подтвердить заказ» на входе имеет скопонованный заказ, а на выходе готовый заказ.

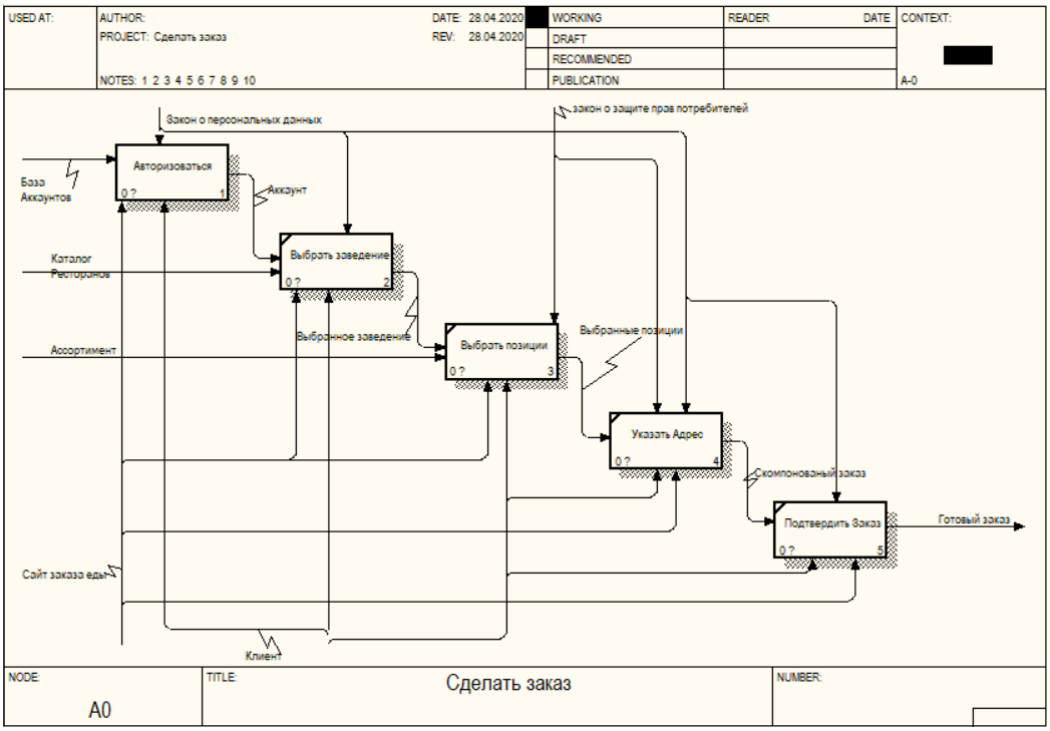

Рисунок 2.2 – Декомпозиция блока «Сделать заказ»

Декомпозиция компонента «Авторизоваться» представлена на рисунке 2.3 и содержит следующие блоки: «Зайти на сайт», «Ввод данных», «Успешный вход». Входным параметром является база аккаунтов, так как необходимо знать: существует аккаунт или нет, иначе сайт предложит зарегистрироваться. После успешной авторизации пользователь сможет просматривать ассортимент заведений и делать заказы.

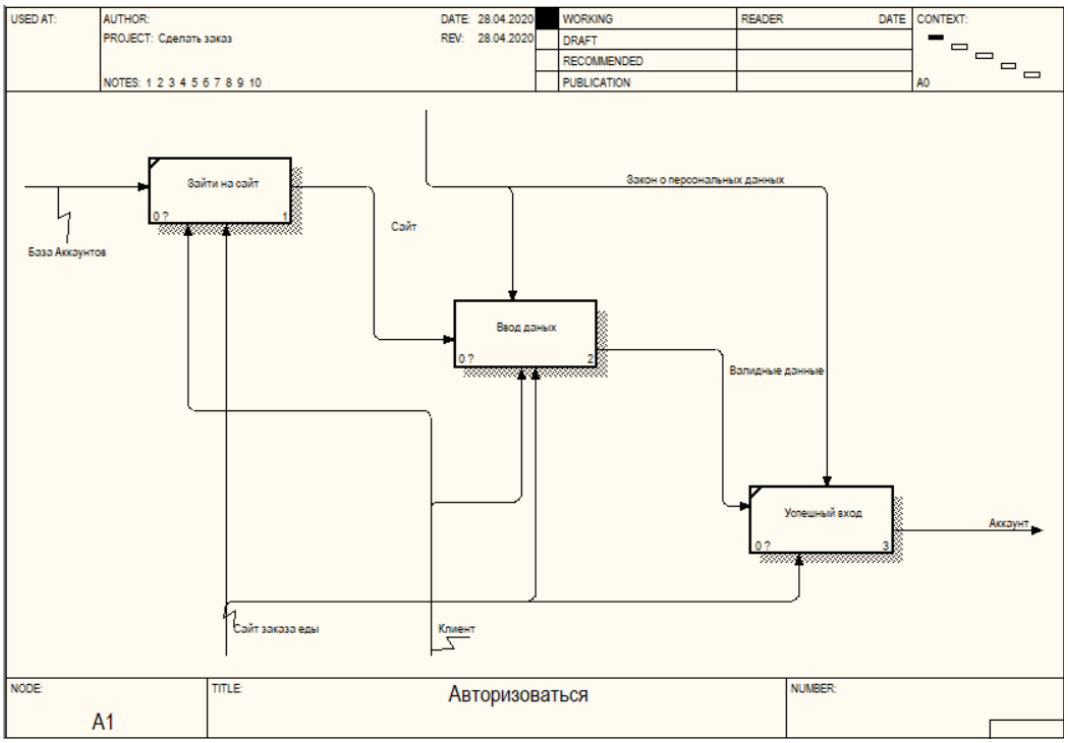

Рисунок 2.3 – Декомпозиция блока «Авторизоваться»

Декомпозиция ввода данных предоставлена на рисунке 2.4 и состоит из следующих компонентов: «Ввод», «Проверка на валидность», «Проверка на соответствие в БД». На данном этапе производится ввод данных и проверка их на правильность. Входными данными является база аккаунтов. На выходе получаем аккаунт конкретного пользователя. Все действия происходят при помощи самого клиент, поэтому он является механизмом.

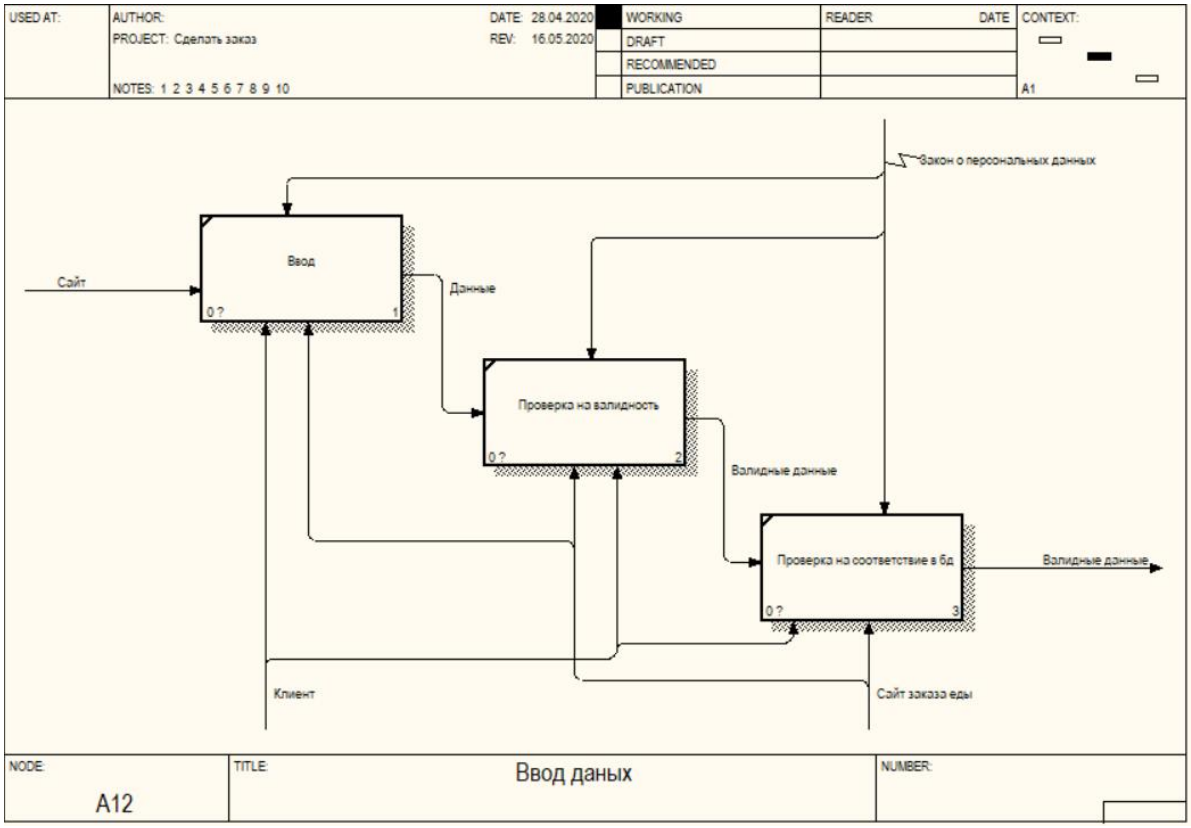

Рисунок 2.4 – Декомпозиция блока «Ввод данных»

Таким образом пользователи данного сервиса с легкостью могут сделать заказ, просмотрев оценку и ассортимент того или иного ресторана. Приятным дополнением станет возможность выбора метода оплаты заказа.

3 СПЕЦИФИКАЦИЯ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ СЕРВИСА ДЛЯ ЗАКАЗА И ДОСТАВКИ ЕДЫ

При взаимодействии актера с системой последняя выполняет ряд работ, которые образуют вариант использования системы (use case). Каждый актер может использовать систему по-разному, то есть инициировать выполнение разных вариантов использования. Таким образом, каждый вариант использования, по существу, есть некоторое функциональное требование к системе (которое может быть разбито на несколько более мелких). Варианты использования не представляют собой конструкцию, напрямую реализуемую в программном коде. Все его поведение реализуется в виде классов и компонент. Варианты использования описывают, что делает програмное средство, но не как она это делает. Каждый вариант использования обычно предполагает наличие нескольких вариантов поведения системы (потоков событий), один из которых является основным, остальные – альтернативными. Основной поток событий определяет последовательность действий системы, направленную на выполнение главной целевой функции данного варианта использования. Альтернативные потоки описывают поведение системы в исключительных ситуациях, например, при ошибках. Описание потоков событий может быть выполнено как в текстовой форме, так и с помощью диаграмм UML, которые будут рассматриваться в дальнейшем.

Диаграммы вариантов использования применяются при бизнес-анализе для моделирования видов работ, выполняемых организацией, и для моделирования функциональных требований к ПС при ее проектировании и разработке. Построение модели требований при необходимости дополняется их текстовым описанием. При этом иерархическая организация требований представляется с помощью пакетов use cases. Диаграмма вариантов использования представлена далее (Рисунок 3.1).

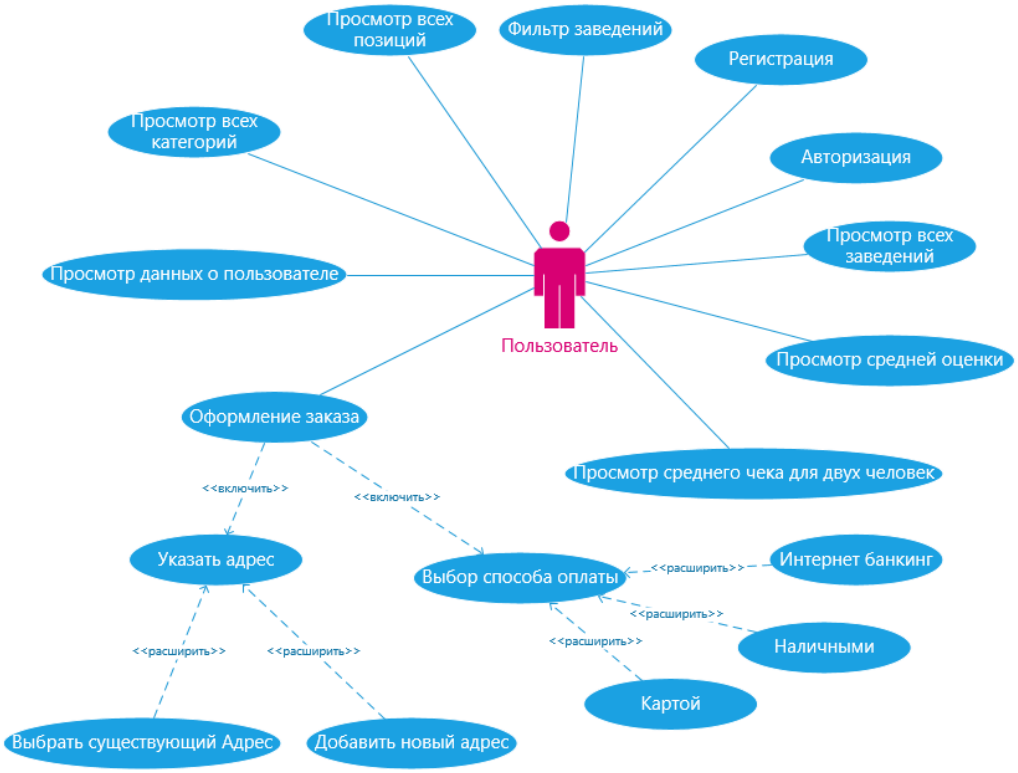

Рисунок 3.1 – Диаграмма вариантов использования сервиса заказа и доставки еды

При входе на сайт пользователю будет предложено авторизоваться или зарегистрироваться. После авторизации ему будет доступно все содержимое сайта.

