По памяти:

Ряд алгоритмов требует выделения дополнительной памяти под временное хранение данных. Как правило, эти алгоритмы требуют $O(\log n)$ памяти. При оценке не учитывается место, которое занимает исходный массив и независящие от входной последовательности затраты, например, на хранение кода программы (так как всё это потребляет $O(1)$). Алгоритмы сортировки, не потребляющие дополнительной памяти, относят к сортировкам на месте.

ЛИТЕРАТУРА

1. Алгоритм сортировки [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Алгоритм_сортировки – Дата доступа 05.04.2018

УДК 004.42

Студ. А.Д. Самаль
Науч. рук. доц. Н.В. Пацей
(кафедра программной инженерии, БГТУ)

СИСТЕМА БРОНИРОВАНИЯ БИЛЕТОВ МАРШРУТНОГО ТАКСИ НА ОСНОВЕ МОДЕЛИ ОБСЛУЖИВАНИЯ SAAS

SaaS расшифровывается как software as a service — программное обеспечение как услуга. SaaS — это модель предоставления лицензии на программное обеспечение по подписке. Чаще всего такое ПО — это облачное решение, т. е. находящееся на серверах в интернете.

Пользователь получает доступ к сервису через браузер или по API. При этом его поддержкой целиком занимается поставщик услуги. Упрощенно говоря, модель SaaS это когда клиент работает с готовым решением онлайн. Оплачивает доступ и максимально быстро получает на руки готовый инструмент.

Самый простой пример SaaS — это Google Docs, бесплатный сервис для работы с документами. Никаких носителей, драйверов и установок. Заводите аккаунт Google, переходите по ссылке и работаете с текстами, таблицами и презентациями прямо в браузере. Причем в документах одновременно с вами могут работать и другие коллеги. Уехав в командировку, можно зайти в свой аккаунт с любого устройства и продолжить печатать нужный документ. При этом сохранять нужно только изменения настроек, остальные данные сохраняются автоматически [1].

По сути, SaaS-сервис это единое программное ядро, которое предоставляется в пользование клиентам. Доступ к системе они получают через сеть и могут изменять настройки на свое усмотрение. Обслуживанием сервиса целиком занимается провайдер услуги, а пользователь только работает в ней.

Популярность этой модели ежегодно растет. Только в прошлом году рынок SaaS увеличился на 21,7%. По прогнозам некоторых экспертов, такая тенденция может сохраниться и в ближайшие годы. Это и не удивительно, ведь так пользователи получают в распоряжение современные технологии практически без усилий со своей стороны [1].

Перед реализацией любой системы необходимо иметь четкое представление о том, как она будет работать, о ее структуре. Для этого была разработана общая схема работы веб-приложения, которая позволит наиболее полно понять, из каких компонентов состоит система, и каким образом они взаимодействуют между собой. Данная схема представлена на рисунке 1.

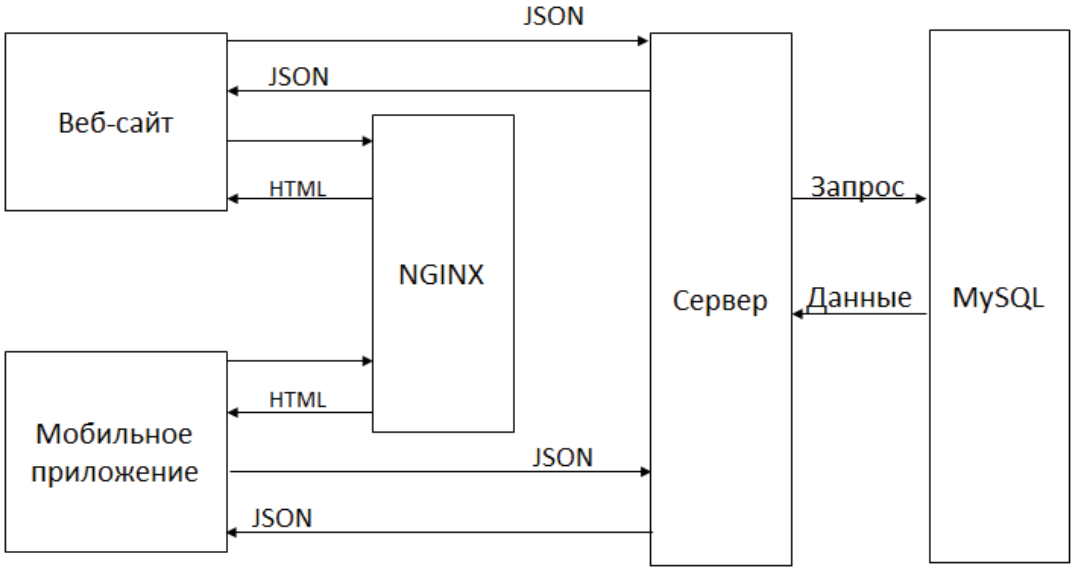

Рисунок 1 – Общая схема работы системы

Система состоит из основного сервера, базы данных, сервера для управления статическими данными (NGINX), а так же двух клиентских приложений – веб-сайт и мобильное приложение. Обращение к серверу происходит посредством API, а в качестве транспорта данных используется формат передачи данных JSON.

Каждой новой зарегистрированной компании будет выделяться поддомен на основном сайте. Отвечать за перенаправление на поддомены будет так же отвечать NGINX.

В качестве основного фреймворка для реализации серверной части используется SpringBoot 2 совместно с Jersey. Jersey является одной из реализаций спецификации JAX-RS (спецификация предназначенная для создания RESTful веб-сервисов). А SpringBoot это «фреймворк для фреймворка», который создан, чтоб упростить задачу разработчикам, которые используют Spring. Поскольку ранее при разработке приложения, имелась необходимость создавать множество файлов конфигураций в формате XML, а SpringBoot автоматически сканирует проект и сам настраивает нужные конфигурационные параметры. Так же при необходимости все параметры разработчик может изменить в своем файле параметров, и SpringBoot будет начинать инициализацию вместе с этим файлом.

