*Федеральное государственное бюджетное образовательное учреждение*

*высшего профессионального образования*

***«Московский государственный технический университет имени Н.Э. Баумана»***   
***(МГТУ им. Н.Э. Баумана)***

**Платформа для ресторанов**

Глоссарий

|  |  |
| --- | --- |
| **Термин** | **Определение** |
| Валидация данных | Проверка на корректность, полноту и непротиворечивость входных, выходных и обрабатываемых данных |
| WEB-сервис | идентифицируемая веб-адресом программная система со стандартизированными интерфейсами. |
| Авторизация | предоставление определённому лицу или группе лиц прав на выполнение определённых действий |
| Аутентификация | процедура проверки легальности пользователя |
| Обфускация | Приведение исходного кода к такому виду, что код становится трудночитаемым, но полностью сохраняет свою функциональность |
| СОА (SOA) | Сервис-ориентированная архитектура (Service Oriented Architecture), [модульный](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) подход к разработке [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), основанный на использовании [распределённых](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F), [слабо связанных](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%BB%D0%B0%D0%B1%D0%BE%D0%B5_%D1%81%D0%B2%D1%8F%D0%B7%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) заменяемых компонентов, оснащённых стандартизированными [интерфейсами](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для взаимодействия по стандартизированным [протоколам](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB). |
| Регистрация | Процедура создания записи пользователя, необходимая для последующей его аутентификации и авторизации |
| WEB-интерфейс | Интерфейс пользователя, предоставляемой системой через Web-браузер |
|  |  |

Принятые сокращения

|  |  |
| --- | --- |
| **Сокращение** | **Расшифровка** |
| ОС | Операционная система |
| ПО | Программное обеспечение |

Введение

Данное техническое задание составляется для проектирования ПО «Платформа для ресторанов». Техническое задание выполняется в соответствии со стандартом ГОСТ 19.201-78 «ЕСПД. Техническое задание, требования к содержанию и оформлению».

Краткое описание предметной области

У людей в больших городах очень высокий темп жизни. Все свое время они тратят на работу, дорогу с/на нее, развлечение и питание. С появлением ресторанов, точек общепита, кальянных кафе и антикафе, люди все меньше времени уделяют приготовлению блюд дома. Однако, для того чтобы поесть в таких местах, необходимо зарезервировать столик, на месте сделать заказ и пока еда готовится ожидать. Нередко происходят ситуации, когда заказанный столик оказывается занятым или то блюдо, ради которого было выбрано именно это место сегодня, не подают. Чтобы решить данную проблему необходимо сделать эти процессы интерактивными для людей. Так, бронирование столика и заказ блюд будет более прозрачным для пользователей ресторана. Так как, большинство ресторанов имею собственные сети с уникальными для конкретной точки способами оплаты, видами меню, наиболее эффективным решением задачи является привлечение систем, состоящие из нескольких узлов, соединенных каналами связи. Это позволит менять тот или иной модуль в зависимости от потребностей той или иной точки сети ресторанов или всей сети в целом. Под узлами понимаются web-сервисы, непосредственно используемые для решения части общей задачи. С ростом количества узлов растёт и вероятность выхода из строя некоторых частей системы во время работы. Потеря работоспособности какого-либо из узлов – неизбежное событие, но можно избежать разрушительных последствий путём быстрого восстановления работоспособности вышедшего из строя узла.

Существующие аналоги

В качестве аналогов проектируемого программного обеспечения можно выделить следующие программные решения:

1. SmartReserve (https://smarteserve.ru).
2. Restoplace(https://www.restoplace.cc/).
3. Iiko(http://iiko.ru/).

Представленные решения предоставляют пользователю только бронирование столика, и представляют собой не только программное решение, но и обязательное аппаратное. Необходимость в специализированных планшетах для менеджеров делает такие системы неоправданно дорогими. Еще одним недостатком систем является отсутствие заказа блюд во время бронирования столика и неактуальность меню в момент бронирования. В связи с чем пользователь тратит время на изучение меню, а в момент посещения им заведения, узнает, что желанного блюда нет в меню. Все эти программные продукты реализованы в виде одного сервера, что усложняет масштабирование системы при увеличении области предоставляемых услуг.

Описание системы

Главное назначение ПО «Платформа для ресторанов» – предоставление пользователю возможности бронировать столик с выбором блюд для подачи. Для этого необходимо предоставить пользователю доступные столики в заведение, актуальное меню, возможность выбора столика и блюд для формирования заказа и оплаты. Для удобства пользователя при поиске в меню блюд предоставляются набор различных фильтров

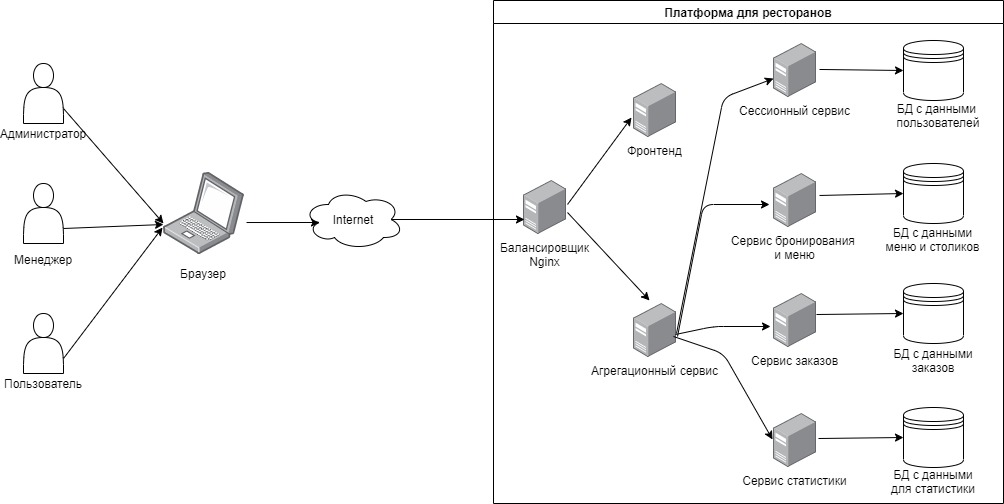


Рис. 1Топология ПО «Платформа для ресторанов».

приведена топология ПО «Платформа для ресторанов». ПО состоит из следующих частей:

* Фронтенд
* Агрегационный сервис
* Сессионный сервис
* Сервис бронирования и меню
* Сервис заказов и платежей
* Сервис статистики

Фронтенд

Основное назначение фронтенда заключается в отображении данных платформы в человеко-понятном и удобном виде, предоставление визуальных инструментов для взаимодействия с сервисами платформы. Предоставление WEB-интерфейса для удобства работы с порталом.

Агрегационный сервис

Предназначен для манипулирования данными между сервисами их агрегацией и отправкой пользователю.

Сессионный сервис

Предназначен для выполнения регистрации, аутентификации и авторизации пользователей в системе.

Сервис бронирования и меню

Предназначен для предоставления списка меню, управление им. Хранение расписания бронирования столиков.

Сервис заказов и платежей

Предназначен для формирования заказов, создания записей об оплате.

Сервис статистики

Предназначен для формирования статистических отчетов о действиях пользователей их выборе и частоте посещения заведения в различные дни.

Назначение разработки

ПО «Платформа для ресторанов» предназначена для предоставление удобного сервиса бронирования столика на вечер с предварительным выбором блюд из меню.

Требования к системе

1. Разрабатываемое программное обеспечение должно обеспечивать функционирование системы с доступностью 99.9%.
2. Максимальное время, выделяемое на профилактические работы 10дней.
3. Время восстановления системы после сбоя не должно превышать 20 минут.
4. Система должна поддерживать возможность «горячего» переконфигурирования системы. Необходимо поддержать возможность добавления нового узла во время работы системы без рестарта.

Требования к функциональным характеристикам

1. По результатам работы модуля сбора статистики медиана времени отклика системы на запросы, добавляющие или изменяющие информацию на портале не должна превышать 10 секунд без учета латентности географического расположения узла.
2. Портал должен обеспечивать возможность запуска в современных настольных браузерах: не менее 85% пользователей Интернета должны иметь возможность пользоваться порталом без какой-либо деградации функционала.

