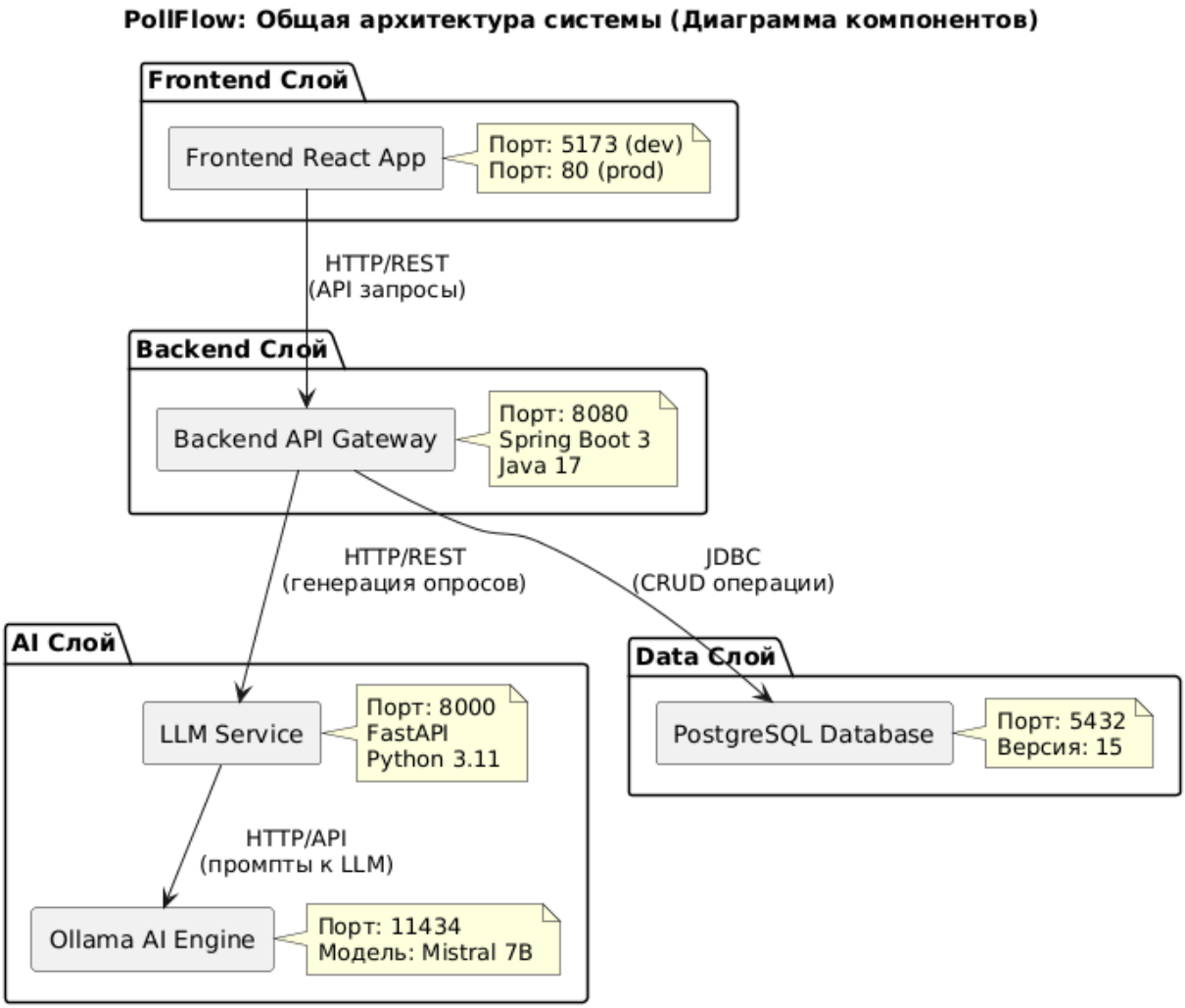


Схема демонстрирует высокоуровневую архитектуру системы, определяя ключевые компоненты и протоколы их взаимодействия.