Пользователь сразу увидит все рестораны, которые есть на сайте и сможет выбрать один из них. Так же он может найти нужный ему ресторан по названию.

После выбора конкретного ресторана пользователь сможет просмотреть всю иноформацию о нем: адрес, название, оценку, средний чек на двух человек, ассортимент и категории. Тут же он может выбрать позиции для последующего заказа.

Оформление заказа делится на несколько этапов: выбор позиций, ввод адреса и выбор способа оплаты. На всех этапах оформления заказа пользователь может изменить заказ или отменить его.

Аккаунт пользователя сохраняет адреса клиента и при повторном заказе он может выбрать уже существующий адрес в его аккаунте.

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СЕРВИСА ЗАКАЗА И ДОСТАВКИ И ЕЁ ОПИСАНИЕ

Для создания сервиса заказа и доставки еды потребовалось спроектировать шесть сущностей: ресторан, предмет, заказ, адрес, посетитель, способ оплаты. Модель изображена на рисунке 4.1.

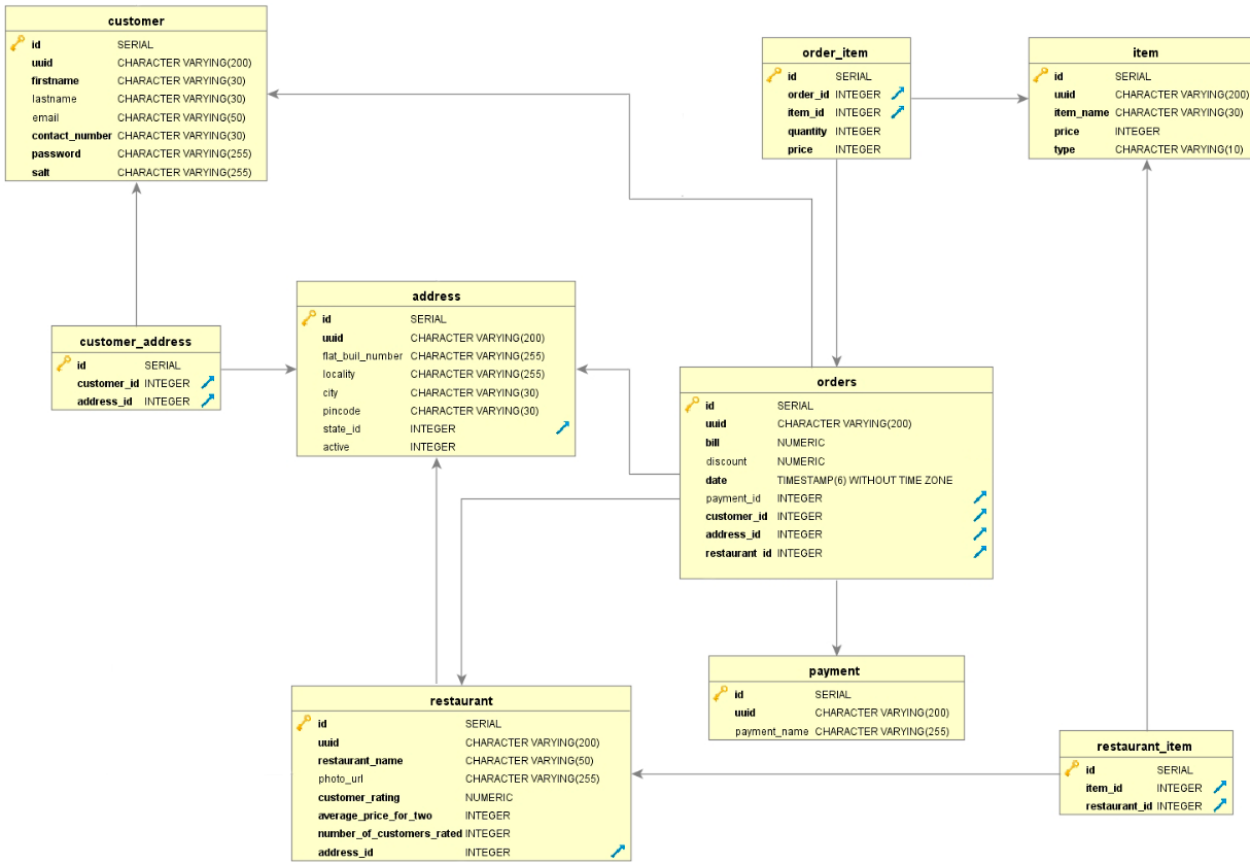

Рисунок 4.1 – Информационная модель системы

Сущность «Вещь» хранит в себе информацию о продоваемом ассортименте:

- id – хранит уникальный номер;
- uuid – хранит токен;
- name – наименование товара;
- price – цена;
- type – категория.

Сущность «Пользователь» хранит в себе информацию о зарегистрированных пользователях приложения:

- id – хранит уникальный номер;
- uuid – хранит токен;

- firstname – хранит имя пользователя;
- lastname – хранит фамилию;
- email – хранить электронную почту пользователя;
- number – хранит номер пользователя;
- password – хранит пароль;

Сущность «Адрес» хранит в себе информацию об адресах:

- id – хранит уникальный номер;
- uuid – хранит токен;
- flat – квартира;
- locality – расположение;
- city – город;
- pincode – код города;

Сущность «Оплата» хранит в себе информацию об методах оплаты: ееназвание, уникальный токен и уникальный номер.

Сущность «Заказ» хранит в себе информацию о заказах:

- id – храни уникальный номер;
- uuid – хранит токен;
- date – хранит дату заказа;
- paymentid – хранит уникальный номер метода оплаты;
- customerid – хранит уникальнй номер пользователя сделавшего заказ;
- addressed – хранит адрес доставки заказа;
- restaurantid – хранит уникальный номер ресторана, из которого совершен заказ;

Сущность «Ресторан» хранит в себе информацию о ресторане:

- id – хранит уникальный номер;
- uuid – хранит токен;
- restaurant name – название ресторана;
- photo url – хранит ссылка на фото ресторана;
- customer rating – хранит оценки посетителей;
- average price for two – хранит среднюю стоимость для двух людей;
- number of customers rated – хранит количество людей оценивших ресторан;
- addressed – хранит адрес ресторана;

В модели есть дополнительные сущности, которые отражают соответствия: предмета и заказа, предмета и ресторана, адреса и пользователя.

5 ОБОСНОВАНИЕ ВЫБОРА КОМПОНЕНТОВ И ТЕХНОЛОГИЙ ДЛЯ РЕАЛИЗАЦИИ КУРСОВОГО ПРОЕКТА

5.1 Технологии и средства для реализации курсового проекта

Данный курсовой проект разработан в среде IntelliJ IDEA на языке программирования Java.

IntelliJ IDEA — одна их тех сред разработки, которая пользуется почтением у настоящих профессионалов. Взирая на то, что за полноценную IDE нужно заплатить, ее используют немногие — лишь те, кому написание кода приносит хорошую результативность и прибыль. Сегодня об IntelliJ IDEA знает, без преувеличения, весь мир. Хотя платформа ориентирована, в первую очередь, на Java-кодинг, ее универсальность позволяет работать с разными языками. Сегодня у IntelliJ IDEA красивый и интуитивный интерфейс, который позволяет разобраться даже новичку.

В первую очередь, команда сконцентрировалась на создании средств для рефакторинга и анализа. Как раз из этого родилась современная IDEA. Развиваясь в этом направлении, создатели смогли разработать продукт, в котором код пишется частично вами, а частично машинным интеллектом. А все начиналось с того, чтобы обеспечить вовсе утилитарные функции: комфортное переименование классов, методов, автоматическое определение и прочие. С Java она дружит больше всего, отлично его понимает и помогает в написании разработчику. Но это не значит, что все заканчивается на Джаве и собственном языке Kotlin. Не менее важным является поддержка их коробки таких передовых технологий, как Groovy, Scala и прочих. Они сочетают в себе возможности динозавров, как Java вместе с функционалом Ruby, Smalltalk и прочих. Не забывают в компании и о трендах, ведь для большинства веб-языков созданы свои отдельные среды, основанные на IntelliJ IDEA

Java — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения.

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) — программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байт-код, в отличие от текста, обрабатывается значительно быстрее.

Java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты

Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов

Адрес подразумевает под собой идентификатор машины в пространстве сети Internet. Он может быть доменным именем или обычным IP. Порт — уникальный номер, с которым связан определённый сокет, проще говоря, его занимает определённая служба для того что бы по нему могли связаться с ней.

Maven — инструмент для автоматизации сборки проектов. С ним работают в основном Java-разработчики. Собрать на Java проект уровня «Hello, world!» можно и с помощью командной строки. Но чем сложнее разрабатываемое ПО и чем больше оно использует сторонних библиотек и ресурсов, тем сложнее будет команда для сборки. Maven разработан для облегчения этой работы.

Одна из главных особенностей фреймворка — декларативное описание проекта. Это значит, что разработчику не нужно уделять внимание каждому аспекту сборки — все необходимые параметры настроены по умолчанию. Изменения нужно вносить лишь в том объёме, в котором программист хочет отклониться от стандартных настроек.

Ещё одно достоинство проекта — гибкое управление зависимостями. Maven умеет подгружать в свой локальный репозиторий сторонние библиотеки, выбирать необходимую версию пакета, обрабатывать транзитивные зависимости.

Для отображение данных на web-сервисе используются jsp страницы.

JSP (Java server pages) — это технология, которая позволяет внедрять Java-код в статичное содержимое страницы, и позволяющая веб-разработчикам динамически генерировать HTML, XML и другие веб-страницы. JSP является составной частью единой технологии создания бизнес-приложений J2EE.

JSP может выполняться практически на любом веб сервере, так как эта технология основана на языке программирования Java, а он, как уже было

сказано выше, не зависит от платформы. Java server pages поддерживает теги HTML, Javascript, а также собственный набор тегов.

ORM (англ. Object-Relational Mapping, рус. *объектно-реляционное отображение*, или преобразование) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Задача - необходимо обеспечить работу с данными в терминах классов, а не таблиц данных и напротив, преобразовать термины и данные классов в данные, пригодные для хранения в СУБД. Необходимо также обеспечить интерфейс для CRUD-операций над данными. В общем, необходимо избавиться от необходимости писать SQL-код для взаимодействия в СУБД.

Java Database Connectivity(JDBC) – это стандартный API для независимого соединения языка программирования Java с различными базами данных (далее – БД).

JDBC решает следующие задачи:

- Создание соединения с БД.
- Создание SQL выражений.
- Выполнение SQL – запросов.
- Просмотр и модификация полученных записей.

Если говорить в целом, то JDBC – это библиотека, которая обеспечивает целый набор интерфейсов для доступа к различным БД.

Для доступа к каждой конкретной БД необходим специальный JDBC – драйвер, который является адаптером Java – приложения к БД.

Hibernate — самая популярная реализация спецификации JPA, предназначенная для решения задач объектно-реляционного отображения (ORM). Целью Hibernate является освобождение разработчика от значительного объёма сравнительно низкоуровневого программирования при работе в объектно-ориентированных средствах в реляционной базе данных. Разработчик может использовать Hibernate как в процессе проектирования системы классов и таблиц «с нуля», так и для работы с уже существующей базой данных.

Библиотека не только решает задачу связи классов Java с таблицами базы данных (и типов данных Java с типами данных SQL), но и также предоставляет средства для автоматической генерации и обновления набора таблиц, построения запросов и обработки полученных данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание SQL- и JDBC-кода. Hibernate автоматизирует генерацию SQL-запросов и освобождает разработчика от ручной обработки

результирующего набора данных и преобразования объектов, максимально облегчая перенос (портирование) приложения на любые базы данных SQL.

Mapping (сопоставление, проецирование) Java-классов с таблицами базы данных осуществляется с помощью конфигурационных XML-файлов или Java-аннотаций. При использовании файла XML Hibernate может генерировать скелет исходного кода для классов длительного хранения. В этом нет необходимости, если используется аннотация. Hibernate может использовать файл XML или аннотации для поддержки схемы базы данных.

Обеспечиваются возможности по организации отношения между классами «один-ко-многим» и «многие-ко-многим». В дополнение к управлению связями между объектами Hibernate также может управлять рефлексивными отношениями, где объект имеет связь «один-ко-многим» с другими экземплярами своего собственного типа данных.

Коллекции объектов данных, как правило, хранятся в виде коллекций Java-объектов, таких, как набор (Set) и список (List). Связанные объекты могут быть настроены на *каскадные* операции. Например, родительский класс Album (музыкальный альбом) может быть настроен на каскадное сохранение и/или удаление своего потомка Track. Это может сократить время разработки и обеспечить целостность. Функция проверки изменения данных (*dirty checking*) позволяет избежать ненужной записи действий в базу данных, выполняя SQL-обновление только при изменении полей персистентных объектов.