Функциональные требования к системе с точки зрения пользователя

ПО должна обеспечивать реализацию следующих функций:

1. Система должна обеспечивать регистрацию и авторизацию пользователей с учётной записью vk/gmail.
2. Система должна обеспечивать аутентификацию пользователей.
3. Система должна обеспечивать разделение пользователей на три роли:

* администратор;
* менеджер;
* пользователь;
* гость;

1. Система должна предоставлять **администратору** следующие функции:

* регистрация новых пользователей;
* удаление пользователей
* изменение информации о пользователе;
* добавление нового/изменение/удаление блюда в меню;
* редактирование бронирования столиков
* изменение информации заказа пользователя;
* просмотр отчетов статистики.

1. Система должна предоставлять **менеджеру** следующие функции:

* добавление нового/изменение/удаление блюда в меню;
* редактирование бронирования столиков
* изменение информации заказа пользователя;
* просмотр отчетов статистики.

1. Система должна предоставлять **клиенту** следующие функции:

* просмотр информации о столиках;
* просмотр меню;
* создание заказа с выбранным столиком и указанными блюдами;
* подтверждение заказа;
* просмотр заказов.

Входные данные системы

* 1. Система должна содержать следующую информацию о пользователях:
  + учётное имя пользователя в системе (необходимо для идентификации);
  + Пароль в хешированном виде по алгоритму SHA256 (симметричный алгоритм);
  + тип пользователя, который может быть следующим: администратор, менеджер, пользователь;
  + имя пользователя;
  + фамилия пользователя;
  + баллы лояльности
  1. Система должна содержать следующую информацию о блюде:
  + идентификатор (уникальный в пределах меню);
  + список ингредиентов;
  + название;
  + тип блюда;
  + Фото блюда;
  + описание;
  + доступность для заказа;
  + теги
  + стоимость;
  1. Система должна содержать следующую информацию о столиках:
  + идентификатор (уникальный в пределах ресторана);
  + номер столика;
  + время когда занят;
  + Количество мест;
  + доступность для заказа;
  + теги
  + стоимость;
  1. Система должна содержать следующую информацию о заказе:
* идентификатор (уникальный в пределах сервиса заказов);
* идентификатор столика;
* идентификатор пользователя, если он авторизован;
* Идентификаторы блюд, которые были заказаны;
* статус заказа;
* итоговая стоимость;
* Уникальный код заказа для идентификации;
* платеж
* дата создания
  1. Система должна содержать следующую информацию о платеже:
  + платежная система;
  + владелец карты в хешированном виде.
  + номер карты в хешированном виде.
  + дата создания

Выходные данные системы

Выходными параметрами системы являются веб-страницы. Доступность веб-страниц определяется ролью пользователя:

**Администратор** имеет доступ к веб-страницам:

* С конфигурацией меню
* С конфигурацией расписания столиков
* Со списками пользователей
* С конфигурацией отчетов статистики

**Менеджер** имеет доступ к веб-страницам:

* С конфигурацией меню
* С конфигурацией расписания столиков
* С конфигурацией отчетов статистики

**Зарегистрированный пользователь** имеет доступ к веб-страницам:

* Личного кабинета;
* Со списком столиков;
* Меню с возможностью фильтрации блюд
* С информацией о заказах;
* С оплатой заказа;
* С информацией о платёжных средствах;

**Гость** получает доступ к веб-страницам:

* Регистрации и авторизации;
* Со списком столиков;
* Меню с возможностью фильтрации блюд
* С информацией о заказах;
* С оплатой заказа;
* С информацией о платёжных средствах;

## Требования по реализации со стороны заказчика.

1. Требуется использовать СОА (сервис-ориентированную архитектуру).
2. Система состоит из микросервисов, отвечающих за свой элемент бизнес-логики.
3. Взаимодействие между сервисами осуществляется посредством HTTP-запросов, WebSocket соединений и/или очереди сообщений.
4. Требуется обеспечить возможность регистрации и авторизации пользователей с использованием аккаунта vk и gmail
5. Необходимо реализовать один веб-интерфейс для фронтенда.
6. Запрещается непосредственное обращение одного сервиса к базам данных других сервисов.
7. Все сервисы, кроме агрегационного недоступны пользователю путем прямого взаимодействия, это реализуется их расположением во внутренней сети.
8. Валидация входных данных должна производиться и на стороне пользователя с помощью JavaScript скриптов, и на стороне сервиса агрегации. Другие сервисы не должны валидировать входные данные, так как пользователь не может к ним обращаться напрямую, сервисы должны получать уже отфильтрованные входные данные от агрегатора.

Требования по программной реализации

1. Пароли учётных записей должны подвергаться хешированию.
2. Узел должен обладать уникальным идентификатором в пределах системы диагностики, который должен храниться в системе диагностики.
3. Система должна предоставлять как минимум два интерфейса – интерфейс пользователя и администратора.
4. Для хранения данных использовать СУБД MongoDB.

Требования к составу и параметрам технических средств

Минимальные требования к компьютерам, на которой развертывается программное обеспечение:

* тактовая частота не менее 1.7 ГГц;
* оперативная память не менее 4 Гб;
* ОС версии не ниже Linux ubuntu 18.04 lts;
* свободное пространство на жестком диске не менее 40 Гб для ОС;

## Функциональные требования к сервисам.

1. ***Фронтенд***– это серверное приложение, при разработке которого необходимо учитывать следующие факторы:

* фронтенд должен принимать запросы по протоколу HTTP и формировать ответы пользователям портала в формате HTML или JSON;
* Фронтенд должен взаимодействовать только с агрегационным сервисом через балансировщик;
* запросы к сервису необходимо осуществляет по протоколу HTTP. Данные необходимо передавать в стандартизированных форматах для упрощения взаимодействия;

1. ***Сессионный сервис*** – это серверное приложение, которое должно отвечать следующим требованиям по разработке.

* Сессионный сервис должен принимать и возвращать данные в формате JSON, multipart/form-data, application/x-www-form-urlencoded по протоколу HTTP.
* Выполнять авторизацию пользователей, проверять и удалять сессию, а также

регистрировать пользователей.

* Поддерживать авторизацию и регистрацию пользователей через vk и gmail.
* База данных, которая содержит информацию о сессиях, должна находиться на этом же сервере.

1. ***Сервис бронирования и меню*** должен быть серверным приложением, которое:

* Должно принимать и отвечать на запросы в формате JSON по протоколу HTTP;
* Поддерживать функции создания, обновления и удаления записей о блюдах.
* Поддерживать функции бронирования, изменения статуса бронирования/отмены для столиков.
* Предоставлять список столиков в формате JSON с пагинацией.
* Предоставлять список блюд в формате JSON с пагинацией.

1. ***Сервис заказов и платежей*** должен быть серверным приложением, которое:

* должно принимать и отвечать на запросы в формате JSON по протоколу HTTP;
* обрабатывать запросы на создание, удаление, редактирование и получение заказа пользователя по ключу – идентификатору пользователя;
* обрабатывать запросы на получение списка заказов.

1. ***Сервис статистики*** должен быть серверным приложением, которое:

* принимает и отвечает на запросы в формате JSON по протоколу HTTP;
* принимает данные о действиях пользователя в формате JSON по протоколу MQTT;
* обрабатывает запросы на создание записи о действиях пользователя;
* предоставляет список действий по фильтру только администратору;

Сценарий взаимодействия фронтенда и сервисов

Рассмотрим, как взаимодействуют фронтенд и сервисы на примере выполнения запроса от пользователя на получение списка заказов.

1. На фронтенд приходит данный запрос пользователя.
2. Пользователю предоставляется пустая страница без контента.
3. Фронтенд отправляет запрос на получение списка заказов с данными пользователя
4. Агрегатор анализирует заголовки запроса пользователя и, если пользователь был авторизован ранее, получает из них токен пользователя – набор символов. Далее выполняется запрос к сессионному сервису для проверки авторизации пользователя, отправляя заголовки HTTP запроса на сессионный сервис. Если не удалось выполнить успешный запрос, пользователю возвращается ошибка.
5. Выполняется запрос к сервису заказов и платежей. Агрегатор отправляет запрос к сервису заказов. Если запрос к сервису заказов не был выполнен успешно, пользователю возвращается ошибка.
6. Выполняется запрос к сервису бронирования и меню. Агрегатор отправляет запрос к сервису, запрашивая блюда, содержащиеся в заказе. Если запрос к сервису бронирования и меню не был выполнен успешно, выполняется деградация содержимого.
7. Если ошибки не произошло, то производится генерация ответа для фронтенда содержимого страницы, полученных от сервиса заказов, сервиса бронирования и меню и сессионного сервиса.