- **Клиентское приложение (Frontend)**, функционирующее на порту 5173, реализует пользовательский интерфейс. Данный компонент обеспечивает интерактивное взаимодействие конечного пользователя с функционалом системы посредством веб-браузера.
- **Серверное приложение (Backend)**, развернутое на порту 8080, исполняет роль центрального координационного узла. Оно ответственно за обработку бизнес-логики, управление сессиями и маршрутизацию запросов между различными сервисами.
- **Система управления базами данных PostgreSQL** (порт 5432) выполняет функцию долговременного хранилища структурированных данных. Включает в себя учетные записи пользователей, метаданные опросов, формулировки вопросов и вариантов ответов, а также результаты их прохождения. Прямой доступ к данному компоненту имеет исключительно серверное приложение.
- **Микросервис LLM** (порт 8000) и **интеллектуальная модель Ollama** (порт 11434) составляют интегрированный модуль генеративного искусственного интеллекта. Их совместная работа направлена на автоматическое создание структурированного контента для опросов на основе заданных тематических параметров.

Взаимодействие компонентов осуществляется по принципу четкого разделения ответственности. Пользовательский интерфейс инициирует операции, которые обрабатываются серверным ядром. Последнее, в зависимости от характера операции, либо осуществляет запросы к базе данных, либо делегирует задачи генерации контента специализированному AI-сервису.