Для построения модели IDEF0 команда использует программу CA AllFusion Process Modeler r7(BPWin). Этот продукт предоставляет простой интерфейс для пользователей, тем самым это позволяет сконцентрироваться на теме без потерь времени на разбор инструментальных средств. Интерактивное выделение объектов обеспечивает постоянную визуальную обратную связь при построении модели. BPwin поддерживает ссылочную целостность, не допуская определения некорректных связей и гарантируя непротиворечивость отношений между объектами при моделировании, позволяет эффективно манипулировать моделями — сливать и расщеплять их, а также имеет широкий набор средств документирования моделей, проектовнаша команда разработала помимо IDEF0 следующие диаграммы: развертывания, последовательностей, компонентов.

5.2 Диаграмма последовательности

Диаграмма последовательности — диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) ИС в рамках какого-либо определённого прецедента (отправка запросов и получение ответов). Используется в языке UML.

Основными элементами диаграммы последовательности являются объекты, которые непосредственно участвуют во взаимодействии, но не используются для отображения возможных статических ассоциаций с другими объектами.

Объекты располагаются с лева на права таким образом, чтобы крайним с лева был тот объект, который инициирует взаимодействие. Неотъемлемой частью объекта на диаграмме последовательности является линия жизни объекта. Линия жизни показывает время, в течение которого объект существует в Системе. Периоды активности объекта в момент взаимодействия показываются с помощью фокуса управления. Временная шкала на диаграмме направлена сверху вниз.

Перед тем, как произойдет вход пользователя в систему будет совершен ряд определенных действий. Сначала пользователь введет данные, после чего контроллер определит, какое действие совершил пользователь. Далее будет формироваться запрос на языке SQL пока к связующему звену, виртуальной базе данных, между реальной базой и концепцией программирования. Запрос будет отправлен в базу данных, где он будет выполнен.

После выполнения запроса, его результаты будут поэтопно возвращены, сначала на Object Relational Mapping, затем переданы контроллеру, который должен их обработать. После завершения обработки контроллер отправляет результаты на страницу сервиса, и они отображаются пользователю.

Данный вид диаграммы отображен на рисунке 5.1.

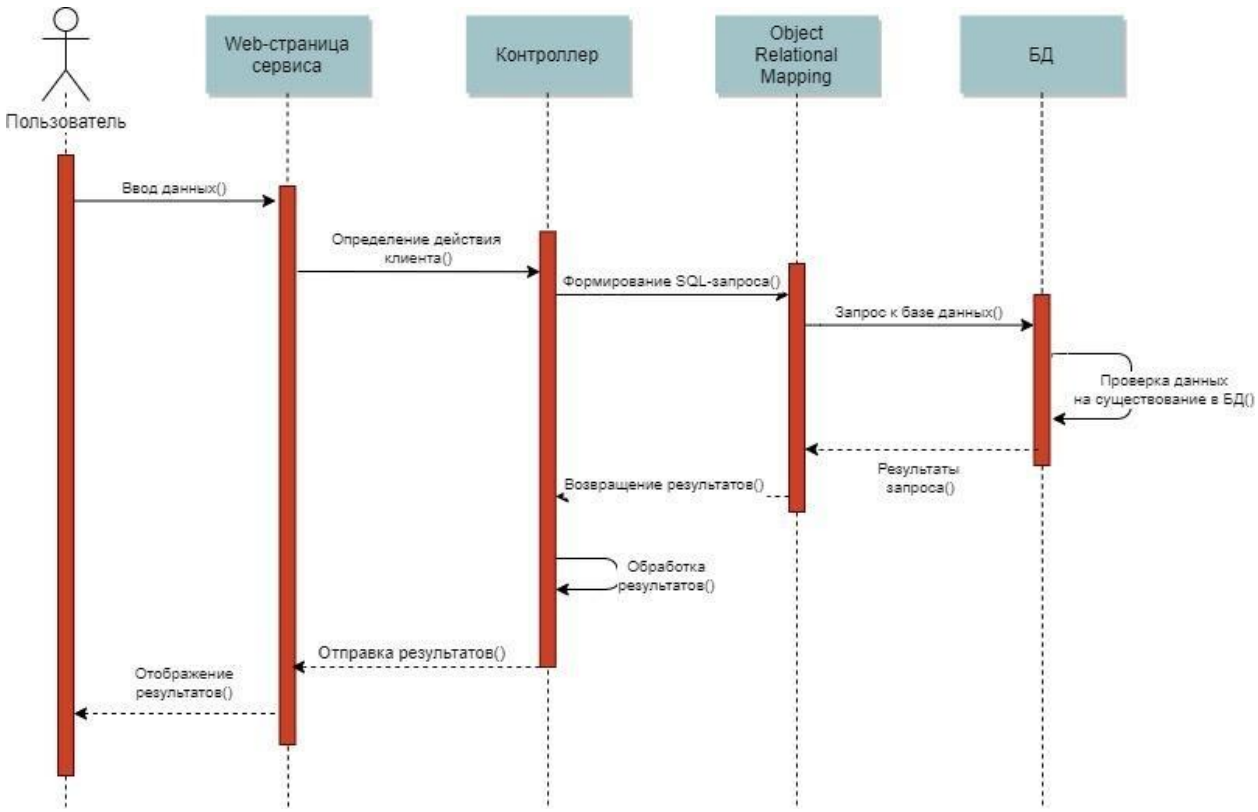

Рисунок 5.1 – Диаграмма последовательности входа пользователя в систему

5.3 Диаграмма развёртывания сервиса

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации.

На диаграмме развертывания отражены узлы, на которых происходит выполнение системы, а также компоненты, в них размещенные.

Диаграмма состояний отражена на рисунке 5.2.

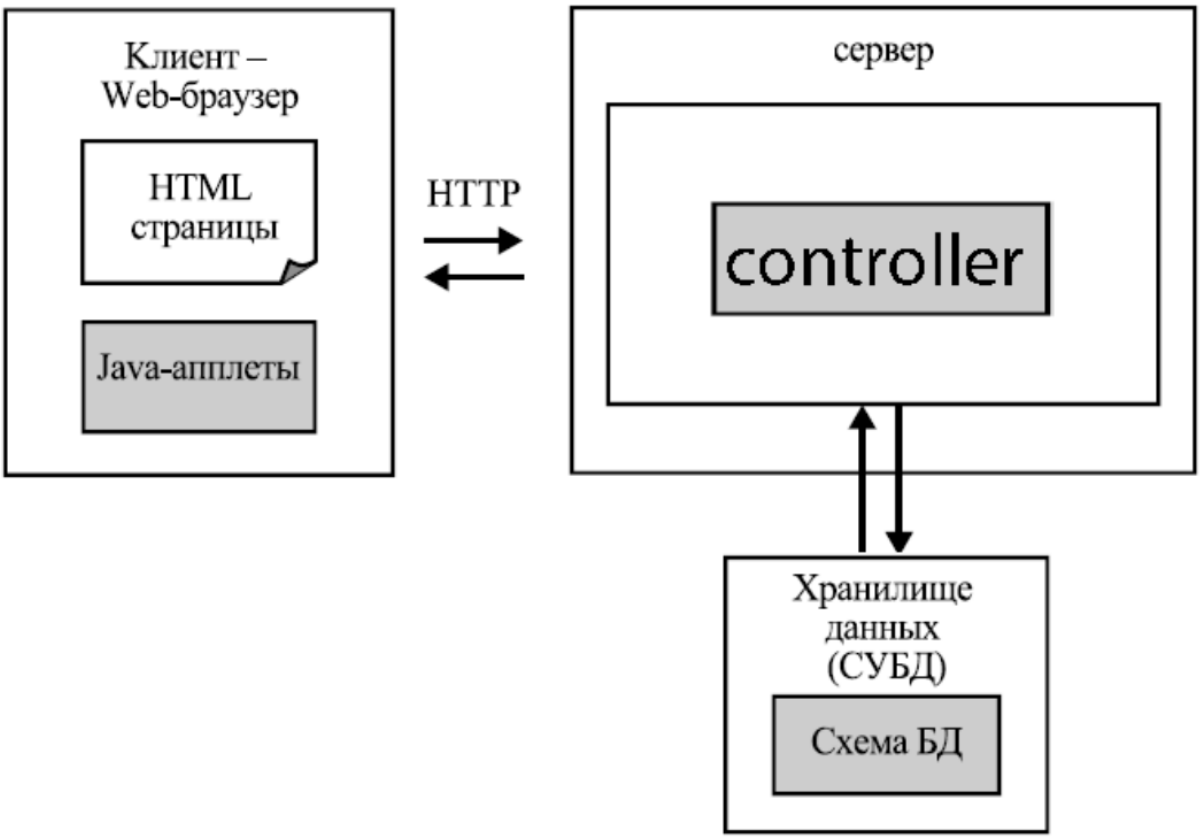

Рисунок 5.2 – Диаграмма развёртывания сервиса заказа и доставки еды

5.4 Диаграмма компонентов системы

Диаграмма компонентов – элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п. С помощью диаграммы компонентов представляются инкапсулированные классы вместе с их интерфейсными оболочками, портами и внутренними структурами (которые тоже могут состоять из компонентов и коннекторов).

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами.

Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Диаграмма компонентов изображена на рисунке 5.3.

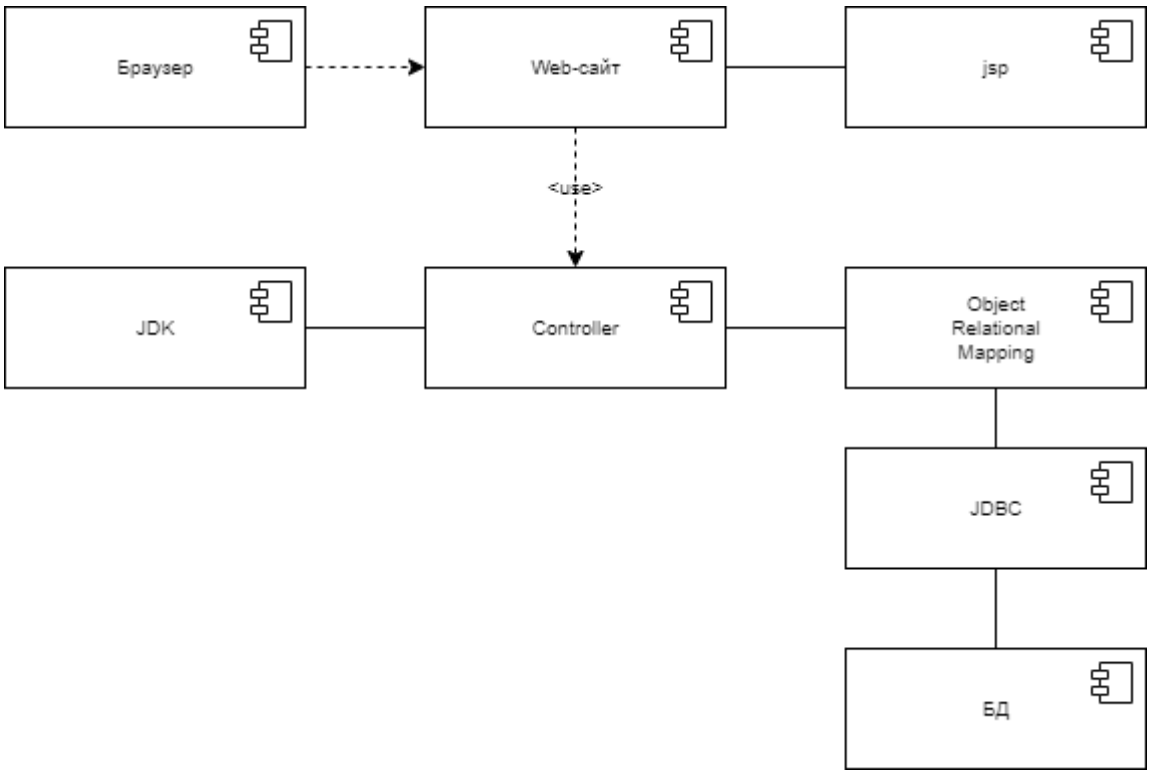

Рисунок 5.3 – Диаграмма компонентов сервиса заказа и доставки еды

С помощью диаграммы компонентов представляются инкапсулированные классы вместе с их интерфейсными оболочками, портами и внутренними структурами (которые тоже могут состоять из компонентов и коннекторов).

5.5 Диаграмма классов

Диаграмма классов является структурной диаграммой языка моделирования UML, которая демонстрирует общую структуру иерархии классов системы, их кооперации, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.
Основными элементами являются классы и связи между ними. Классы характеризуются при помощи атрибутов и операций.

Атрибуты описывают свойства объектов класса. Большинство объектов в классе получают свою индивидуальность из-за различий в их атрибутах и взаимосвязи с другими объектами. Однако, возможны объекты с идентичными

значениями атрибутов и взаимосвязей. Т.е. индивидуальность объектов определяется самим фактом их существования, а не различиями в их свойствах. Имя атрибута должно быть уникально в пределах класса. За именем атрибута может следовать его тип и значение по умолчанию. Операция есть функция или преобразование. Операция может иметь параметры и возвращать значения. Диаграмма классов отражена на рисунках А.1.-А.7 Приложения А.