Сценарий взаимодействия сервисов с СУБД

Рассмотрим, как происходит взаимодействие сервисов с СУБД на примере сервиса “бронирования и меню” и запроса на получение списка блюд.

1. Сервис бронирования и меню получает запрос от агрегатора и преобразовывает его в запрос к СУБД.
2. Сервис преобразует фильтры для поиска блюд в формат СУБД.
3. СУБД выполняет поиск блюд, используя предоставленные критерии поиска
4. Каждый результат приводится к единому объекту, полученному от СУБД.

Сценарии функционирования системы

Создание нового блюда

Действие должно выполняться пользователем с типом учётной записи «Администратор» или «Менеджер» посредством интерфейса и состоять из следующих шагов.

1. Администратор вводит логин и пароль. Если аутентификация пройдена успешно, то осуществляется переход к следующему шагу, в противном случае система должна предложить пользователю повторить этот шаг.
2. Администратор выбирает функцию «Добавить блюдо» и открывается страница с информационными полями для создания меню.
3. Администратор заполняет все обязательные информационные поля.
4. Система проводит валидацию введённых данных, проверяя заполнение необходимых полей и соответствие введенной информации требуемым шаблонам.
5. Администратор подтверждает ввод данных. В случае ввода некорректных данных система должна выдавать пользователю сообщение об ошибке с указанием положения некорректных данных и предложить повторить шаг 3.

Удаление блюда

Действие должно выполняться пользователем с типом учётной записи «Администратор» или «Менеджер» посредством интерфейса и состоять из следующих шагов.

1. Администратор вводит логин и пароль. Если аутентификация пройдена успешно, то осуществляется переход к следующему шагу, в противном случае система должна предложить пользователю повторить этот шаг.
2. Администратор переходит на страницу управления меню.
3. Администратор получает список меню
4. У необходимого блюда выбирает функцию удалить блюдо
5. Система выдает сообщение о подтверждении удаления узла и возможных последствиях.
6. Администратор подтверждает удаление, система удаляет выбранный узел.

Редактирование блюда

Действие должно выполняться пользователем с типом учётной записи «Администратор» или «Менеджер» посредством интерфейса и состоять из следующих шагов.

1. Администратор вводит логин и пароль. Если аутентификация пройдена успешно, то осуществляется переход к следующему шагу, в противном случае система должна предложить пользователю повторить этот шаг.
2. Администратор переходит на страницу управления меню.
3. Администратор получает список меню
4. У необходимого блюда администратор выбирает функцию «Редактировать блюдо» и переходит на страницу редактирования с представленными информационными полями о блюде.
5. Администратор заполняет все обязательные информационные поля.
6. Система проводит валидацию введённых данных, проверяя заполнение необходимых полей и соответствие введенной информации требуемым шаблонам.
7. Администратор подтверждает ввод данных. В случае ввода некорректных данных система должна выдавать пользователю сообщение об ошибке с указанием положения некорректных данных и предложить повторить шаг 3.

Регистрирование нового пользователя

Действие должно выполняться пользователем с типом учётной записи «Администратор» посредством интерфейса администратора и состоять из следующих шагов.

1. Администратор вводит логин и пароль. Если аутентификация пройдена успешно, то осуществляется переход к следующему шагу, в противном случае система должна предложить пользователю повторить этот шаг.
2. Администратор переходит на страницу управления пользователями.
3. Администратор выбирает функцию «Добавить пользователя».
4. Администратор заполняет все обязательные информационные поля.
5. Система выполняет проверку ввода всех необходимых данных.
6. Администратор подтверждает добавление пользователя. В случае неудачного выполнения проверки на шаге 4 система выдает сообщение об ошибке с указанием положения некорректных данных.

Регистрация пользователем

Действие должно выполняться пользователем, не зарегистрированным в системе, и состоит из следующих шагов.

1. Гость выбирает функцию "регистрация".
2. Заполняет необходимые поля.
3. Система выполняет проверку ввода всех необходимых данных.
4. Гость подтверждает регистрацию. В случае неудачного выполнения проверки на шаге 3 система выдает сообщение об ошибке с указанием положения некорректных данных.

Регистрация пользователем через vk

Действие должно выполняться пользователем, не зарегистрированным в системе, и состоит из следующих шагов.

1. Гость выбирает функцию "регистрация через vk".
2. Пользователь перенаправляется на vk и даёт согласие на использование данных его учётной записи. При отсутствии согласия – перенаправляется на страницу ошибки.
3. Пользователь перенаправляется на страницу регистрации, поля ввода заполнены данными из учётной записи mail.
4. Система выполняет проверку ввода всех необходимых данных.
5. Гость подтверждает регистрацию. В случае неудачного выполнения проверки на шаге 4 система выдает сообщение об ошибке с указанием положения некорректных данных.

Удаление пользователя

Действие должно выполняться пользователем с типом учётной записи «Администратор» посредством интерфейса администратора и состоять из следующих шагов.

1. Администратор вводит логин и пароль. Если аутентификация пройдена успешно, то осуществляется переход к следующему шагу, в противном случае система должна предложить пользователю повторить этот шаг.
2. Администратор переходит на страницу управления пользователями.
3. У необходимого пользователя администратор выбирает функцию «Удалить пользователя».
4. Система выдает сообщение о подтверждении удаления пользователя и возможных последствиях.
5. Администратор подтверждает удаление, система удаляет выбранного пользователя.

Замечание: необходимо предусмотреть невозможность удаления администратора, который в данный момент выполняет операции администрирования.

Формирование заказа

Действие должно выполняться пользователем зарегистрированным или не зарегистрированным в системе и состоит из следующих шагов.

1. Пользователь переходит на страницу “Забронировать столик”
2. Выбирает необходимый столик из списка. При просмотре рядом с каждым столиком написана цена и свободное время.
3. У выбранного столика открывается меню настроек бронирования. Пользователь указывает время бронирования (или бронирование на весь вечер)
4. Система выполняет проверку ввода всех необходимых данных.
5. Пользователь переходит на страницу меню и выбирает соответствующие блюда из меню по фильтрам.
6. Система выполняет проверку ввода всех данных и перенаправляет на итоговый заказ
7. Пользователь подтверждает заказ. В случае неудачного выполнения проверки на шаге 3 система выдает сообщение об ошибке с указанием положения некорректных данных.

Оплата заказа

Действие должно выполняться пользователем, зарегистрированным в системе, и состоит из следующих шагов.

1. Пользователь выбирает раздел "заказы".
2. Выбирает из списка заказ, который необходимо оплатить и переходить на страницу оплаты.
3. Пользователь заполняет все информационные поля
4. Система выполняет проверку ввода всех необходимых данных.
5. Пользователь подтверждает оплату. В случае неудачного выполнения проверки на шаге 3 система выдает сообщение об ошибке с указанием положения некорректных данных.

Пользовательский интерфейс

Для реализации серверного приложения фронтенда используется схема «модель – вид – элемент управления» (Model – View – Controller, MVC). Пользовательский интерфейс представляет собой “вид” из этой схемы. Такое разделение позволяет абстрагировать пользовательский интерфейс от взаимодействия фронтенда с бекендами : разработка интерфейса осуществляется с помощью языка разметки HTML при помощи языка шаблонизации ejs.

Пользовательский интерфейс в разрабатываемой системе должен обладать следующими характеристиками:

* ***Кроссбраузерность*** – способность интерфейса работать в основных браузерах, таких как:
  + Google Chrome
  + Opera
  + Firefox
  + Edge
* ***«Плоский» дизайн*** – это один из новых трендов в мире дизайна, суть которого заключается в отказе от объемных элементов (теней элементов, объемных кнопок и т. д.) и в замене их плоскими аналогами.
* ***Расширяемость*** – возможность легко расширять и модифицировать пользовательский интерфейс. Это свойство обеспечивается за счет модульности исходного кода пользовательского интерфейса (разбиение на файлы, общие блоки, наследование от общего шаблона).
* ***Доступность*** – любая страница портала доступна c минимумом переходов с главной страницы портала. Это стандартное правило при проектировании пользовательского интерфейса, оно позволяет сильно упростить доступ пользователей к контенту на портале и приводит к хорошей структуре страниц портала.
* ***Интуитивно понятный интерфейс*** – все кнопки имеют подписи при наведении на них, многие содержат иконки, облегчающие восприятие пользователем. Используются стандартные элементы управления.