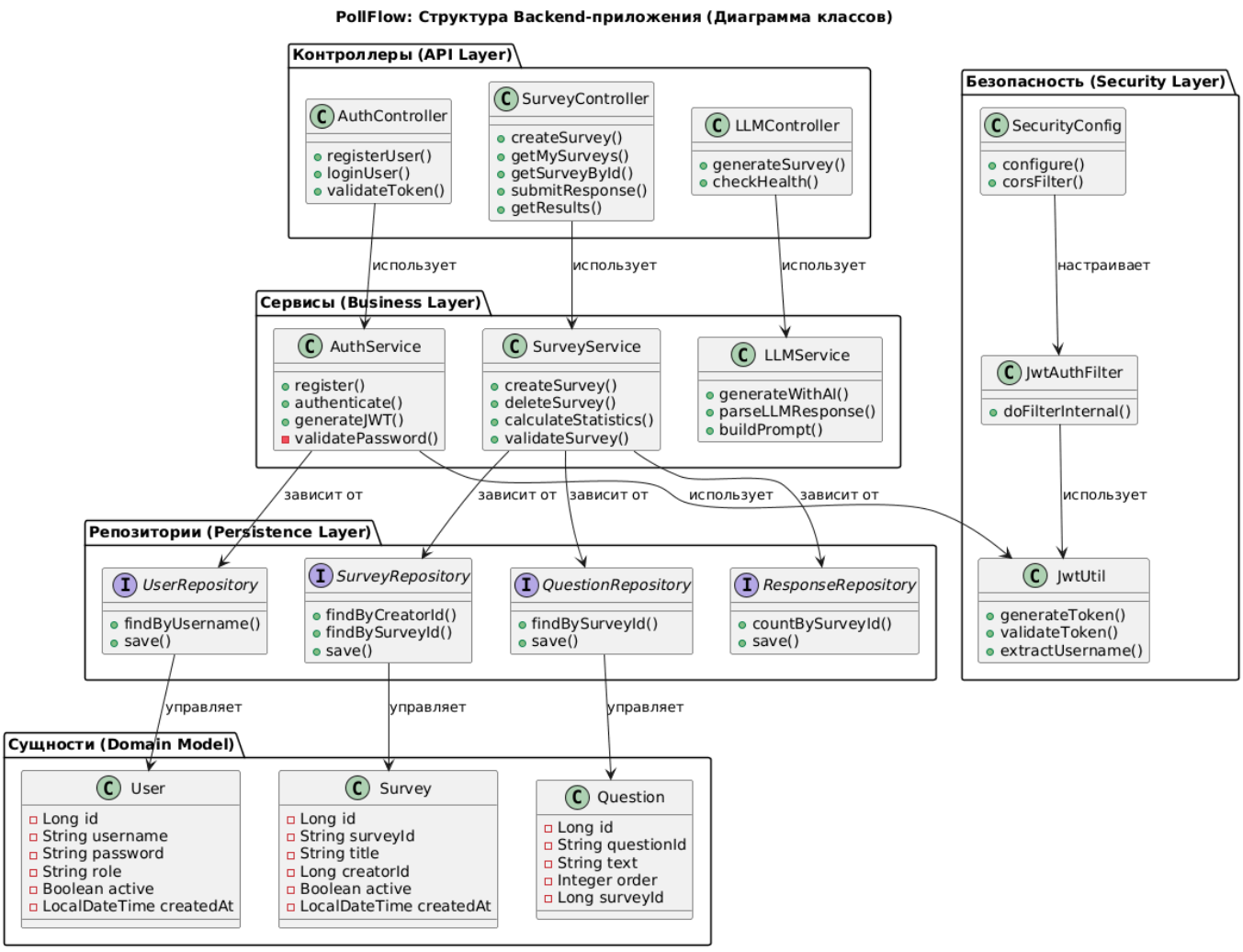


Схема детализирует логическую организацию серверного компонента, построенного на основе фреймворка Spring Boot.

- **Слой контроллеров (Controller Layer)** выступает в качестве точки входа для HTTP-запросов. Контроллеры (AuthController, SurveyController, LLMController) ответственны за прием запросов, первичную валидацию формата данных и передачу их на обработку бизнес-логике.
- **Слой сервисов (Service Layer)** содержит основную бизнес-логику приложения. Классы AuthService, SurveyService, LLMService реализуют конкретные алгоритмы: аутентификацию пользователей, управление жизненным циклом опросов, расчет аналитики и взаимодействие с внешними AI-сервисами.
- **Подсистема безопасности (Security Subsystem)**, основанная на JWT-токенах, обеспечивает защиту API. Компоненты JwtAuthFilter и SecurityConfig осуществляют сквозную проверку авторизации и настройку политик доступа до передачи запроса контроллерам. Утилита JwtUtil предоставляет методы для работы с токенами.
- **Слой репозиториев (Repository Layer)** абстрагирует доступ к данным. Интерфейсы (UserRepository, SurveyRepository и др.) инкапсулируют операции с объектами предметной области в базе данных, следуя шаблону Data Access Object (DAO).

Обработка запроса представляет собой последовательный конвейер. После успешной проверки безопасности в JwtAuthFilter запрос поступает в соответствующий контроллер, который вызывает необходимый сервис. Сервис, для выполнения своей задачи, использует один или несколько репозиториев для извлечения или сохранения данных. Результат возвращается по цепочке обратно клиенту.