6 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ

6.1 Диаграмма состояний

Диаграмма состояний покажет нам все возможные состояния, в которых может находиться объект, а также процесс смены состояний в результате внешнего влияния. Диаграмма состояний отражена на рисунке 6.1. Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

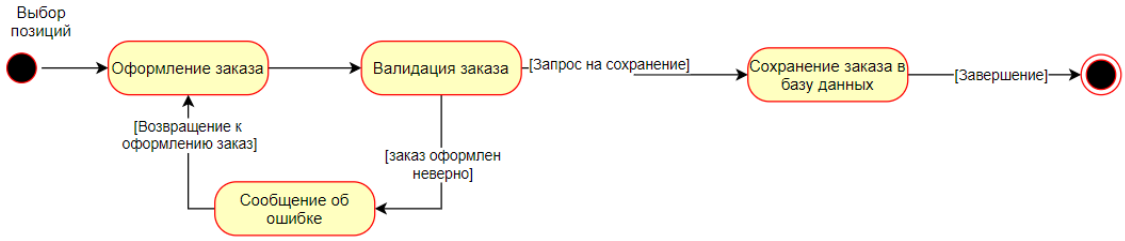

Рисунок 6.1 – Диаграмма состояний во время регистрации нового пользователя

6.2 Алгоритм авторизации

Блок-схема алгоритма авторизации представлена на рисунке 6.2.

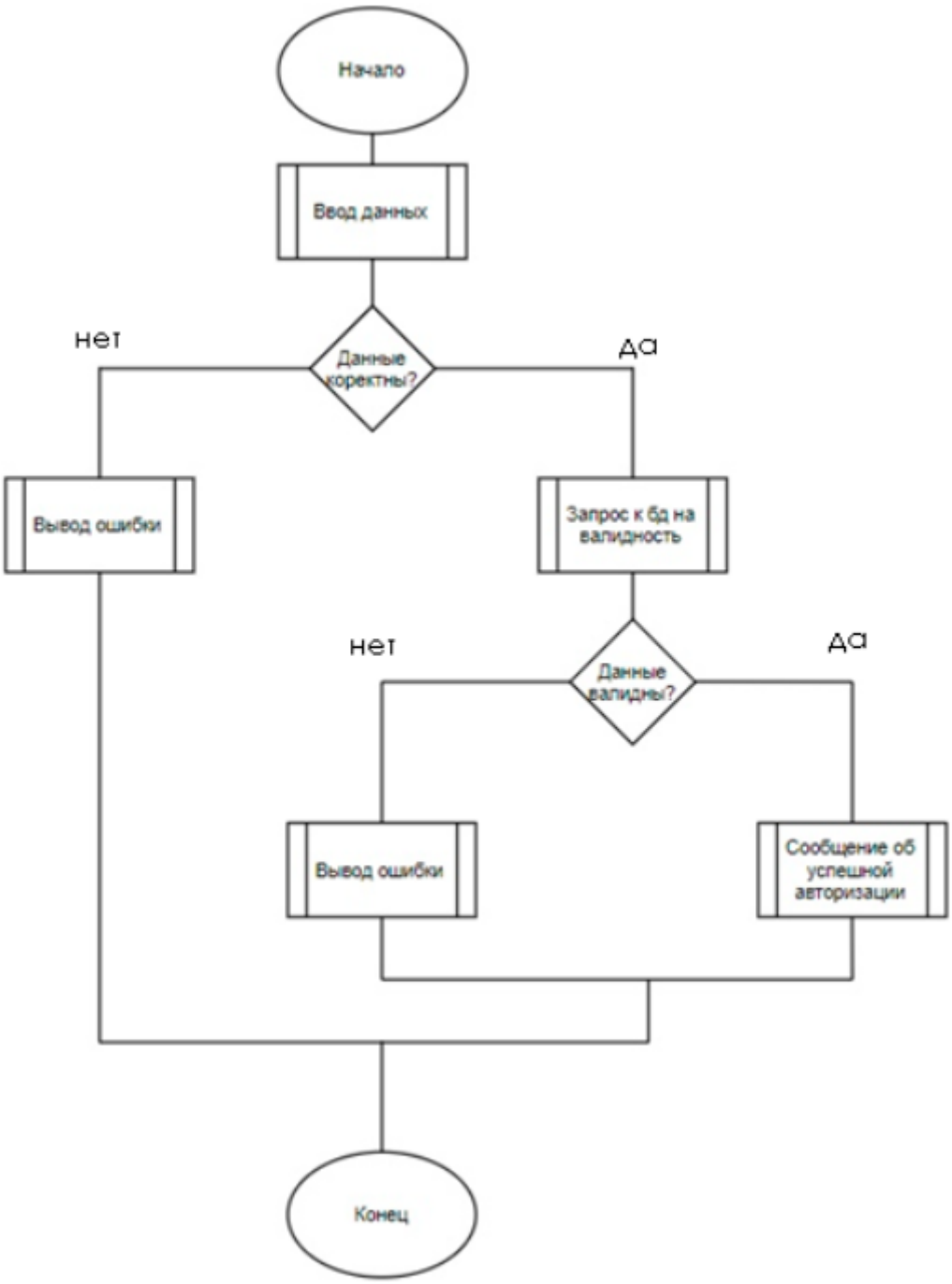

Рисунок 6.2 – Блок-схема алгоритма авторизации

Первым делом пользователь вводит свои личные данные, необходимые для авторизации аккаунта. После чего если данные корректны выполняется запрос к базе данных для проверки на соответствие. Если все этапы были успешно завершены, то появится сообщение об успешной авторизации. Иначе появится сообщение об ошибке.

6.3 Алгоритм заказа еды

Блок-схема алгоритма заказа еды представлена на рисунке 6.3.

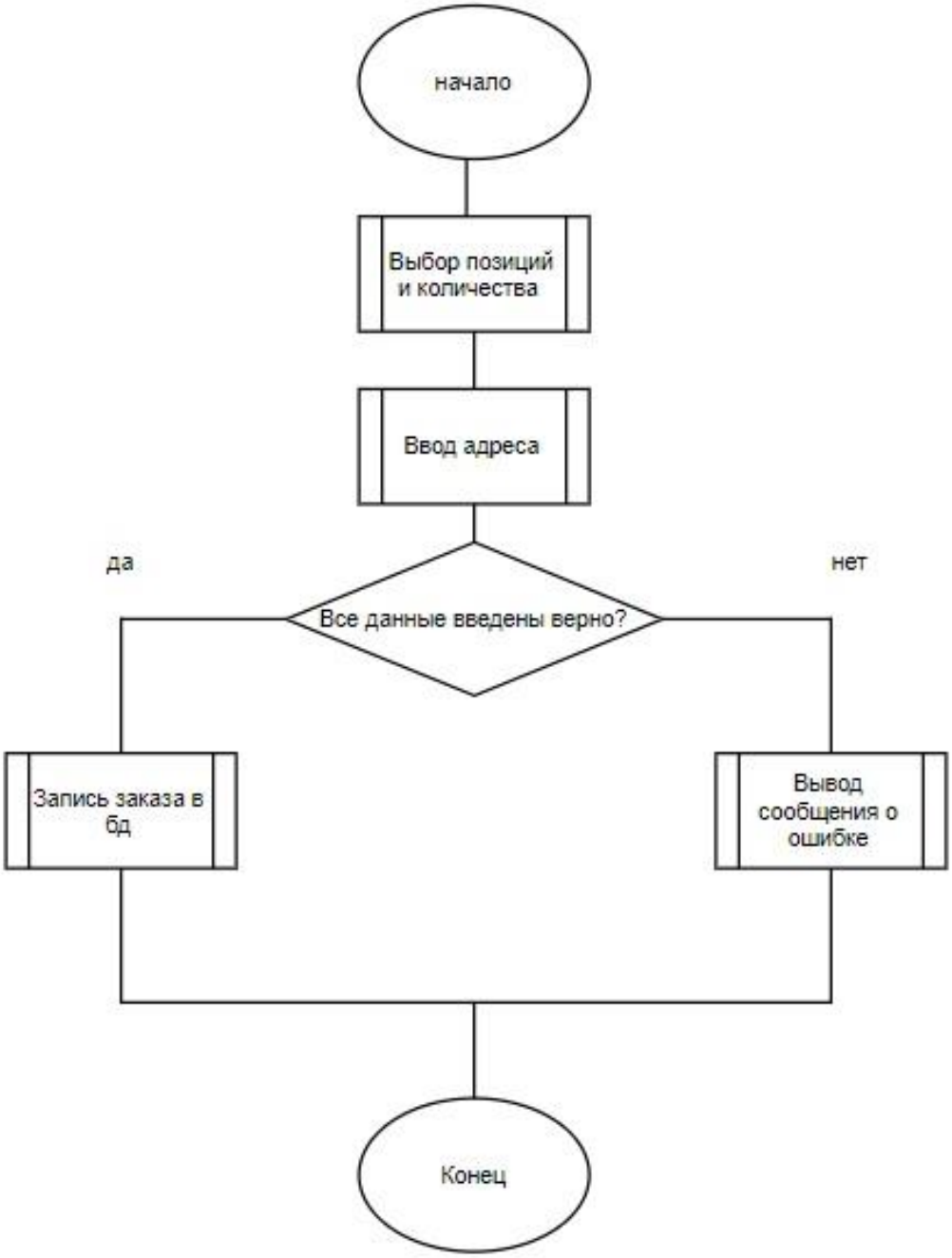

Рисунок 6.3 – Блок-схема алгоритма закза еды

Сразу посетитель должен выбрать предпочтительную продукцию и её

количество. Затем ввести полный адрес, на который нужно доставить заказ. Если данные введены верно, то заказ оформится и запишится в базу данных, а если неверно, то появится сообщение об ошибке.

7 ОПИСАНИЕ ПРИМЕНЕНИЯ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ

7.1 MVC

Термин модель-представление-контроллер был задуман как способ организации некоторых из ранних приложений графического пользовательского интерфейса.

Если оперировать понятиями высокого уровня, архитектурный шаблон MVC означает, что приложение MVC будет разделено, по крайней мере, на три части (рисунок 7.3.1).

Модели содержат или представляют данные, с которыми работают пользователи. Они могут быть простыми моделями представлений, которые только представляют данные, передаваемые между представлениями и контроллерами; или же они могут быть моделями предметной области, которые содержат бизнес-данные, а также операции, преобразования и правила для манипулирования этими данными.

Представления применяются для визуализации некоторой части модели в виде пользовательского интерфейса.

Контроллеры обрабатывают поступающие запросы, выполняют операции с моделью и выбирают представления для визуализации пользователю.

Основная цель применения этой концепции состоит в отделении бизнес-логики от её визуализации. За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения.

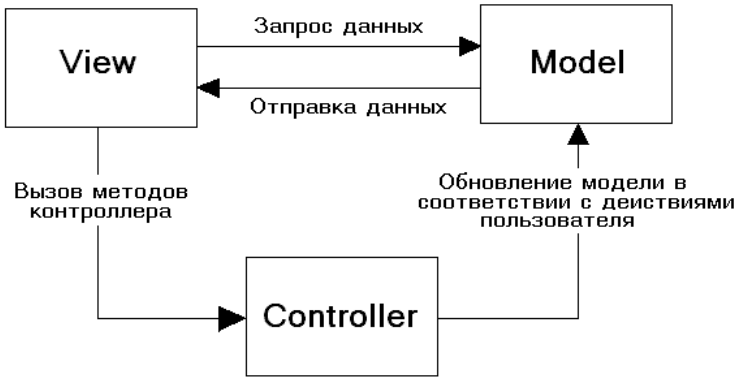

Рисунок 7.3.1 – Паттерн «MVC»

Плюсами использования данного паттрна будут так же:

− удобство выводить разные представления (view) для разных типов устройств, при этом пользуясь одними и теми же данными;

− облегчается поддержка и тестирование кода.

Таким образом, данный паттерн полностью описывает работу работы.

7.2 Фабричный метод

Фабричный метод — это порождающий паттерн проектирования, который определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.

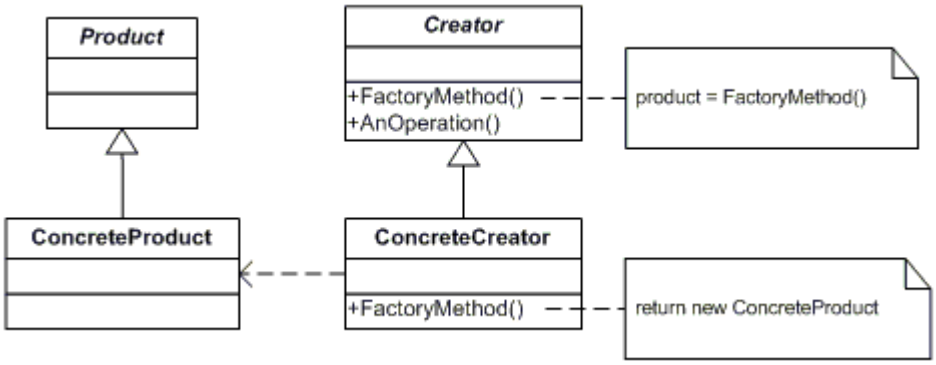

Рисунок 7.3.2. – Паттерн «фабричный метод»

Определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать. Фабричный метод позволяет классу делегировать создание подклассов. Используется, когда:

- классу заранее неизвестно, объекты каких подклассов ему нужно создавать.
- класс спроектирован так, чтобы объекты, которые он создаёт, специфицировались подклассами.
- класс делегирует свои обязанности одному из нескольких вспомогательных подклассов, и планируется локализовать знание о том, какой класс принимает эти обязанности на себя

7.2 Декоратор

Декоратор — структурный шаблон проектирования, предназначенный для динамического подключения дополнительного поведения к объекту. Шаблон Декоратор предоставляет гибкую альтернативу практике создания подклассов с целью расширения функциональности.