***Быстрый отклик*** – используются специальные техники для ускорения работы страниц портала такие как:

* сжатие изображений
* отложенное выполнение скриптов
* обфускация скриптов и объединение файлов.

Концептуальный дизайн

Концептуальная модель системы в нотации IDEF0

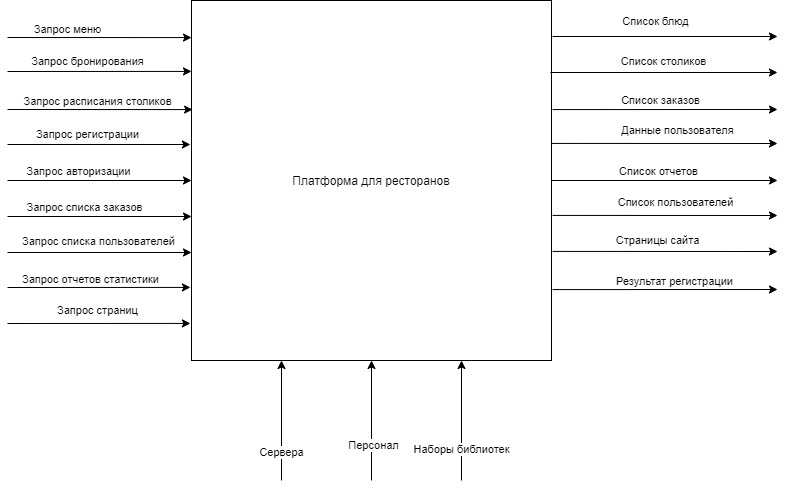


Рис. 2. Концептуальная модель системы в нотации IDEF0.

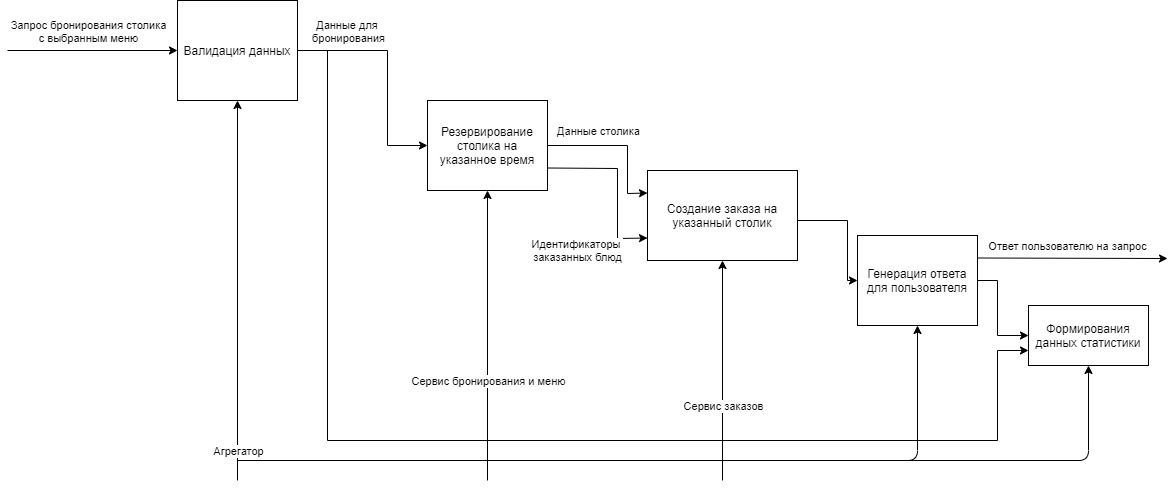


Рис 3. Детализированная концептуальная модель системы в нотации IDEF0 для запроса бронирования столика.

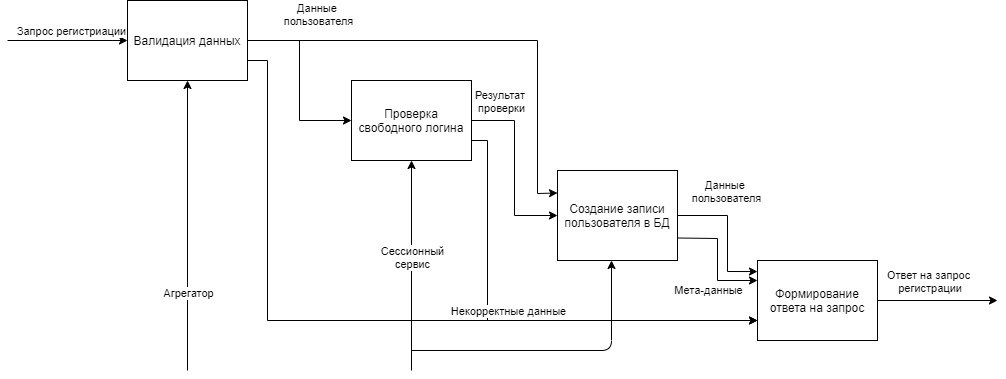


Рис 4. Детализированная концептуальная модель системы в нотации IDEF0 для регистрации клиента.

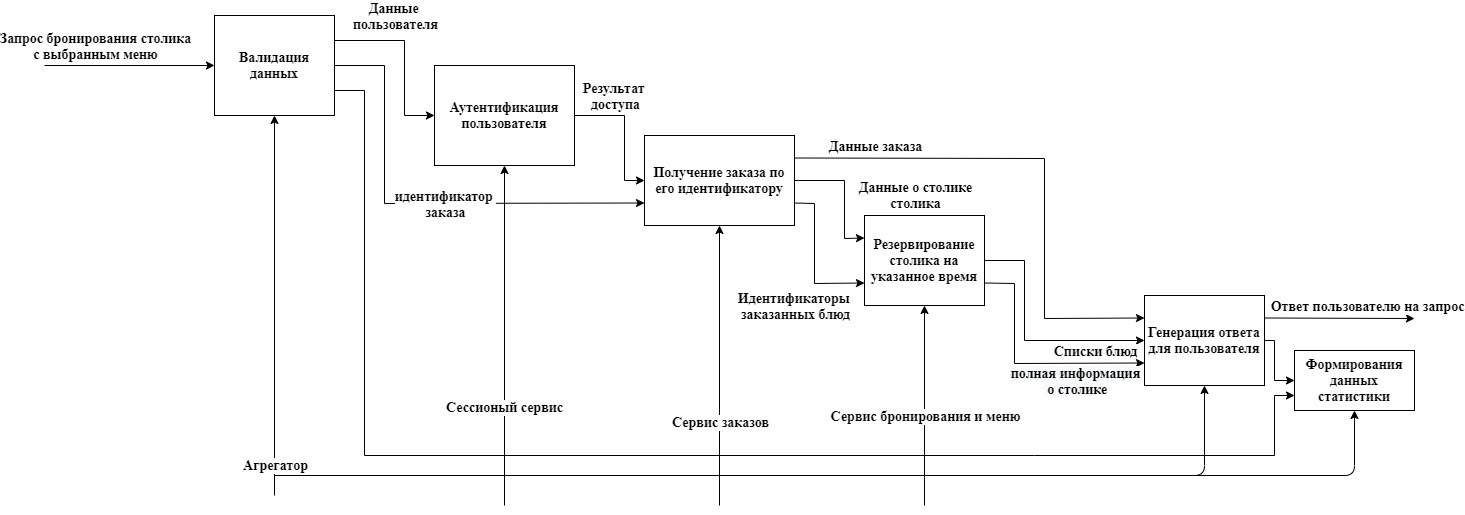


Рис 5. Детализированная концептуальная модель системы в нотации IDEF0 для запроса списка заказов.