Для хранения всех данных пользователей была выбрана СУБД MySQL. Основными ее плюсами является то, что она полностью бесплатна и достаточно быстрая с большими объемами информации. Схема базы данных представлена на рисунке 2.

При проектировании базы данных для SaaS приложений много внимания стоит уделить вопросу безопасности, а именно как изолировать данные одних пользователей от других. Или иными словами – реализовать мультитенантность. Ведь трудно убедить клиента, что его личные данные в безопасности, если клиент знает, что приложение физически используется совместно с другими клиентами [2-3]. Всего существует четыре подхода для реализации мультитенантности:

- Построение баз данных, где таблицы каждого клиента находятся в отдельной схеме.
- средства безопасности базы данных, которые позволяют использовать механизмы контроля доступа на уровне базы данных.
- партиционирование для изоляции данных клиента.
- сочетание всех выше перечисленных подходов.

Таким образом, можно сделать вывод, что мультитенантность должна быть продумана как с точки зрения бизнеса, так и с технической стороны. Кроме того, мультитенантное SaaS-приложение позволяет сделать снизить стоимость решения для клиентов, т. е. максимально повысить его конкурентно способность.

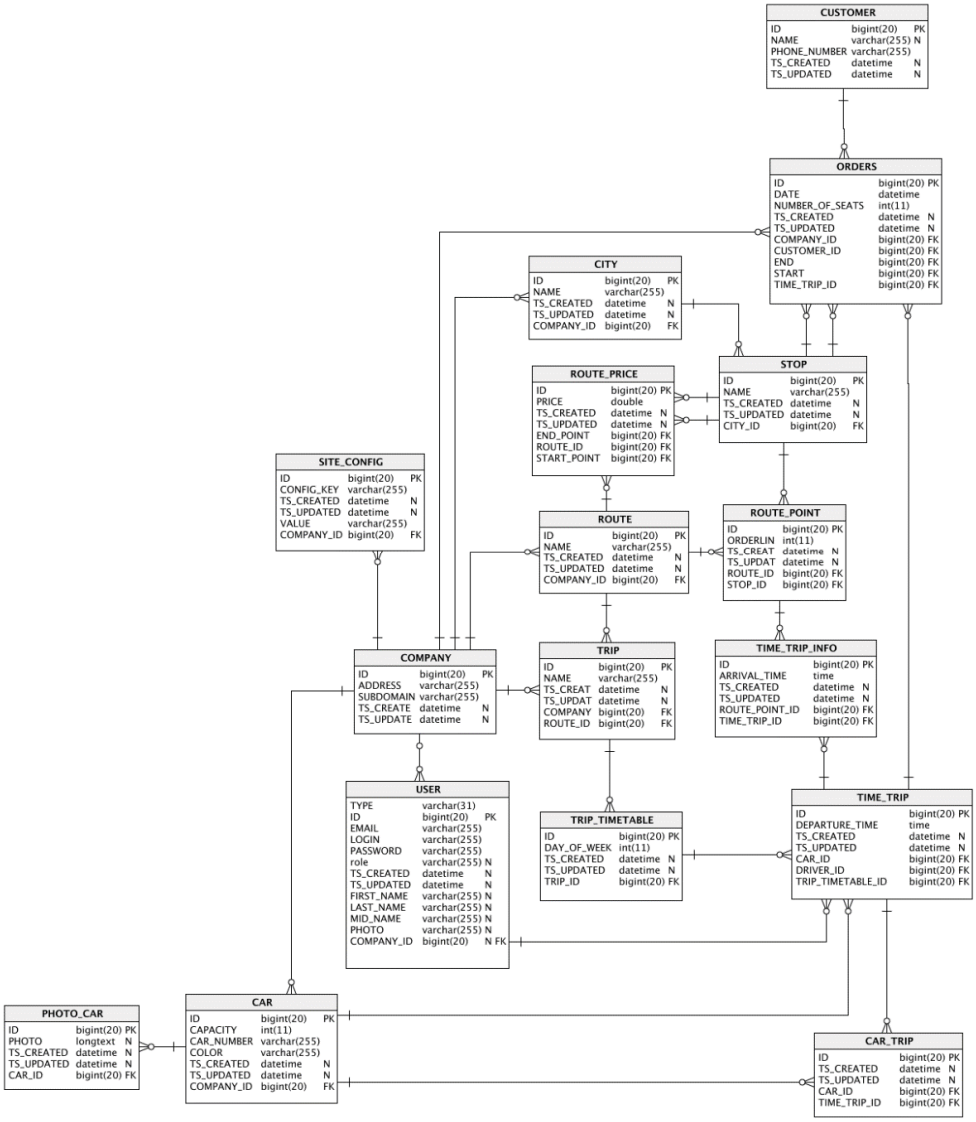

Рисунок 2 – Схема базы данных

ЛИТЕРАТУРА

1. What is SaaS. [Электронный ресурс] - Режим доступа: https://azure.microsoft.com/en-us/overview/what-is-saas/ (дата обращения 10.04.2018).

2. Мультитенантная архитектура для SaaS приложений. [Электронный ресурс] - Режим доступа: https://habrahabr.ru/company/microsoft/blog/145027/ (дата обращения 04.04.2018).

3. Multi-tenancy in the cloud: Why it matters. [Электронный ресурс] - Режим доступа: https://www.computerworld.com/ article/2517005/data-center/multi-tenancy-in-the-cloud--why-it-matters.html (дата обращения 09.04.2018).