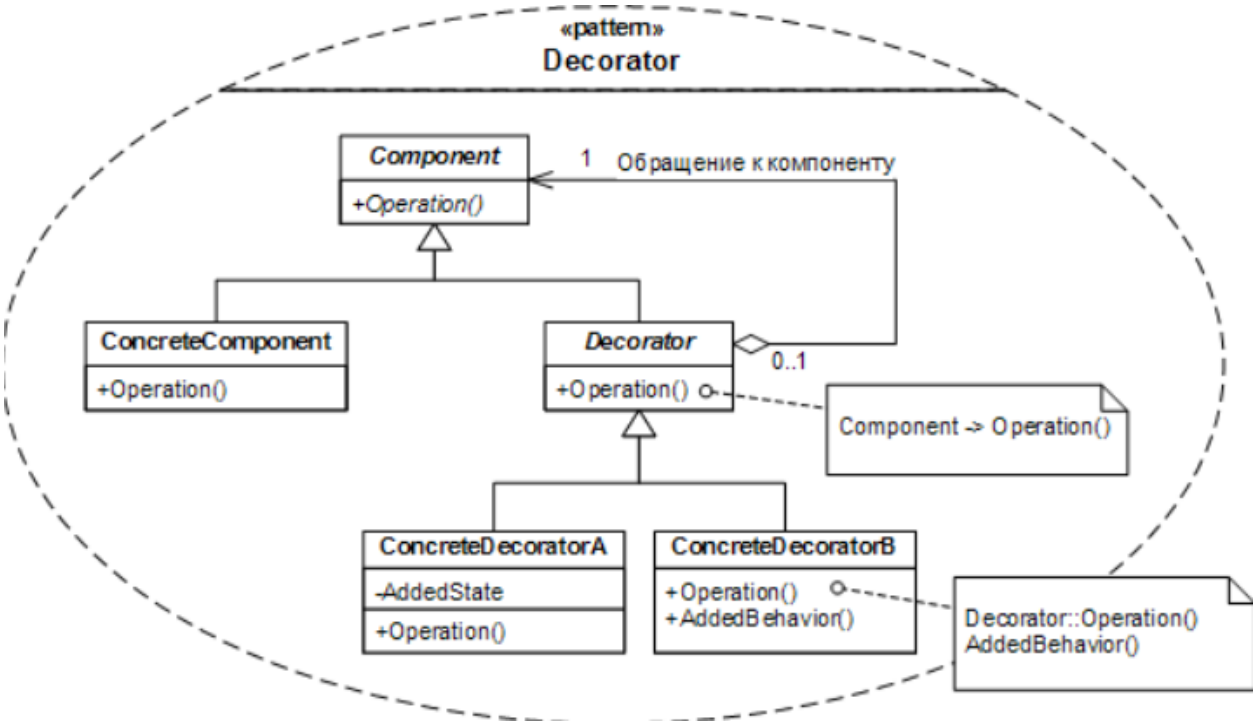

Рисунок 7.3.3. – Паттерн «Декоратор»

Объект, который предполагается использовать, выполняет основные функции. Однако может потребоваться добавить к нему некоторую дополнительную функциональность, которая будет выполняться до, после или даже вместо основной функциональности объекта.

8 РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ СИСТЕМЫ

Для развертывания системы сервиса для заказа еды необходимо иметь java 8.

Для хранения данных была использована СУБД PostgreSQL. Поэтому рекомендуется иметь ее в установленном виде на компьютере.

Генерация таблиц происходит после вставки в СУБД скрипта tables.sql. Вставка значений благодаря скрипту insert.sql.

Проект содержит фреймворк сборки Maven, поэтому все использовавшиеся зависимости будут подгружены сами.

Далее запускаем приложение: вводим в терминал npm start.

Таким образом, для корректной работы и обеспечения полного функционирования системы необходимы такие инструменты, как Java 8+, СУБД PostgreSQL, Apache Maven.

9 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗБОТАННОГО СЕРВИСА ЗАКАЗА И ДОСТАВКИ ЕДЫ

Когда пользователь заходит на сайт, появляется главная страница (Рисунок 9.1).

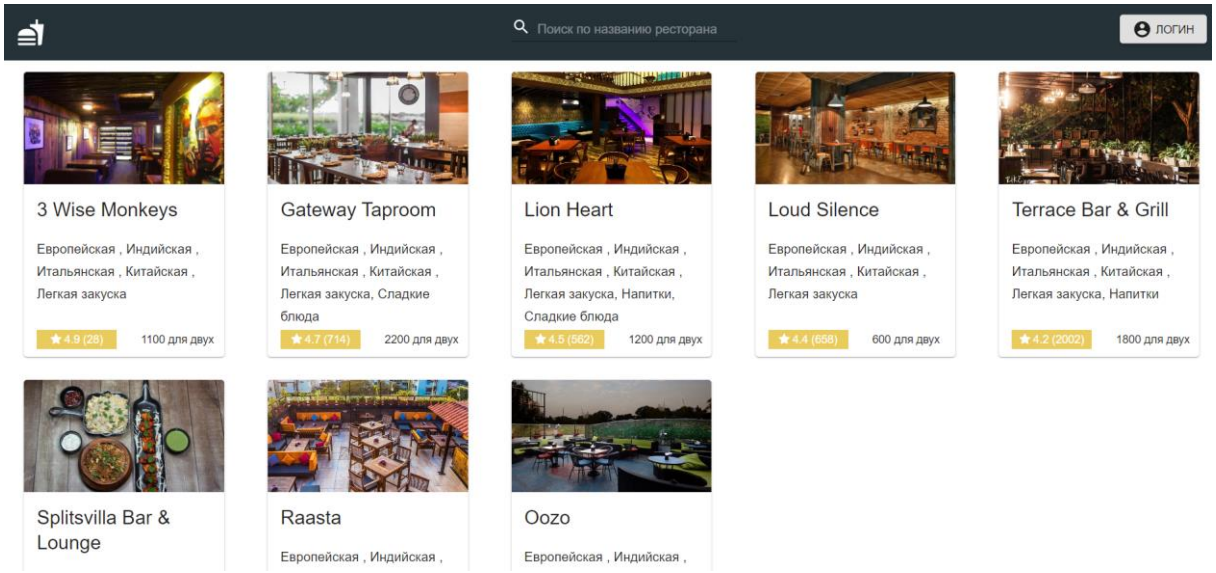

Рисунок 9.1 – Главная страница сайта

Для входа в верхнем правом углу есть кнопка логин нажав на которую мы попадем на страничку авторизации (рисунок 9.2).

При некорректном вводе пользователь увидит ошибки (рисунок 9.3).

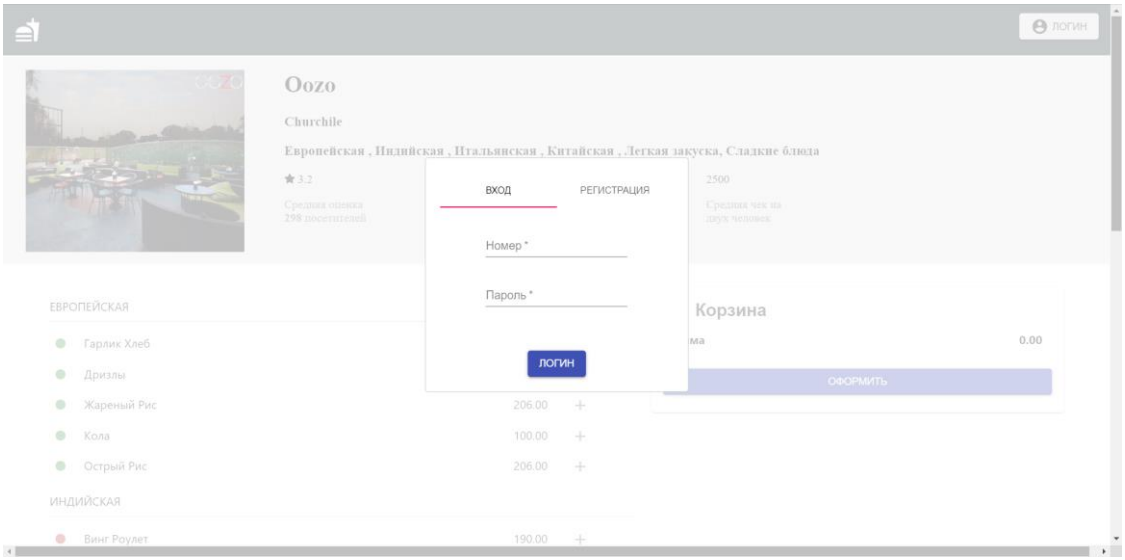

Рисунок 9.2 – Страница авторизации

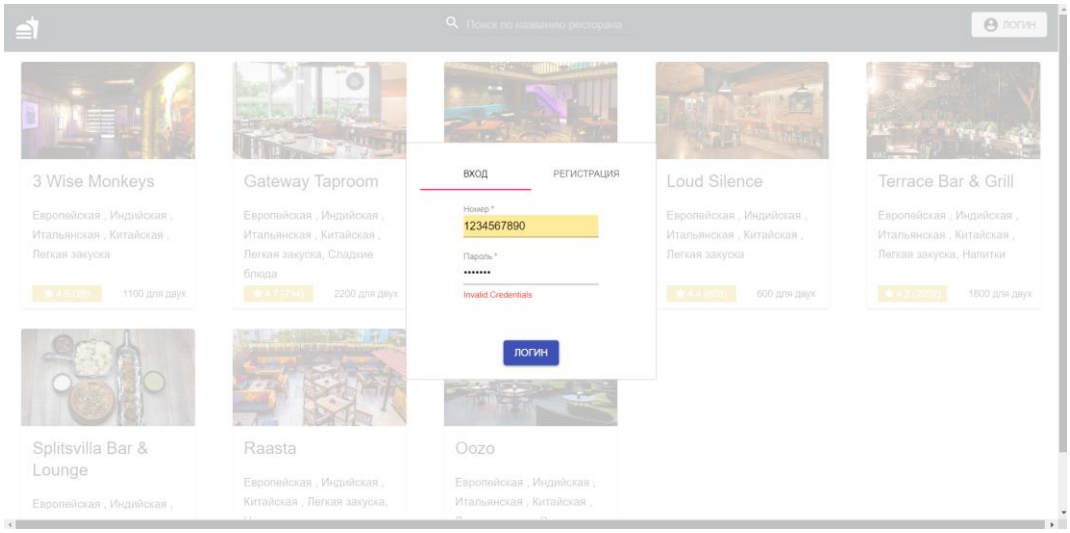

Рисунок 9.3 – Ошибка при авторизации

Если пользователь не зарегистрирован на сайте, ему будет предложено это сделать (рисунок 9.4). Если пользователь будет вводить невалидные данные или вообще не будет вводить их, то он увидит сообщение об ошибке (рисунок 9.5).

