Рубрика: Информатизация на предприятиях ДФО
УДК 004.415.2

Проектирование и разработка REST API для предприятия АО «Авиакомпания «Аврора»

Гришин Владислав Александрович
бакалавр 4 курса

Владивостокский государственный университет экономики и сервиса
г. Владивосток

Работа посвящена проблеме информатизации авиационной компании. Рассматривается задача создания REST API, который должен стать фундаментом для перехода на микросервисную архитектуру.

***Ключевые слова и словосочетания**: разработка REST API, масштабирование информационных систем, Управление правами доступа в веб-сервисах, автоматизация бизнес-процессов*

Designing and developing Design and development of REST API for the enterprise AO «Aviakompaniya «Avrora»

The work is dedicated to the issue of informatization of an aviation company. It examines the task of creating a REST API, which is intended to become the foundation for transitioning to a microservice architecture.

***Key words:** REST API development, Python, Django, information system scalability, web service access rights management, business process automation.*

В эпоху быстрого развития информационных технологий и цифровой трансформации бизнес-процессов авиационная отрасль сталкивается с рядом вызовов и задач, требующих современных и эффективных решений. АО "Авиакомпания "Аврора", в рамках стремления к повышению оперативности учета материальных ценностей и оптимизации ресурсного планирования, поставила перед собой задачу разработки REST API как основы для построения микросервисной архитектуры. Данный проект не только направлен на решение текущих задач учета и управления, но и открывает перспективы для масштабирования и интеграции с новыми сервисами, что является ключевым аспектом в условиях быстро меняющегося рынка [1].

Проект разработки REST API инициирован на фоне актуальных потребностей авиакомпании в унификации и автоматизации процессов учета, особенно в контексте использования планшетных компьютеров iPad среди пилотного состава и бортпроводников. Использование устаревших методов учета, отсутствие единой информационной системы приводило к увеличению времени на подготовку и анализ данных, что существенно снижало оперативность принятия управленческих решений и, как следствие, эффективность работы компании в целом. Одной из ключевых проблем, с которой сталкивалась "Авиакомпания "Аврора", является ее зависимость от сторонних информационных систем, в том числе от систем, предоставляемых Группой "Аэрофлот". Такая зависимость значительно ограничивала возможности компании в плане добавления собственного, специфичного для ее бизнес-процессов функционала. В большинстве случаев использование приобретенных или корпоративных информационных систем не предусматривает гибкость в настройке и адаптации под уникальные потребности отдельного предприятия, что вынудило "Авиакомпанию "Аврора" искать собственные пути решения данных задач. Разработка собственного REST API стала стратегическим решением, направленным на преодоление этих ограничений и создание условий для будущего масштабирования

и развития информационной инфраструктуры компании.

В процессе анализа процессов учёта была выявлена потребность в разработке собственной информационной системы, способной автоматизировать учет планшетных компьютеров, а также интегрировать данные с другими системами и сервисами компании. Для достижения новых уровней оперативности и гибкости было принято решение развивать проект в направлении создания REST API и перехода к микросервисной архитектуре.

Разработка REST API для АО "Авиакомпания "Аврора" является ответом на вызовы современности, предоставляя компании инструменты для более эффективного управления ресурсами, ускорения обработки и анализа данных. В рамках данной статьи будет рассмотрена, архитектура и ключевые особенности созданного API, а также его роль в дальнейшем развитии информационной инфраструктуры авиакомпании.

Также в ходе анализа текущей ситуации на предприятии и учета потребностей персонала АО "Авиакомпания "Аврора", руководством было вынесено стратегическое решение о разработке индивидуализированных пользовательских интерфейсов в рамках нового REST API. Это решение направлено на оптимизацию информационного потока и повышение эффективности работы каждого сотрудника за счет учета их роли и задач в компании. Спроектированные интерфейсы, включая административную панель на базе Django и клиентскую часть на React, предназначены для того, чтобы предоставить каждому работнику доступ к необходимым данным и функционалу, исключая при этом элементы, не относящиеся к его компетенции или задачам. Такой подход позволит сделать работу с системой максимально интуитивно понятной и эффективной даже для новых пользователей, способствуя быстрой адаптации и минимизации времени на обучение [2].

Таким образом, разработка не просто улучшит текущие процессы учета и управления ресурсами, но и заложит основу для масштабируемой и гибко настраиваемой информационной системы, отвечающей будущим потребностям авиакомпании.