Диаграммы прецедентов

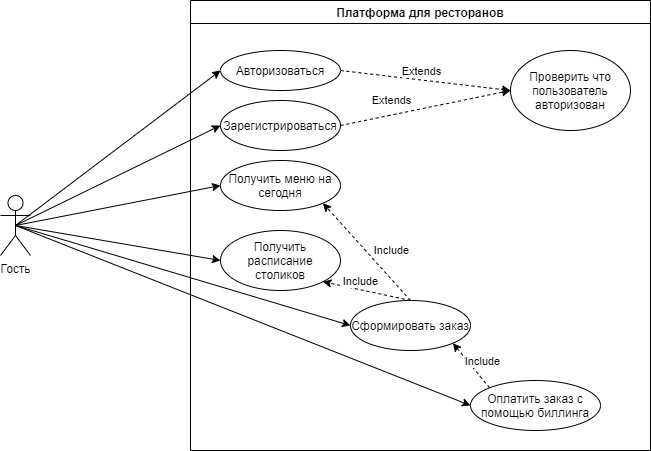


Рис.6. Диаграмма прецедентов пользователя с точки зрения **гостя**

**Спецификация сценария регистрации пользователя в системе**

**Нормальный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на регистрацию в системе | Генерация страницы для заполнения данных |
| Отправка данных | Система валидирует данные и проверяет наличие пользователя с аналогичным логином  Отправка страницы результата регистрации |

**Спецификация сценария регистрации пользователя в системе**

**Альтернативный ход сценария**.

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на регистрацию в системе | Генерация страницы для заполнения данных |
| Актер отправляет некорректные данные | Система валидирует данные и отправляет результат некорректных данных |

**Спецификация сценария регистрации пользователя в системе.**

**Альтернативный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на регистрацию в системе | Генерация страницы для заполнения данных |
| Актер отправляет логин, который уже занят | Система сообщает, что выбранный узел не существует. |

**Спецификация сценария авторизации пользователя в системе**

**Нормальный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на авторизацию в системе | Генерация страницы для заполнения данных |
| Отправка данных | Система валидирует данные и проверяет наличие пользователя с указанным логином и проверяет пароль для данного пользователя  Отправка страницы результата авторизации |

**Спецификация сценария авторизации пользователя в системе**

**Альтернативный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на авторизацию в системе | Генерация страницы для заполнения данных |
| Актер отправляет неверный пароль | Система валидирует данные и проверяет наличие пользователя с указанным логином и проверяет пароль для данного пользователя  Отправка результата провальной авторизации |

**Спецификация сценария авторизации пользователя в системе**

**Альтернативный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на авторизацию в системе | Генерация страницы для заполнения данных |
| Актер отправляет логин незарегистрированного в системе | Система валидирует данныеи отправляет результат о провальной авторизации |

**Спецификация сценария получения меню в системе**

**Нормальный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на получение меню в системе | Генерация страницы с предустановленными блюдами из меню с возможностью заполнения формы для их фильтрации |
| Отправка данных для получения меню по фильтрам | Система валидирует, трансформирует данные и производит поиск по указанным критериям блюда из меню.  Отправка списка блюд в формате JSON |

**Спецификация сценария получения меню в системе**

**Альтернативный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на получение меню в системе | Генерация страницы с предустановленными блюдами из меню с возможностью заполнения формы для их фильтрации |
| Актер отправляет фильтры, согласно которым блюда в меню нет | Система валидирует, трансформирует данные и производит поиск по указанным критериям блюда из меню.  Отправка пустого списка блюд в формате JSON |

**Спецификация сценария оплаты заказа гостем в системе**

**Нормальный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на оплату заказа в системе | Генерация страницы с формой платежной системы |
| Отправка данных корректных данных | Система валидирует данные и подтверждает оплату Отправка страницы успешной оплаты |

**Спецификация сценария оплаты заказа гостем в системе**

**Альтернативный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на оплату заказа в системе | Генерация страницы с формой платежной системы |
| Актер отправляет корректные данные, но оплата происходит с задержкой | Система валидирует данные и ожидает ответа от платежной системы.  Отправка страницы с ожиданием платежа от платежной системы  Отправка сообщения что оплата прошла |

**Спецификация сценария оплаты заказа гостем в системе**

**Альтернативный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос на оплату заказа в системе | Генерация страницы с формой платежной системы |
| Актер отправляет корректные данные, но оплата происходит с задержкой, но средств недостаточно | Система валидирует данные и ожидает ответа от платежной системы.  Отправка страницы с ожиданием платежа от платежной системы  Отправка сообщения что оплата прошла |

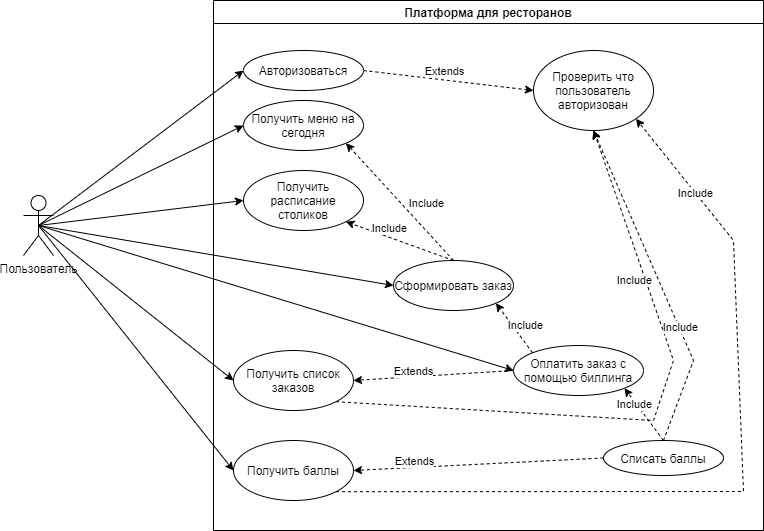


Рис. 7. Диаграмма прецедентов с точки зрения **пользователя**

**Спецификация сценария формирования заказа**

**Нормальный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Актер запрашивает форму бронирования | Генерация страницы бронирования |
| Актер выбирает дату, столик и блюда из меню | Система валидирует данные,  Отправка страницы результата формирования заказа, в которой указан номер телефона для связи, время брони, столик и список блюд |

**Спецификация сценария формирования заказа**

**Альтернативный ход сценария**.

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Актер запрашивает форму бронирования | Генерация страницы бронирования |
| Актер отправляет некорректную дату | Система валидирует данные и отправляет результат о некорректных данных |

**Спецификация сценария формирования заказа**

**Альтернативный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Актер запрашивает форму бронирования | Генерация страницы бронирования |
| Актер выбирает дату, столик и блюда из меню, которые сейчас недоступны | Система валидирует данные и отправляет результат о попытке заказать блюдо, которое не подается |

**Спецификация сценария оплаты заказа авторизованным пользователем**

**Нормальный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Актер запрашивает форму оплаты | Генерация страницы оплаты |
| Актер не включает в итоговую сумму баллы | Генерация страницы с формой платежной системы с полной стоимостью заказа |
| Отправка данных корректных данных | Система валидирует данные и подтверждает оплату Отправка страницы успешной оплаты |

**Спецификация сценария оплаты заказа авторизованным пользователем**

**Альтернативный ход сценария**.

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Актер запрашивает форму оплаты | Генерация страницы оплаты |
| Актер включает в итоговую сумму баллы, не покрывающие заказ | Система валидирует данные, проверяет наличие баллов, списывает их в счет заказа  Генерирует страницу с формой платежной системы с остаточной стоимостью заказа |
| Отправка данных корректных данных | Система валидирует данные и подтверждает оплату Отправка страницы успешной оплаты |

**Спецификация сценария оплаты заказа авторизованным пользователем**

**Альтернативный ход сценария**.

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Актер запрашивает форму оплаты | Генерация страницы оплаты |
| Актер включает в итоговую сумму баллы, полностью покрывающие заказ | Система валидирует данные, проверяет наличие баллов, списывает их в счет заказа, кроме 1 рубля  Генерирует страницу с формой платежной системы со стоимостью 1 рубль |
| Отправка данных корректных данных | Система валидирует данные и подтверждает оплату Отправка страницы успешной оплаты |

**Спецификация сценария оплаты заказа авторизованным пользователем**

**Альтернативный ход сценария**.

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Актер запрашивает форму оплаты | Генерация страницы оплаты |
| Актер включает в итоговую сумму баллы, полностью покрывающие заказ | Система валидирует данные, проверяет наличие баллов, списывает их в счет заказа, кроме 1 рубля  Генерирует страницу с формой платежной системы со стоимостью 1 рубль |
| Отправка данных, оплата происходит с задержкой | Система валидирует данные и ожидает ответа от платежной системы.  Отправка страницы с ожиданием платежа от платежной системы  Отправка сообщения что оплата прошла |