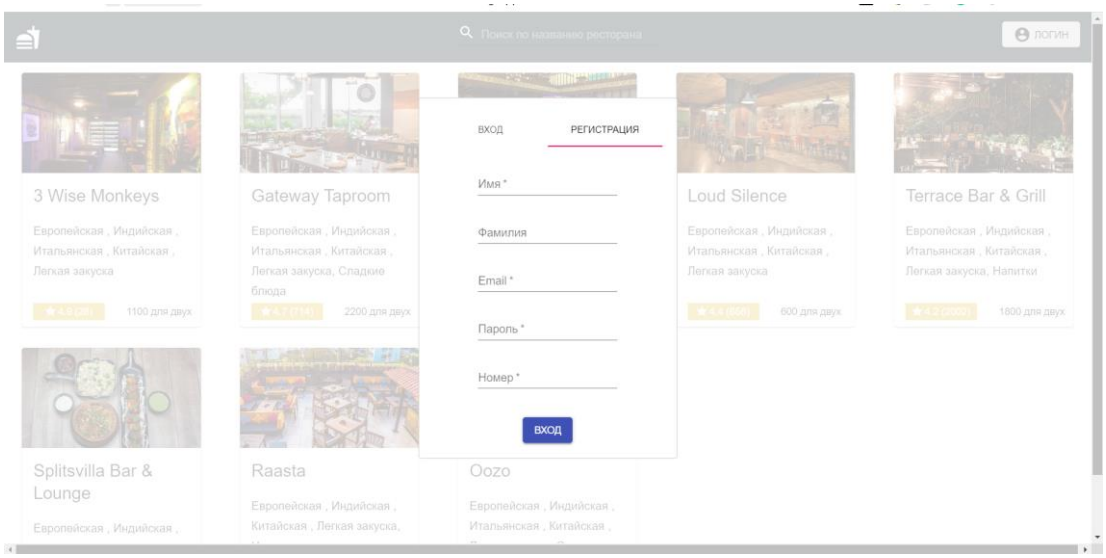

Рисунок 9.4 – Страница регистрации

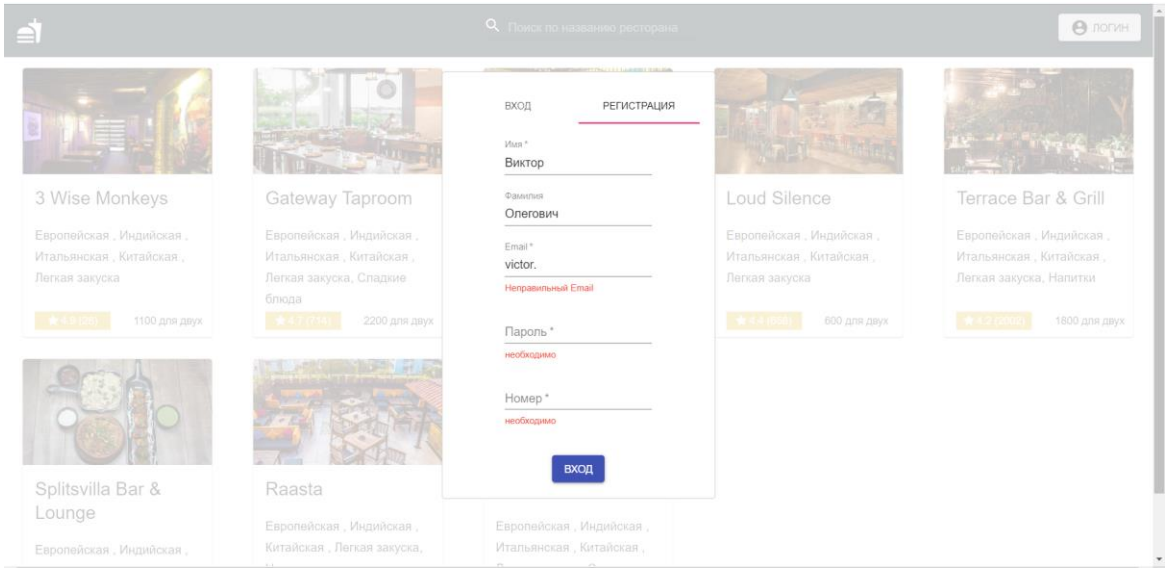

Рисунок 9.5 – Ошибка при регистрации

После успешного входа в нижнем левом углу появится надпись об входе (рисунок 9.6).

На главное странице пользователь сможет выбрать ресторан, основываясь на свои предпочтения. После выбора одного из ресторанов клиент попадает на страницу ресторана, где представлен весь ассортимент данного заведения, разбитый на категории (рисунок 9.7).

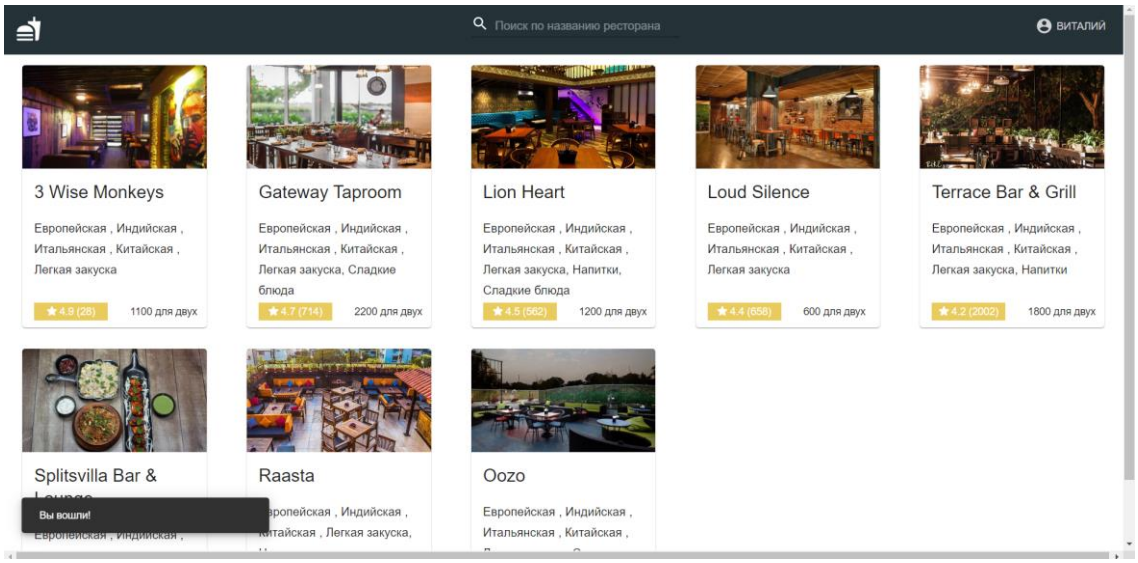

Рисунок 9.6 – Успешный вход

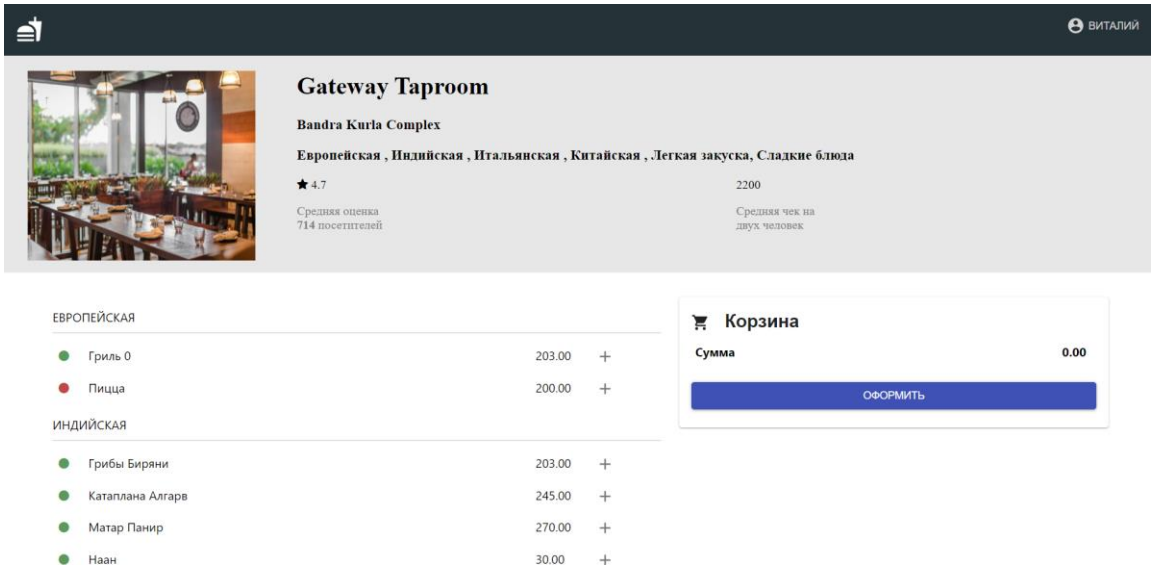

Рисунок 9.7 – Страница одного из ресторанов

Здесь пользователь может выбрать продукцию на свой вкус и добавить её в заказ. Количество товара можно регулировать как в корзине, так и в самом ассортименте (рисунок 9.8). Если пользователь не авторизовался или не добавил товар в корзину, то при нажатии кнопки оформить он увидит ошибку (рисунок 9.9).

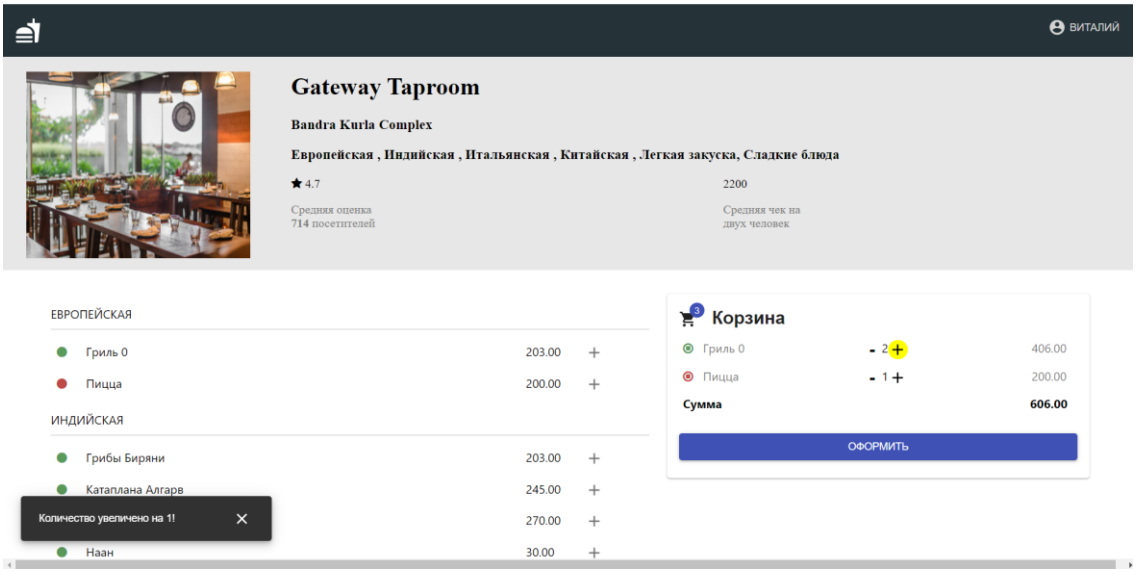

Рисунок 9.8 – Добавление товаров в корзину

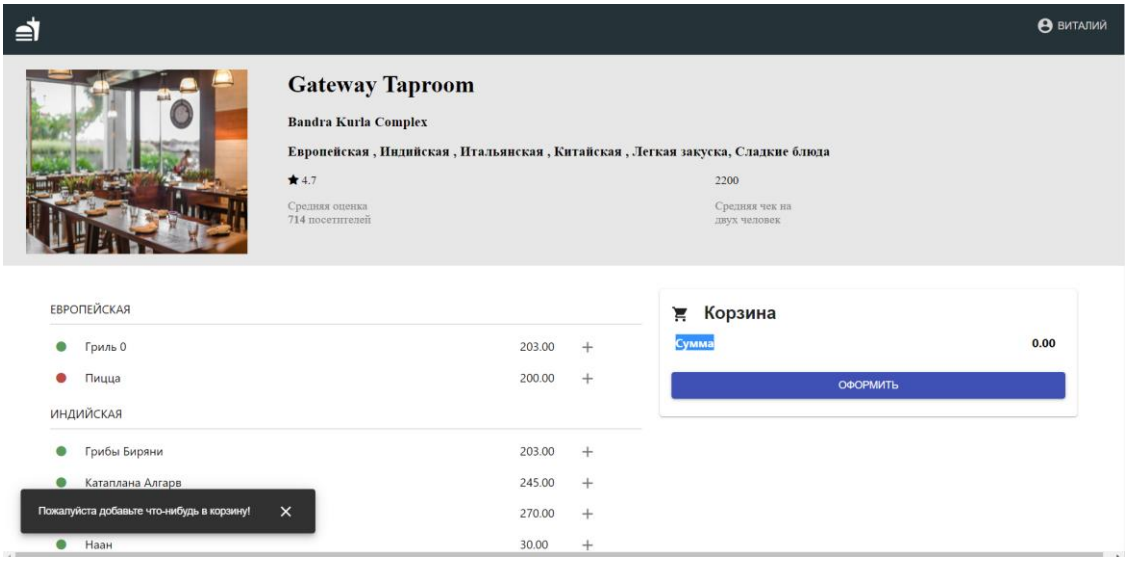

Рисунок 9.9 – Ошибка при оформлении

После нажатия на кнопку оформить пользователь попадет на страницу оформления заказа (рисунок 9.10), на которой он должен будет указать адрес и способ оплаты. Здесь пользователь может указать ранее добавленный адрес или добавить новый (рисунок 9.11). При добавлении адреса ттоже присутствует валидация (рисунок 9.12).