Есть множество разновидностей API, но для этой задачи больше всего подходит REST (также известна как RESTful) API, особенностью которой является использование стандартных методов HTTP для обмена данными между клиентом и сервером. REST API строится на принципах представительного состояния и передачи (Representational State Transfer), что подразумевает легкость в интеграции и высокую масштабируемость [3]. Основные особенности REST API включают в себя:

– отсутствие состояний (Statelessness): в REST каждый запрос считается независимым, и сервер не сохраняет информацию о состоянии клиента между запросами, что упрощает архитектуру и повышает надежность системы;

– единообразие интерфейса (Uniform Interface): REST API предоставляет стандартизированный способ взаимодействия, что облегчает разработку и интеграцию различных систем и приложений;

– использование стандартных HTTP-методов: Таких как GET для получения данных, POST для создания новых записей, PUT для обновления существующих и DELETE для их удаления. Это делает API интуитивно понятным и легким в использовании;

– система представлений ресурсов: В REST API данные представляются в виде ресурсов, к которым можно обращаться через URL. Ресурсы могут быть представлены в различных форматах, таких как JSON или XML.

Основным языком разработки стал Python из-за того, что на нём разработана большая часть технических решений в авиакомпании, а также из-за мощных инструментов разработки REST API. Главным инструментов разработки был выбран веб-фреймворк под названием Django Rest Framerowk, который позволяет с большой скоростью создать REST API, а также имеет в себе множество инструментов, которые помогают в создании типовых, для учётных задач, функций и интерфейсов.

Благодаря использованию HTTP запросов удастся реализовать полную интеграцию и синхронизацию между всеми веб-сервисами авиакомпании при помощи единого REST API, который будет принимать и обрабатывать все направленные запросы.

Для того чтобы определить функциональные требования, предъявляемые к системе, необходимо, прежде всего, выявить лиц, заинтересованных в этой системе, а затем определить тот функционал, который им требуется для осуществления своей деятельности. Модель прецедентов описывает желаемое поведение системы (ее функционал) с точки зрения пользователя.

Пользователями приложения будут являться исключительно сотрудники компании АО «Авиакомпания «Аврора». Пользователей при этом можно разделить на три группы:

– сотрудники СИТ – сотрудники, которые будут администрировать систему добавляя новые записи, а также изменяя или удаляя уже существующие;

– сотрудники склада – другие сотрудники, которые используют платформу для просмотра записей с целью последующего экспортирования в свои системы для анализа или дальнейшей работы;

– пилоты и бортпроводники.

Для каждой группы пользователей было принято решение разграничить доступ ко всей информации независимо от части микросервиса с которой они пытаются получить к ней доступ.

Был принят следующий принцип разграничения прав пользователей по группам:

– Admin – полный набор прав, доступ к административной панели;

– Editor – отсутствие доступа к административной панели, возможность отправлять только PATCH запросы с целью изменения уже созданных записей;

– Viewer – отсутствие всех прав, кроме просмотра списка всех записей (метод GET);

– OPP – зарезервированная группа для будущих пользователей.

Финальную схему, отображающую пользователей и иерархию ролей можно увидеть на рисунке 1.