**Спецификация сценария оплаты заказа авторизованным пользователем**

**Альтернативный ход сценария**.

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Актер запрашивает форму оплаты | Генерация страницы оплаты |
| Актер включает в итоговую сумму баллы, полностью покрывающие заказ | Система валидирует данные, проверяет наличие баллов, списывает их в счет заказа, кроме 1 рубля  Генерирует страницу с формой платежной системы со стоимостью 1 рубль |
| Отправка данных, оплата не происходит. Актер закрыл сайт/страницу и время проведения платежа прошло | Система помечает заказ отмененным, баллы возвращаются на счет пользователя. |

Рис. 8. Диаграмма прецедентов с точки зрения менеджера

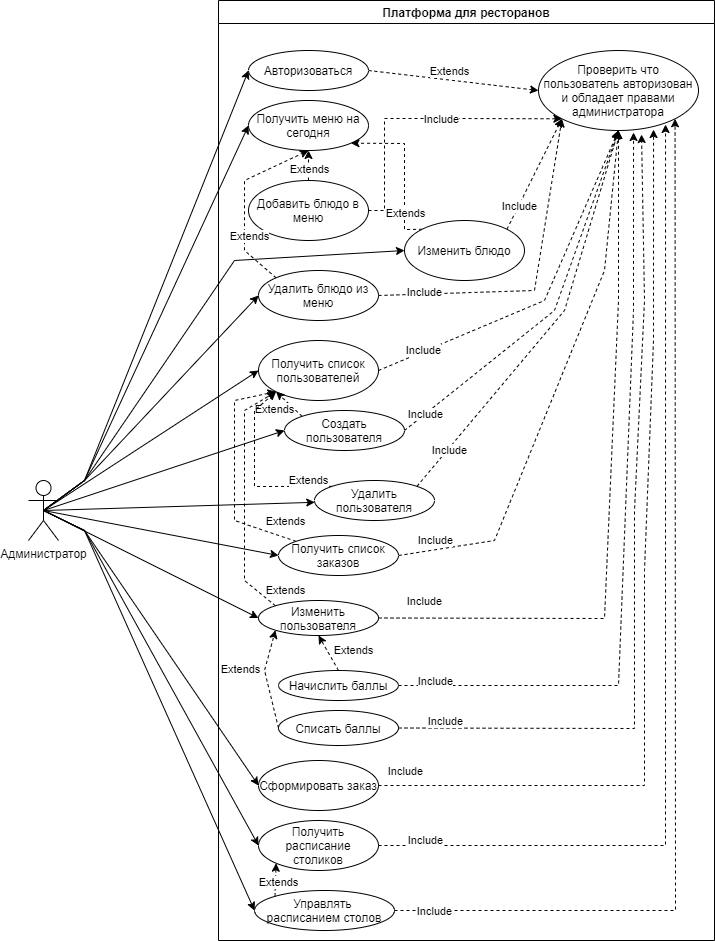


Рис. 9. Диаграмма прецедентов с точки зрения администратора

**Спецификация сценария создания блюда**

**Нормальный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос создания блюда | Система отправляет форму для ввода данных |
| Актер заполняет форму | Система валидирует данные, создает запись.  Отправка страницы результата создания блюда |

**Спецификация сценария создания блюда**

**Альтернативный ход сценария**.

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос создания блюда | Система отправляет форму для ввода данных |
| Актер заполняет форму с некорректными данными | Система валидирует данные.  Отправка страницы об ошибке заполнения формы |

**Спецификация сценария создания блюда**

**Альтернативный ход сценария**.

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос создания блюда | Система отправляет форму для ввода данных |
| Актеру не заполняет форму на создание блюда и либо завершает работу с порталом, либо осуществляет выполняет другие операции дальше |  |

**Спецификация сценария удаления блюда**

**Нормальный ход сценария.**

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос удаления блюда из меню | Система запрашивает подтверждения операции |
| Актер подтверждает удаление | Система удаляет блюдо  Отправка страницы результата удаления блюда |

**Спецификация сценария удаления автомобиля**

**Альтернативный ход сценария**.

|  |  |
| --- | --- |
| **Действия актера** | **Отклик системы** |
| Запрос удаления блюда из меню | Система запрашивает подтверждения операции |
| Актер не подтверждает удаление и либо завершает работу с порталом, либо осуществляет другие операции дальше |  |

Логический дизайн

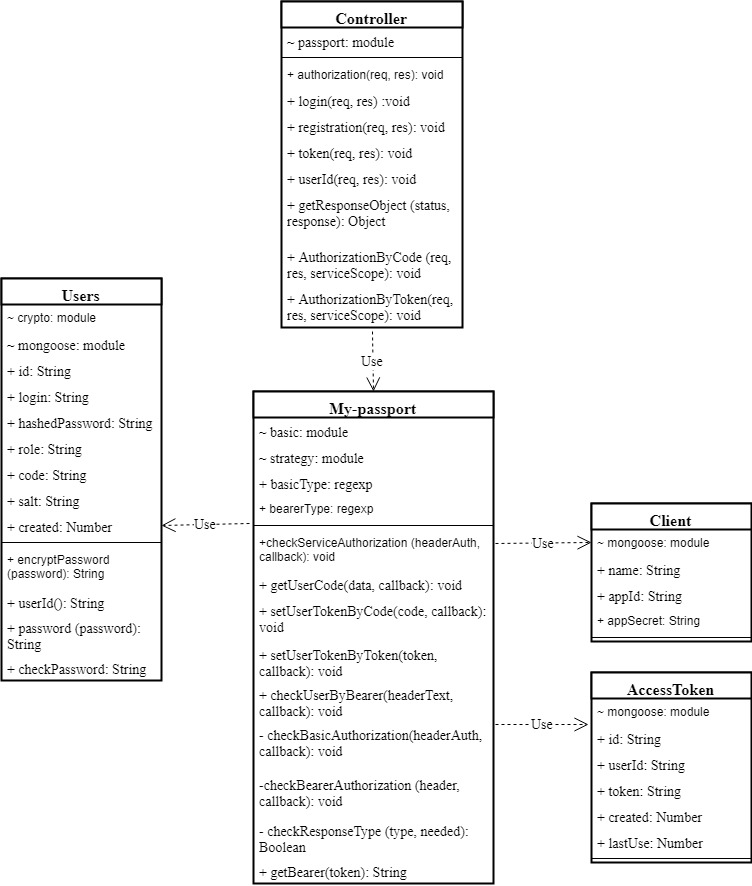


Рис. 9. Диаграмма классов "Сессионного бекенда" (уровень реализации)

**Спецификации классов**

Класс **Controller** представляет собой базовый класс для обработки запросов Сессионного бекенда.

Методы класса **Controller**:

|  |  |
| --- | --- |
| Название метода | Описание метода |
| Authorization(req, res): void | Отправляет страницу для авторизации |
| Login(req, res): void | Обрабатывает запрос на авторизацию пользователя |
| Registration(req, res): void | Обрабатывает запрос на регистрацию пользователя |
| Token(req, res):void | Обрабатывает запрос получение токена по refreshToken |
| userId(req, res):void | Обрабатывает запрос служебной информации о пользователе |
| getResponseObject(status, response): Object | Формирует ответ для фронтенда |
| AuthorizationByCode(req,res, serviceScope): void | Обрабатывает запрос авторизации по code |
| AuthorizationByToken(req, res, serviceScope): void | Обрабатывает запрос авторизации пользователя по Token |

Атрибуты класса **Controller**:

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| Passport | Class: module | Класс для проверки авторизации |

Класс **My-passport** представляет собой набор методов для удобной и быстрой проверки авторизации, выполнение регистрации и предоставление токена для авторизированных пользователей

Методы класса **My-passport**:

|  |  |
| --- | --- |
| Название метода | Описание метода |
| checkServiceAuthorization(headerAuth, callback): void | Проверяет авторизирован ли сервис запрашивающий информацию |
| getUserCode(data, callback): void | Возвращает информацию о пользователе по code |
| setUserTokenByToken(token, callback):void | Установка нового токена для пользователя по refreshToken |
| setUserTokenByCode(code, callback):void | Установка токена для пользователя по code |
| checkBasicAuthorization(headerAuth, callback): void | Проверяет basic авторизацию пользователя |
| checkBearerAuthorization(headerAuth, callback): void | Проверяет bearer авторизацию пользователя |
| checkResponseType(type, needed): Boolean | Проверяет тип запроса авторизации |
| getBearer(token): String | Возвращает Bearer токен |

Атрибуты класса **My-passport**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| Basic | Package: module | Модуль для считывания заголовка запроса |
| Strategy | Package: module | Модуль для формирования стратегии авторизации |
| basicType | Public: regexp | Регулярное выражение проверки basic авторизации |
| bearerType | Public: regexp | Регулярное выражение проверки bearer авторизации |

Класс **Client** представляет собой запись о сервиса. Класс используется для проверки доверительных сервисов, которым можно предоставлять информацию и обрабатывать их запросы

Атрибуты класса **Client**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| Mongoose | Package: module | Модуль для взаимодействия объекта с записью в БД |
| Name | Public: String | Имя сервиса |
| appId | Public: String | Id сервиса |
| appSecret | Public: String | Код сервиса |

Класс **User** представляет собой профиль пользователя. Класс используется для работы с пользователями и для хранения их в базе данных с помощью технологии ORM, которая позволяет работать с сущностями базы данных как с обычными объектами.