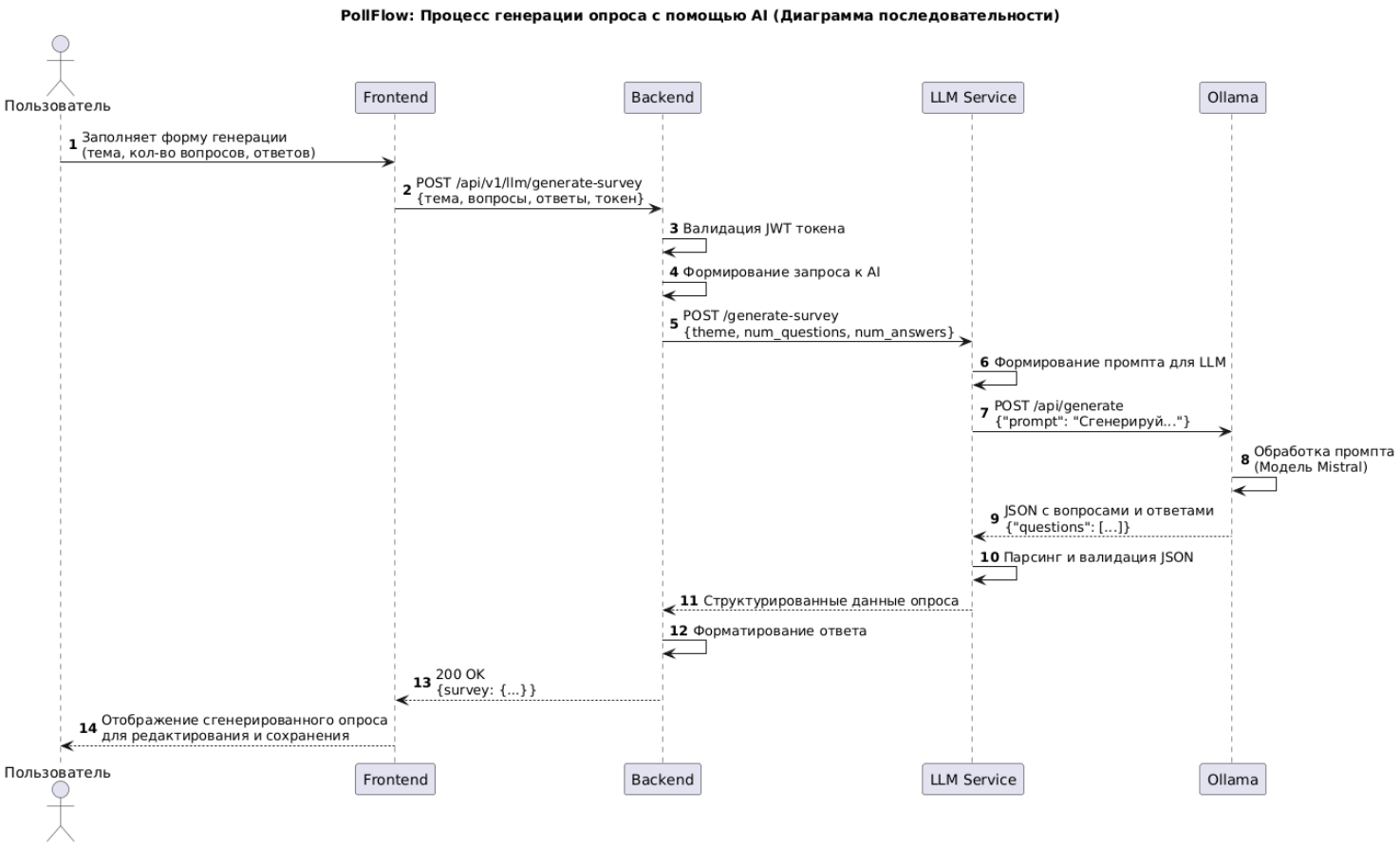
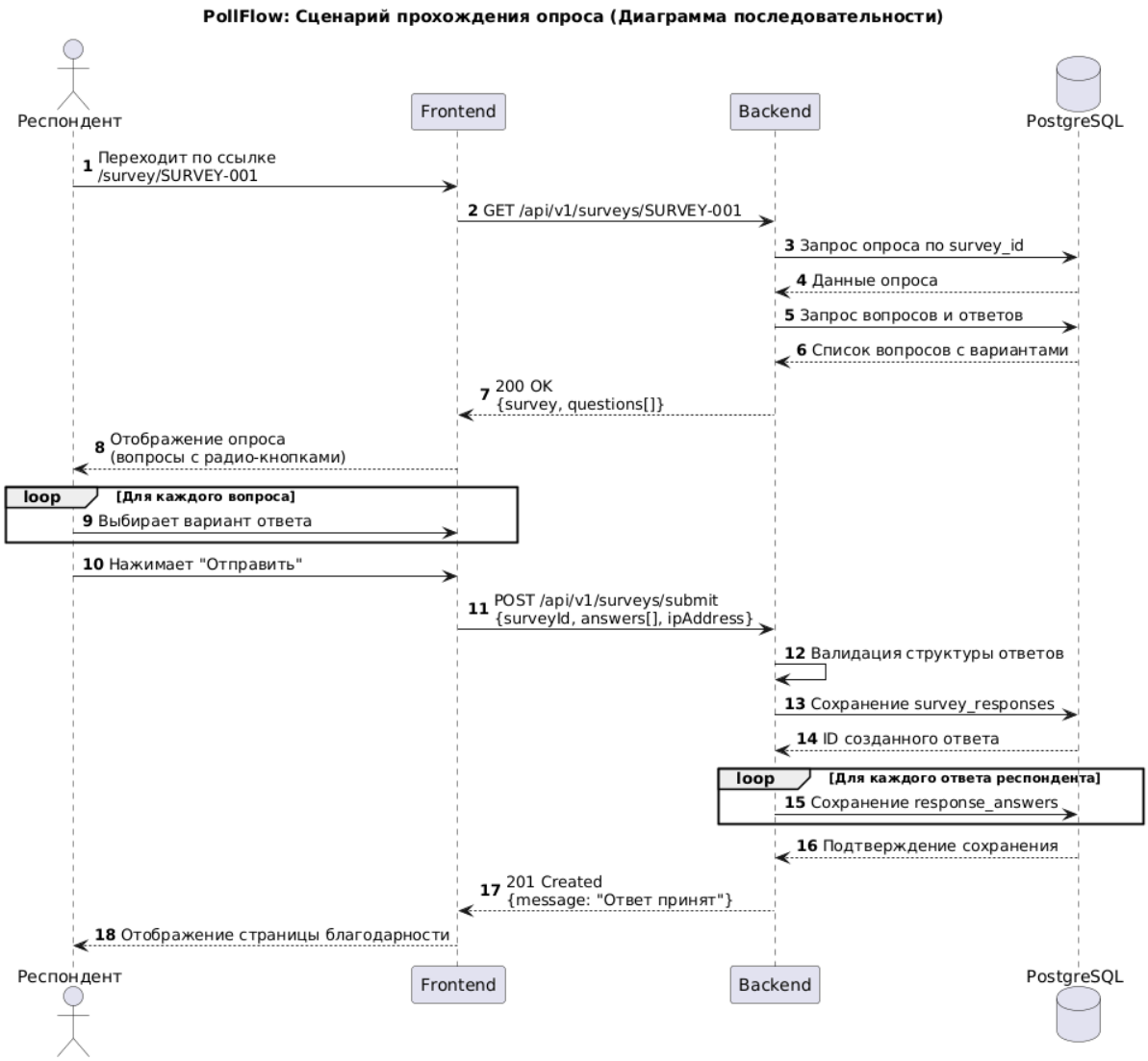