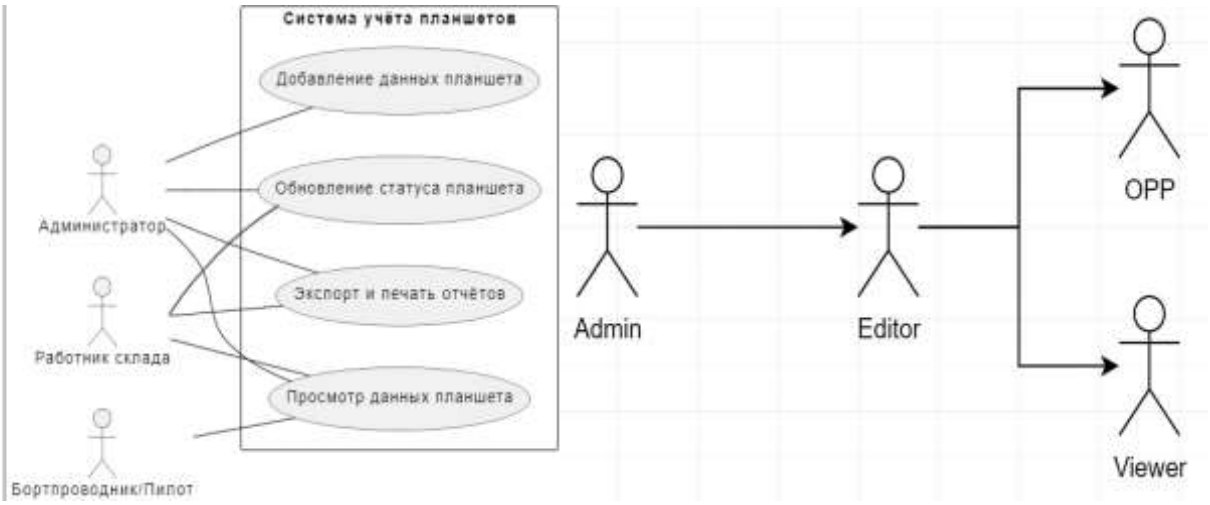

Рис. 1. Схема групп пользователей и иерархии ролей

Роли пользователей распределены иерархично от администратора, который имеет права на любое редактирований, до обычного пользователя, который имеет права только на первичный просмотр общей информации.

Также независимо от роли пользователь никогда не будет иметь прямого доступа к базе данных из-за принципа работы REST API, который является некой прослойкой между клиентской частью веб-сервиса и базой данных.

Принцип связи клиентской стороны с базой данных при помощи REST API проиллюстрирован на рисунке 2.

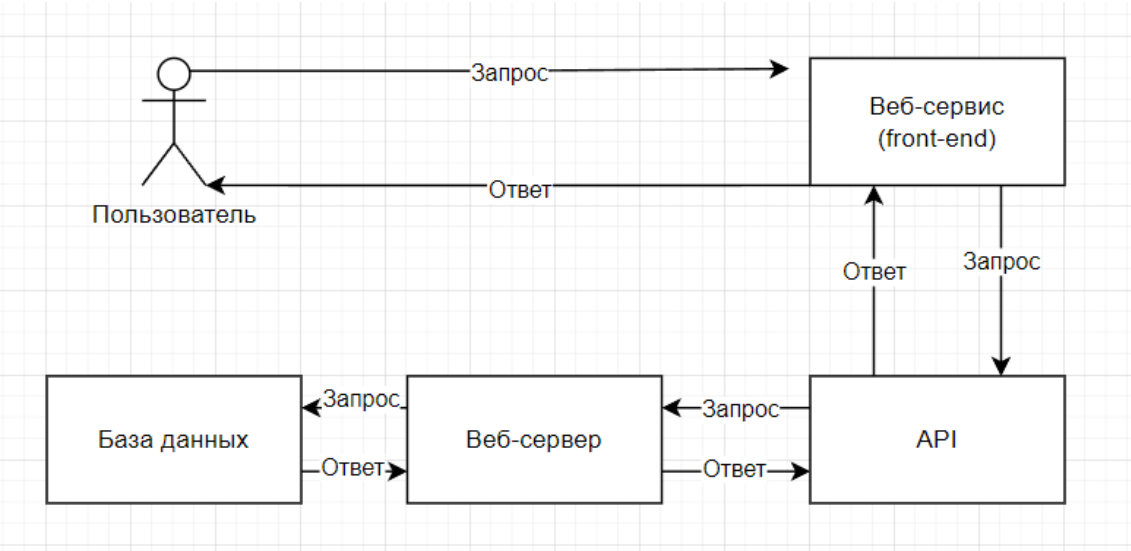

Рис. 2. Визуализация связи веб-сервиса и базы данных

REST API, настроенный для обработки GET, POST, и PATCH запросов, служит основой для гибкого взаимодействия между клиентом и сервером в современных веб- и мобильных приложениях. Это обеспечивает стандартизированный способ обмена информацией, где каждый тип запроса выполняет уникальную роль в управлении данными.

GET запросы используются для получения данных. Когда пользователь взаимодействует с веб-сервисом или мобильным приложением, система может отправить GET запрос к REST API для извлечения необходимой информации. Эта информация затем может быть отображена пользователю. Принцип работы заключается в том, что API интерпретирует запрос, обращается к базе данных или другому источнику данных, извлекает требуемые данные и отправляет их обратно клиенту в формате, например, JSON или XML.