Методы класса **User**:

|  |  |
| --- | --- |
| Название метода | Описание метода |
| encryptPassword(password): String | Возвращает полученный пароль в захешированном виде |
| userId():String | Возвращает id пользователя в формате строки |
| сheckPassword(password): String | Проверяет соответствие пароля |
| Password():String | Возвращает хешированный пароль пользователя |

Атрибуты класса **User**:

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| Crypto | Package: module | Модуль предоставляющий криптографический функционал |
| Mongoose | Package: module | Модуль для реализации технологии ORM |
| Id | Public: String | Id пользователя в базе данных |
| Login | Public: String | Логин пользователя |
| hashedPassword | Public: String | Захешированный пароль пользователя |
| Role | Public: String | Роль пользователя в системе |
| Code | Public: String | Уникальный код пользователя для авторизации |
| Salt | Public: String | Ключ для шифрования пароля |
| Created | Public: Number | Дата создания записи в формате UNIX time |

Класс **AccessToken** представляет собой запись о предоставленных пользователям токенам. Класс используется для работы с токенами и для хранения их в базе данных с помощью технологии ORM, которая позволяет работать с сущностями базы данных как с обычными объектами.

Атрибуты класса **AccessToken**:

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип атрибута | Описание атрибута |
| Mongoose | Package: module | Модуль для реализации технологии ORM |
| Id | Public: String | Id accessToken'а |
| userId | Public: String | Id владельца токена |
| Token | Public: String | Токен |
| Created | Public: Number | Дата создания записи в формате UNIX time |
| lastUse | Public: Number | Дата последнего использования токена в формате UNIX time |

На рисунках 10, 11 представлены диаграммы деятельности для авторизации и регистрации соответственно.



Рис. 10 Диаграмма деятельности для варианта использования "авторизация" для пользователя



Рис. 11. Диаграмма деятельности для варианта использования "Регистрация" для пользователя

На рисунке 12 представлена диаграмма последовательности действий для сложного запроса "Список заказов пользователя"

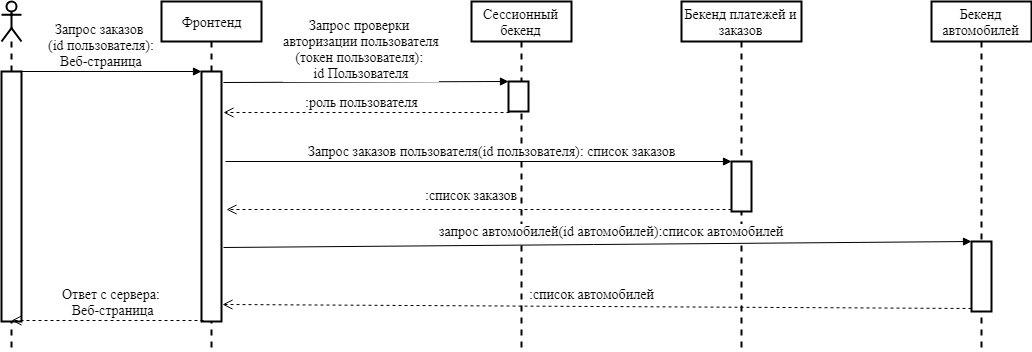


Рис. 12. Диаграмма последовательности действий при запросе списка заказов пользователя

(концептуальный уровень)

В нашей системе предполагается распределенное хранение данных. Диаграмма потоков данных, представленная на Рис. 13, описывает, какая информация в каком хранилище данных будет храниться. Каждое хранилище данных будет реализовано в виде базы данных.

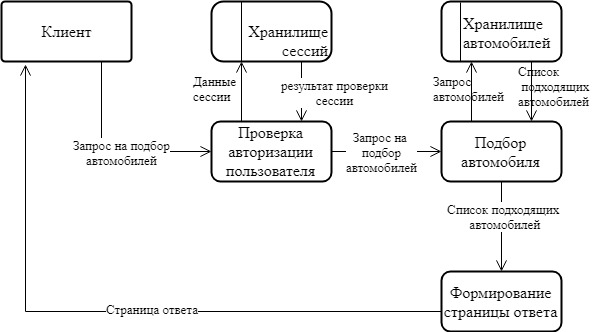


Рис. 13. Диаграмма потоков данных системы.

**Структура баз данных**

В данном разделе приведена структура баз данных системы. На рисунке 14 приведена ER-диаграмма баз данных.

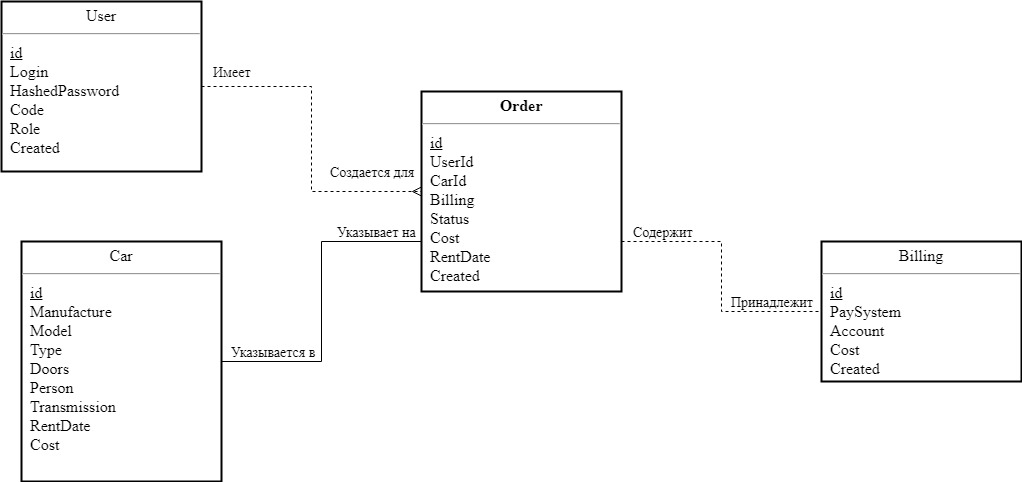


Рис. 14. ER-диаграмма баз данных системы.

Сущность **User** содержит информацию о пользователе.

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Описание |
| Id | ObjectId | Id записи в базе данных |
| Login | String | Логин пользователя |
| HashedPassword | String | Пароль пользователя в захешированном виде |
| Role | String | Роль пользователя |
| Created | Number | Дата создания записи в формате UNIX Time |

Сущность **Car** содержит информацию об автомобиле.

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Описание |
| Id | ObjectId | Id записи в базе данных |
| Manufacture | String | Производитель автомобиля |
| Model | String | Модель автомобиля |
| Type | String | Тип кузова автомобиля |
| Doors | Number | Количество дверей у автомобиля |
| Person | Number | Количество человек, которых можно перевести в машине |
| Transmission | String | Тип коробки передач |
| RentDate | Object | Время аренды автомобиля |
| Cost | Number | Стоимость аренды автомобиля в день |

Сущность **Order** содержит информацию о заказе пользователя.