Рисунок 9.10 – Страница оформления заказа

Рисунок 9.11 – Страница добавления нового адреса

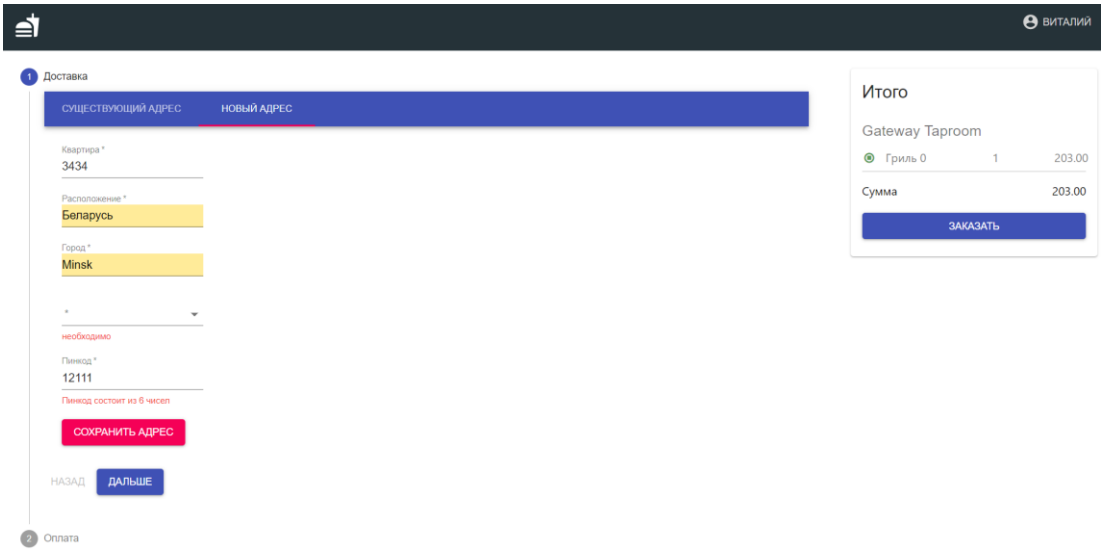

Рисунок 9.12 – Ошибка при добавлении нового адреса

После добавлении нового адреса он отобразится на странице офрмления заказа (рисунок 9.13). После того как пользователь определился с адресом, ему нужно выбрать метод оплаты (рисунок 9.14).

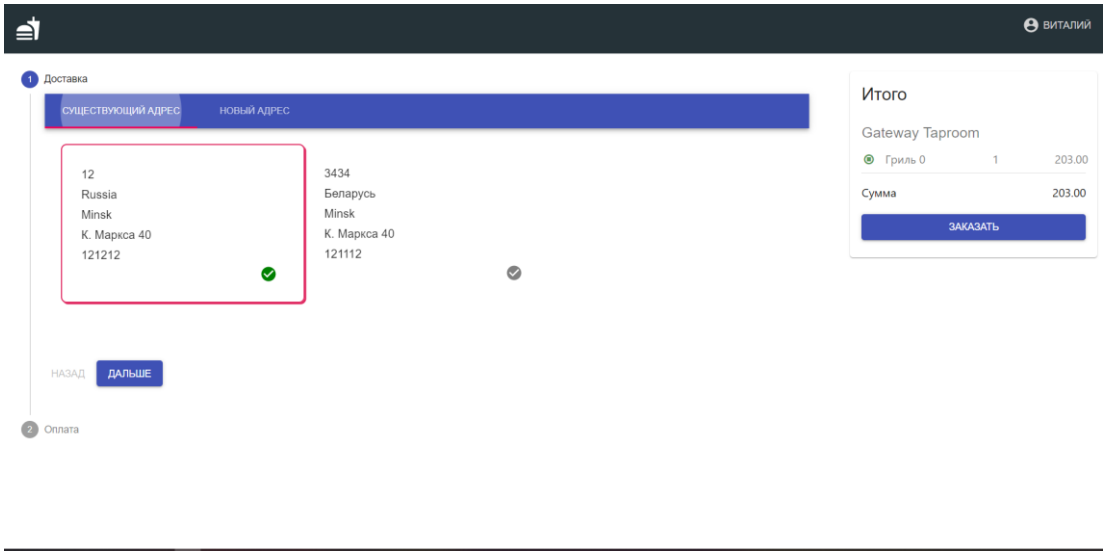

Рисунок 9.13 – Выбор существующего адреса

Рисунок 9.14 – Выбор метода оплаты

Если пользователь хочет внести изменения в заказ, ему следует нажать на кнопку изменить (рисунок 9.15) и он вернется на страницу заказа. После подтверждения заказа в нижнем левом углу экрана появится уведомление (рисунок 9.16).

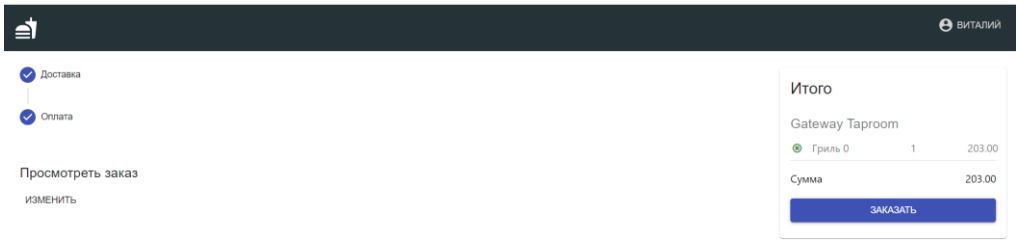

Рисунок 9.15 – Изменение заказа

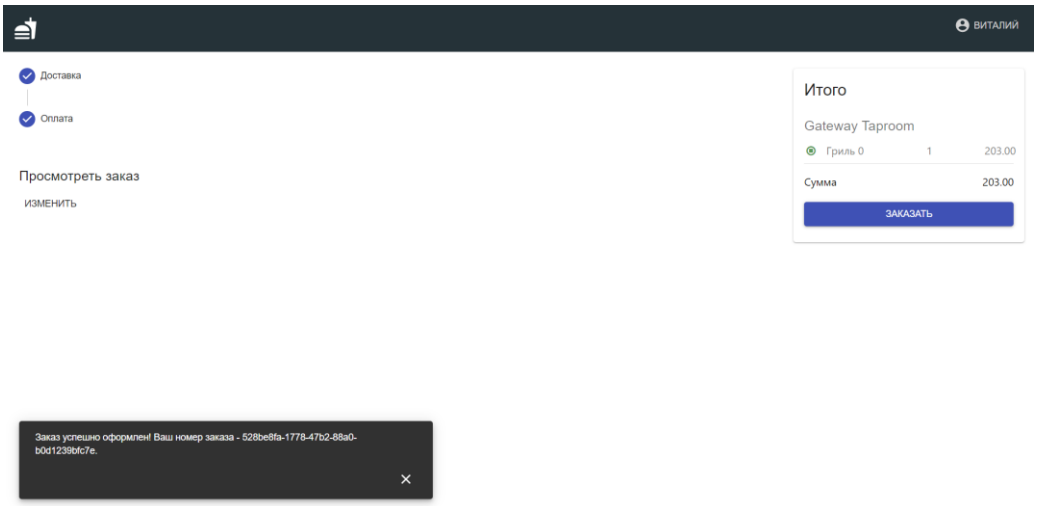

Рисунок 9.16 – Заказ подтвержден

Все заказы можно увидеть в профиле клиента.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта было разработано приложение, которое позволяет оптимизировать работу сервиса заказа и доставки еды, повысить эффективность работы персонала при оформлении заказов. Реализация такого приложения значительно поспособствует оптимизации работы ресторана в целом, поскольку все данные будут храниться в приложении. Любая операция с заказами может быть совершена в течение нескольких минут.

Для правильной разработки проекта необходима тщательная проработка предметной области. Это позволяет правильно и в полной мере реализовать функционал приложения.

В результате разработки курсового проекта цель, поставленная в начале работы, выполнена: программное приложение реализовано.

Рассматривая работу самого приложения, стоит выделить, что система была протестирована, не обнаружив значительных ошибок, получив при этом неплохие результаты производительности. Интерфейс программного приложения прост для понимания и лаконичен, он сделан так, чтобы пользователь смог легко работать с приложением. Данные выводятся в удобном формате.

Предполагается, что данное программное приложение будет использоваться людьми, которые совершают какой-либо заказ.

Была разработана блок-схема программы, которая наглядно отображает её работу.

В дальнейшем в приложение могут быть внесены изменения для усовершенствования производимости, также новые возможности функциональности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Постников М.Л., Шведенко В.В., Шведенко В.Н., Щекочихин О.В. Моделирование адаптивной системы управления промышленным предприятием. Монография. – Кострома: Общество «Знание», 2010 – 168 с.

[2] Котлер Ф. Основы маркетинга. Краткий курс / пер. с англ. – М.: Вильямс, 2016. – 656 с.

[3] Харрингтон Д., Эсселинг К.С., Нимвеген Х.В. Оптимизация бизнес-процессов. Документирование, анализ, управление, оптимизация. Изд.: БМикро, Азбука. 2002 г. – 320 с.

[4] Лонг Д., Бастани К. Java в облаке. Spring Boot, Spring Cloud, Cloud Foundry. – М.: СПб.: Питер, 2018 – 624 с.

[5] Уоллс К. Spring в действии. – М.: ДМК Пресс, 2013. – 752 с.

[6] Шёгин Г. PostgreSQL 11. Мастерство разработки. – М.: ДМК Пресс, 2019 – 352 с.

[7] Постников А.М., Шведенко П.В. Теоретические аспекты построения информационной модели системы управления предприятием. Монография. – СПб.: Питер, 2017. – 245 с.

[8] EFSET Education First [Электронный ресурс]. – Режим доступа: https://www.efset.org/ru/english-score/cefr/.

[9] Редмонд Э. Семь баз данных за семь недель: введение в современные базы данных и идеологию NoSQL / Эрик Редмонд, Джим Р. Уилсон; под ред. Жаклин Картер; пер. А. А. Слинкин. - Москва: ДМК Пресс, 2015. – 383 с.

[10] UML. Классика CS. 2-у изд./Пер. с англ.; Под общей редакцией проф. С.Орлова - СПб.: Питер, 2006. – 736 с.: ил.

[11] Шилдт Г. Java. Полное руководство. Изд.: Вильямс. 2018 г. – 1488 с.

ПРИЛОЖЕНИЕ А
(Диаграмма классов)