POST запросы применяются для создания новых данных. Когда пользователь вводит информацию через интерфейс приложения, например, заполняя форму регистрации или добавляя новый элемент в список, приложение отправляет POST запрос к API для сохранения этой информации в базе данных. REST API обрабатывает запрос, создает новую запись в базе данных с предоставленными данными и часто возвращает идентификатор созданной записи или другие подтверждающие данные.

PATCH запросы предназначены для обновления существующих данных. Этот тип запроса используется, когда необходимо модифицировать часть данных без необходимости замены всей записи. Например, если пользователь обновляет свой профиль, приложение может отправить PATCH запрос с измененными данными. API тогда найдет соответствующую запись в базе данных и обновит только указанные поля, оставив остальные данные без изменений.

Этот процесс позволяет приложениям эффективно взаимодействовать с сервером, минимизируя количество передаваемых данных и обеспечивая актуальность информации без избыточных запросов или передачи ненужных данных. Благодаря четко определенным методам

HTTP, REST API предлагает стандартизированный и интуитивно понятный способ управления данными, что делает разработку более предсказуемой и упрощает интеграцию различных сервисов и платформ.

Пользователь ни в один из моментов использования любой части веб-сервиса не получает прямого контакта к базе данных. При каждом отправленном запросе пользователем происходит несколько этапов валидации прохождение которых требуется для успешной записи новой информации в базу данных

Принцип связи клиентской стороны с базой данных через REST API включает несколько ключевых моментов, которые делают этот метод предпочтительным для современных веб-приложений и микросервисов. REST API действует как посредник между клиентом (например, веб-приложением или мобильным приложением) и сервером, где хранятся данные, таким образом обеспечивая безопасность, масштабируемость и гибкость в взаимодействии [4].

Также возможность логирования всех запросов к REST API является критически важной функцией для обеспечения безопасности, отладки и анализа данных. Логирование позволяет администраторам и разработчикам отслеживать все входящие и исходящие запросы, анализировать поведение системы и быстро реагировать на любые аномалии или ошибки. Каждый запрос к API может быть зарегистрирован с указанием времени запроса, IP-адреса клиента, типа запроса (например, GET, POST, PATCH), запрашиваемого ресурса, статуса ответа и другой сопутствующей информации.

Логирование предоставляет полную картину взаимодействия клиентов с системой, что особенно полезно в многопользовательских и масштабируемых приложениях, где отслеживание каждого действия пользователя может быть критично для поддержания высокого уровня производительности и безопасности [5].

В заключение, проект разработки REST API, инициированный АО "Авиакомпания "Аврора", ставит перед собой амбициозные цели по автоматизации и оптимизации учетных процессов и ресурсного планирования. Это не только позволит компании повысить оперативность и эффективность управленческих решений, но и обеспечит гибкость в интеграции с новыми сервисами и технологиями. Разработка индивидуализированных пользовательских интерфейсов в рамках нового REST API подчеркивает стремление компании к учету уникальных потребностей каждого сотрудника и оптимизации информационного потока. Таким образом, "Авиакомпания "Аврора" не только решает текущие задачи, но и заложит основу для будущего развития и масштабирования своей информационной системы, что является ключевым фактором успеха в быстро меняющемся мире авиационной индустрии

---

1. How to Build Microservices with REST APIs? [Электронный ресурс] // Talend.com – Режим доступа: https://www.talend.com/resources/how-to-build-microservices-rest-apis/ (Дата обращения: 08.04.2024).
2. Проектирование и разработка интерфейсов пользователя: что нужно знать. [Электронный ресурс] // Codernet.ru – Режим доступа: https://codernet.ru/articles/drugoe/proektirovanie_i_razrabotka_interfejsov_polzovatelya_chto_nuzhno_znat/ (Дата обращения: 08.04.2024).
3. Architectural Styles and the Design of Network-based Software Architectures / Fielding Roy Thomas // Университет Калифорнии, Ирвайн – Режим доступа: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm (Дата обращения: 09.04.2024).
4. What Is a REST API? Examples, Uses, and Challenges. [Электронный ресурс] // Blog.postman.com – Режим доступа: https://blog.postman.com/rest-api-examples/ (Дата обращения: 09.04.2024).
5. 10 REST API Logging Best Practices / Rayna Thomasson // ClimbTheLadder.com – Режим доступа: https://climbtheladder.com/10-rest-api-logging-best-practices/ (Дата обращения: 09.04.2024).