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Описание |
| Id | ObjectId | Id записи в базе данных |
| userId | String | Id владельца заказа |
| carId | String | Id арендуемой машины |
| Billing | Billing | Платеж заказа |
| Status | String | Статус заказа |
| Cost | Number | Итоговая стоимость заказа |
| Created | Number | Дата создания заказа в формате UNIX Time |
| RentDate | Object | Время аренды автомобиля |

Сущность **Billing** содержит информацию о заказе пользователя.

|  |  |  |
| --- | --- | --- |
| Название поля | Тип | Описание |
| Id | ObjectId | Id записи в базе данных |
| PaySystem | String | Название способа оплаты |
| Account | String | Номер карты с которой сделан перевод |
| Cost | Number | Сумма оплаты |
| Created | Number | Дата создания заказа в формате UNIX Time |

**Пользовательский интерфейс**

**Физический дизайн системы.**

Выбор операционной системы

Согласно требованиям технического задания, разрабатываемый портал должен обладать высокой доступностью, работать на типичных архитектурах ЭВМ (Intel x86, Intel x64), а так же быть экономически недорогим для сопровождения. Таким образом, можно сформулировать следующие требования к операционной системе:

* **Распространенность.** На рынке труда должно быть много специалистов, способных администрировать распределенную систему, работающую под управлением выбранной операционной системы.
* **Надежность**. Операционная система должна широко использоваться в стабильных проектах, таких как Mail.Ru, yander.ru, Google.com. Эти компании обеспечивают высокую работоспособность своих сервисов, и на их опыт можно положиться.
* **Наличие требуемого программного обеспечения.** Выбор операционной системы не должен ограничивать разработчиков в выборе программного обеспечения, библиотек.
* **Цена.**

Под данные требования лучше всего подходит ОС Ubuntu [13].Ubuntu — это дистрибутив, использующий ядро Linux. Как и все дистрибутивы Linux, Ubuntu является ОС с открытым исходным кодом, бесплатным для использования.. Ubuntu простой в эксплуатации дистрибутив, имеющий систему управления пакетами (APT). Поставляется как в клиентской (с графическим интерфейсом), так и в серверной (без графического интерфейса) версиями.

Выбор СУБД для профилей пользователей

Проанализируемтехническое задание на разработку бекенда профилей. Из его сценариев использования можно выделить следующие важные моменты:

* **Отсутствует необходимость комбинирования и сравнения профилей друг с другом.** Отсюда возможно сделать вывод, что нам не требуется использовать реляционные СУБД.
* **Структура профилей может меняться.** В процессе развития портала могут появляться новые поля. По этой причине лучше всего использовать СУБД, которая не задает жесткую схему хранения данных.
* **СУБД должна быть быстрой и масштабируемой.** По этой причине СУБД должна иметь высокую скорость отклика, поддерживать репликацию и щардинг.

Нереляционная СУБД MongoDB [12] полностью отвечает приведенным выше требованиям. MongoDB - нереляционная система управления базами данных. Относится к классу документо-ориентированных СУБД. В качестве языка запросов используется язык, в основе которого лежит JavaScript. JavaScript легок для освоения, что снижает порог вхождения для эксплуатации этой СУБД. Документы хранятся в виде JSON - стандартном для JavaScript способе представления данных. Эта СУБД поддерживает индексацию, профилирование запросов (выявление статистики потребления ресурсов запросами), репликацию, шардирование. MongoDB не поддерживает транзакции. Эта СУБД не требует описания схемы данных.

В данном проекте выбрана MongoDB. Данная СУБД использует асинхронную репликацию (при асинхронной репликации изменение данных в репликах базы данных происходит асинхронно, то есть не обязательно во время транзакции, возможно, после ее завершения) типа «ведущий – ведомый» (данные может изменять только «ведущий», читать данные может как «ведущий», так и «ведомый»).

Выбор СУБД для меню и столиков

В соответствии с техническим заданием разработка бекенда автомобилей предусматривает следующие требования:

* **Безопасность хранения данных**. Несанкционированный доступ к счетам пользователей должен быть невозможен.
* **Структура автомобилей может меняться.** В процессе развития портала могут появляться новые поля. По этой причине лучше всего использовать СУБД, которая не задает жесткую схему хранения данных.
* **Масштабируемость.** Выбранная СУБД должна поддерживать репликацию, шардирование.

Нереляционная СУБД MongoDB [12] полностью отвечает приведенным выше требованиям.

Выбор языка разработки компонент портала

Проанализируем техническое задание на разработку портала. Исходя из приведенных требований к системе, можно выявить требования к языку программирования:

* **Наличие разнообразных библиотек**. Использование готовых библиотек ускоряет разработку программного обеспечения. Также важно, что благодаря использованию распространенных оттестированных библиотек, снижается вероятность ошибки. Это повышает надежность программного обеспечения. К этому же требованию можно отнести наличие стандартных средств управления библиотеками.
* **Совместимость с выбранными ранее технологиями**. Выбранный язык должен уметь взаимодействовать с ОС Linux, СУБД MongoDB.
* **Высокая скорость разработки.** На ранних этапах разработки портала технические требования часто меняются. Язык программирования должен позволять как можно быстрее вносить изменения в коды программ . Это позволяет успешнее конкурировать с аналогичными сервисами.

Под данные требования хорошо подходит язык JavaScript [15]. Современный JavaScript – это «безопасный» язык программирования общего назначения. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально был ориентирован на браузеры, в которых это не требуется. Относится к классу языков с динамической типизацией. JavaScript обладает мощной стандартной платформой для разработки серверных решений NodeJs. Под этот язык существует хорошо развитая инфраструктура: менеджеры пакетов (npm), различные фреймворки (Express, Mongoose, Yoeman) для разработки веб-приложений. JavaScripy активно развивается.

По сравнению с C/C++ этот язык избавляет от ручного управления памятью. Но при этом скорость его работы ниже.

По сравнению с Java/C# язык JavaScript избавляет от необходимости перекомпиляции проекта после каждого изменения. Также JavaScript позволяет вести разработку быстрее за счет высокоуровневости языка, а наличия большого числа библиотек для разработки серверных приложений дает ему дополнительные конкурентные преимущества.

Важными преимуществами языка JavaScript являются возможность создавать на нем кроссплатформенные приложения. Кроме того, он является свободно распространяемым языком программирования, что также является большим преимуществом.

Выбранный язык позволяет вести эффективную разработку, а богатая платформа NodeJs позволяет писать безопасное программное обеспечение.

Выбор фреймворка для разработки портала

К выбору фреймворка в нашей работе предъявляются те же требования, что и к выбору языка программирования:

* большое число стандартных возможностей;
* совместимость с выбранными ранее технологиями;
* высокая скорость разработки.

Для разработки системы «Портал для мастеров и клиентов в сфере бытовых услуг» будем использовать фреймворки, основанные на парадигме MVC (Мodel-View-Controller, модель - вид - элемент управления):

* модель представляет объекты, хранимые в системе: информацию о пользователях, данные платежных документов, информацию об активных сессиях;
* вид отвечает за визуализацию моделей. К примеру, для мобильной версии портала и для портала, ориентированного на ПК, могут применяться разные варианты отображения;
* элемент управления отвечает за взаимодействие пользователя и программного обеспечения.

Express — свободно распространяемый фреймворк для разработки серверных приложений на языке JavaScript. Фреймворк Express позволяет разрабатывать серверные приложения по парадигме MVC. В ***Модели*** прописываются классы для ORM (Object-Relation Mapping),которые позволяют связывать объекты классов с сущностями базы данных. В представлении (View) прописывается то, как отображать модель пользователь (обычно это HTML/Jade шаблоны). Элемент управления (Controller) должен содержать в себе логику каждой конкретной страницы.

Среди аналогов Express стоит отметить koajs, но его возможности сильно ограничены по сравнению с Express.

По сравнению с аналогичными фреймворками Express обладает следующими преимуществами:

* проработанная документация (существует и на русском языке);
* гибкая настройка проекта
* проработанная система обработки запросов.
* разнообразие возможностей.

В результате проведенного анализа для разработки портала был выбран

фреймворк Express.

Обеспечение надежности портала

В данной системе «Портал для аренды автомобилей» для фронтенда и бекендов целесообразно применить экранирование связи между сервисами. Это обеспечит чистоту каналов системы: в случае попытки отправить запрос к какому-либо сервису в обход фронтенда, запрос будет проигнорирован.