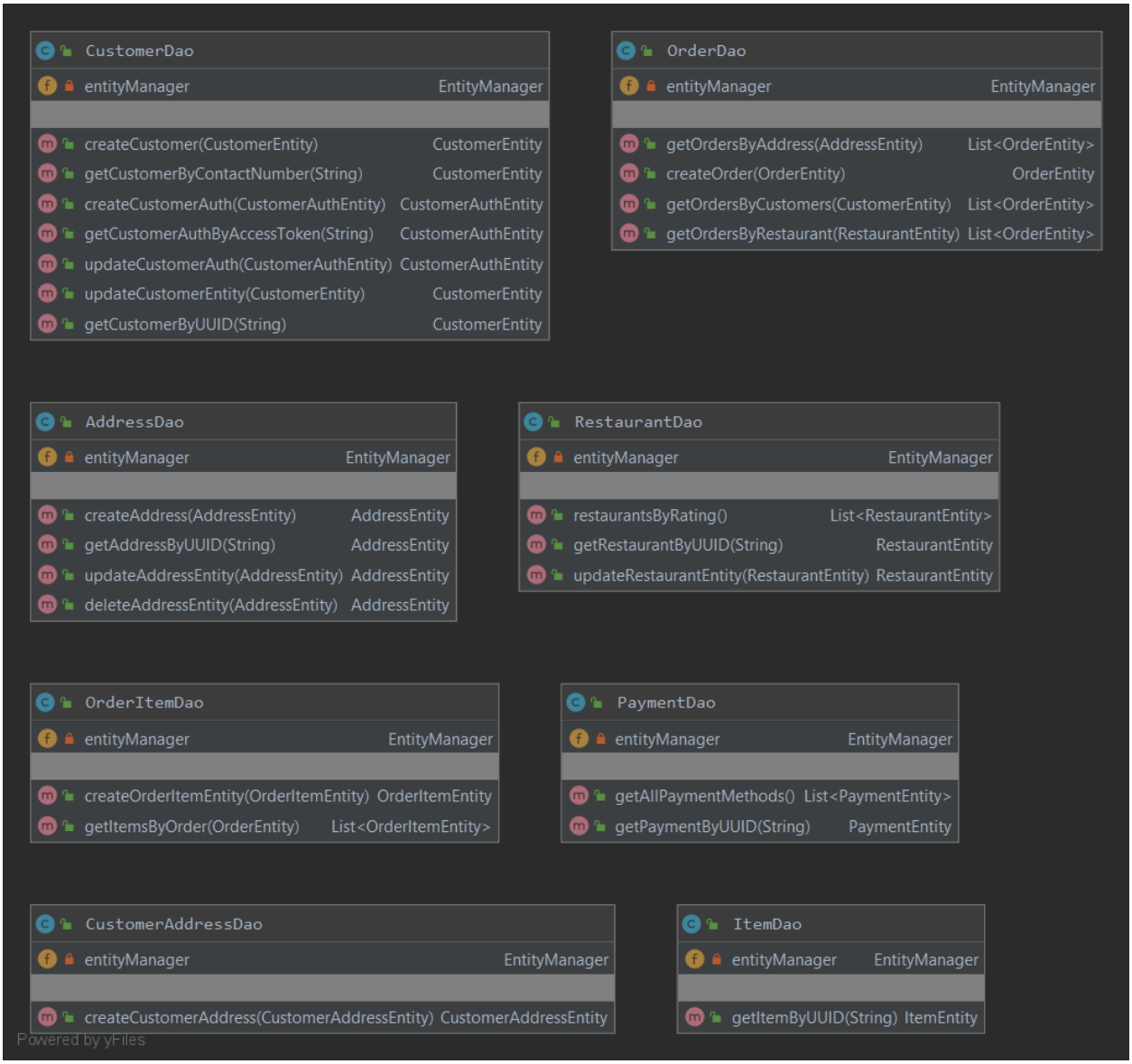

Рисунок А.1 – Диаграмма классов dao

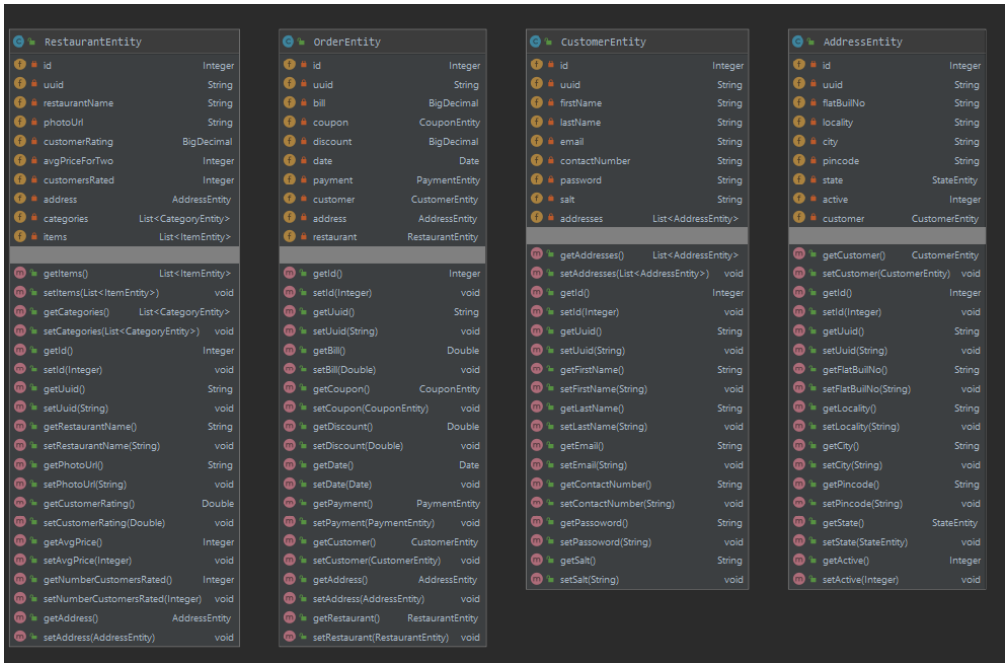

Рисунок А.2 – Диаграмма классов Entity

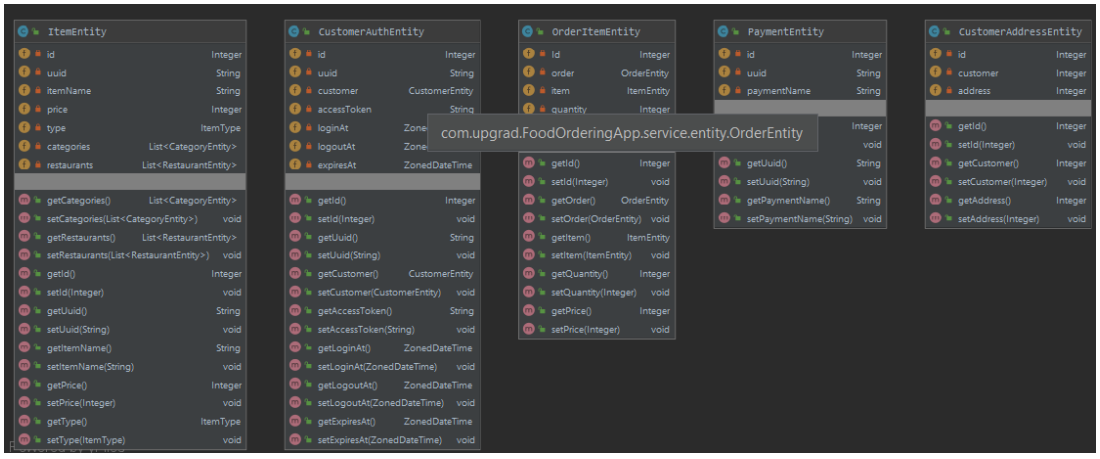

Рисунок А.3 – Продолжение диаграмма классов Entity

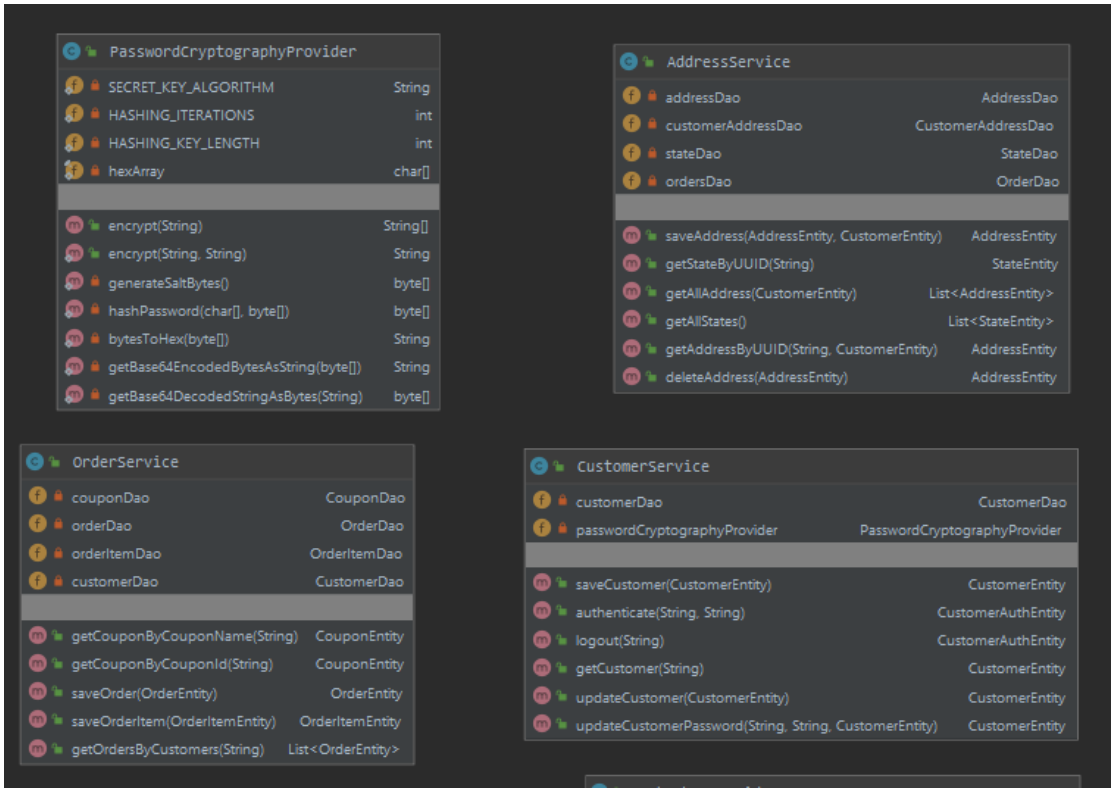

Рисунок А.4 – Диаграмма классов service

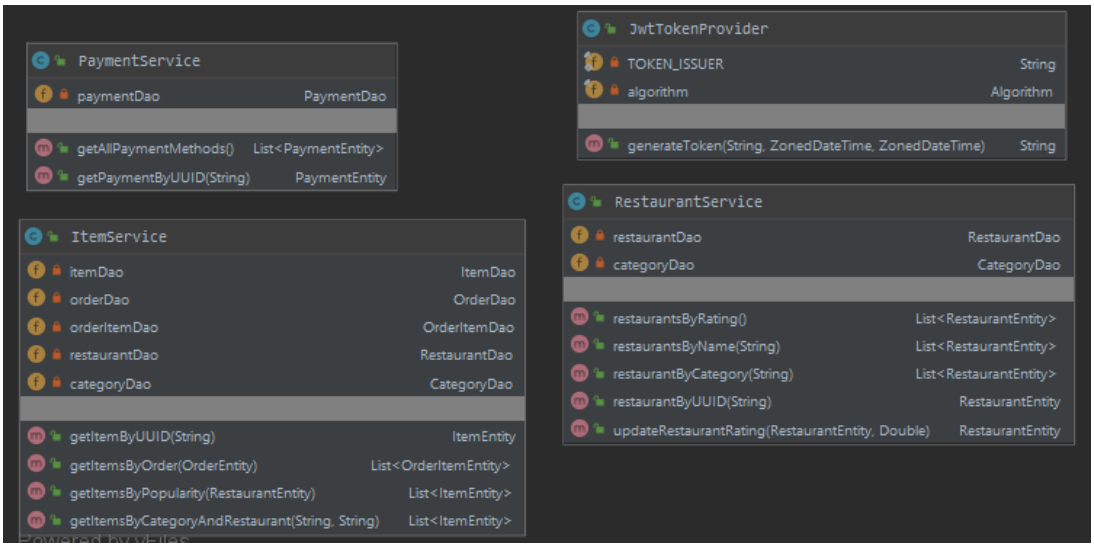

Рисунок А.5 – Продолжение диаграммы классов service

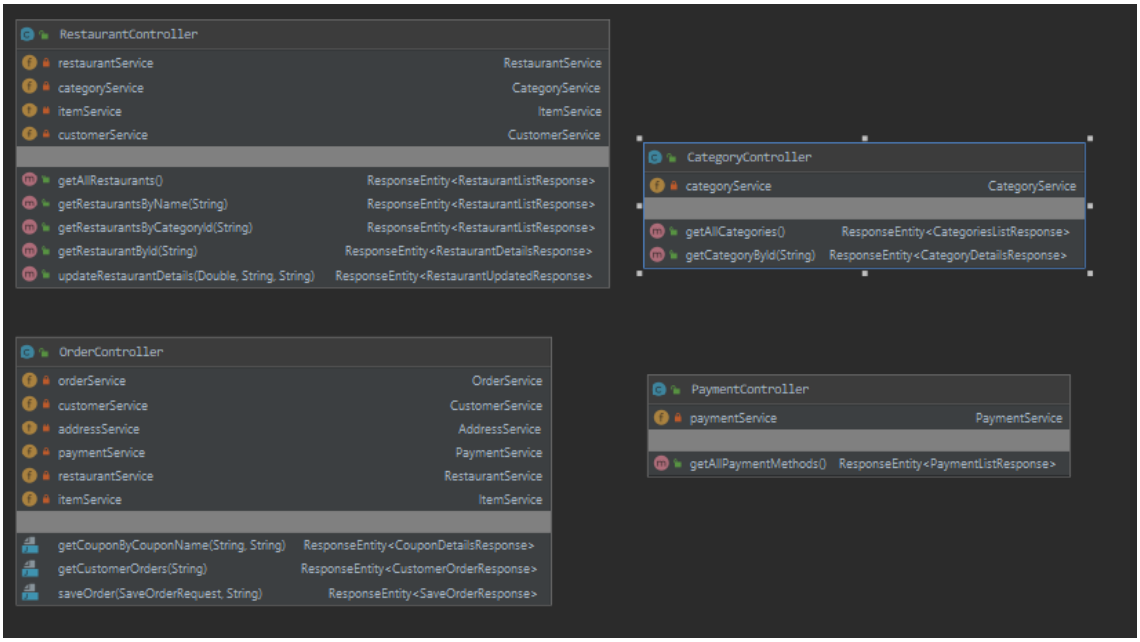

Рисунок А.6 – Диаграммы классов controller

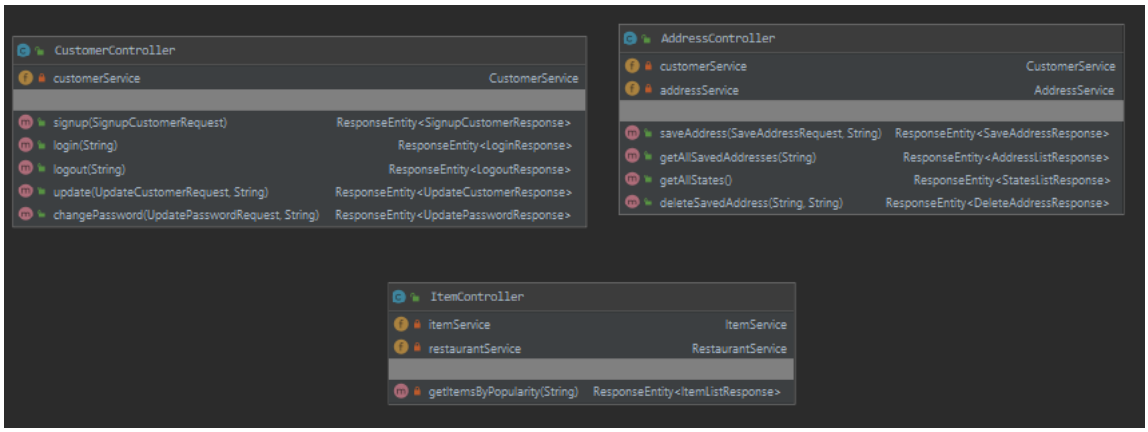

Рисунок А.7 – Продолжение диаграммы классов controller