Схема описывает последовательность действий при использовании функции автоматического создания анкеты на основе генеративных нейросетей.

- **Микросервис LLM (FastAPI)** выполняет функции адаптера и парсера. Он получает от основного сервера структурированный запрос, преобразует его в специализированный текстовый промпт, пригодный для обработки языковой моделью, а затем выполняет синтаксический разбор (парсинг) полученного от ИИ ответа в согласованный формат (JSON).
- **Движок Ollama с моделью Mistral** является ядром генерации. Принимая текстовый промпт (например: *«Сгенерируй 5 вопросов о корпоративной культуре с 4 вариантами ответов в каждом»*), модель продуцирует семантически связанный текст, соответствующий инструкции.

Исходные параметры от пользователя поступают на **Backend-сервер**, который перенаправляет их в **LLM-сервис**. Тот формирует промпт и отправляет его в **Ollama**. Сгенерированный языковой моделью текст возвращается в **LLM-сервис** для парсинга, после чего структурированные данные передаются обратно на **Backend-сервер** и, наконец, отображаются в интерфейсе **пользователя** для последующего редактирования и сохранения.



Диаграммы последовательности иллюстрируют полные циклы взаимодействия между компонентами архитектуры для реализации ключевых пользовательских сценариев.

- **Создание опроса вручную:** Сценарий детализирует процесс от нажатия кнопки сохранения в интерфейсе до фиксации данных в СУБД. Ключевым элементом является передача JWT-токена для авторизации запроса и атомарное сохранение связанных данных в таблицы surveys, questions и answers.
- **Генерация опроса средствами ИИ:** Сценарий расширяет общую архитектурную схему, демонстрируя цепочку синхронных вызовов от backend-сервера через LLM-сервис к AI-движку и обратно, подчеркивая асинхронный характер операции генерации.
- **Анонимное прохождение опроса:** Данный сценарий подтверждает, что функционал участия в опросе не требует предварительной аутентификации. Респондент получает доступ к форме по прямой ссылке, а его ответы сохраняются с привязкой только к идентификатору опроса, обеспечивая анонимность.
- **Анализ результатов:** Сценарий показывает процесс преобразования первичных данных из таблицы response_answers в агрегированную аналитическую сводку. Серверное приложение выполняет расчет статистических показателей (процентное распределение ответов) перед отправкой результата пользователю.

Представленные сценарии операционализируют статическую архитектуру, наглядно демонстрируя различные потоки данных и особенности взаимодействия компонентов в зависимости от типа инициируемой пользовательской операции.

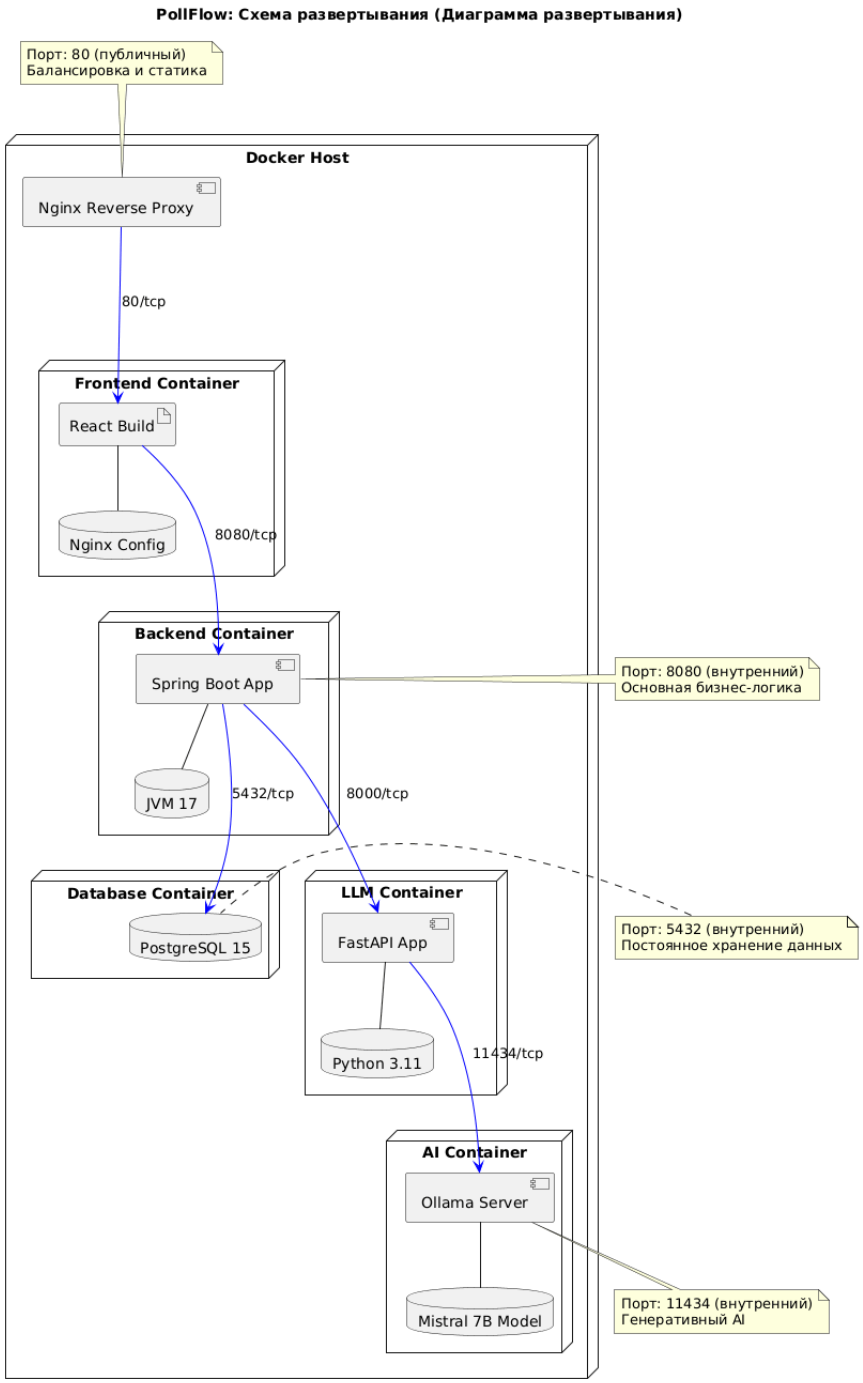


Схема представляет собой классификацию механизмов защиты, реализованных в системе.

- Механизмы разделены по уровням воздействия: аутентификация и авторизация (JWT), криптографическая защита данных (BCrypt), контроль доступа на уровне транспорта (CORS) и предотвращение атак типа «отказ в обслуживании» (Rate Limiting).
- **JWT (JSON Web Tokens)** обеспечивает stateless-аутентификацию, где каждый защищенный запрос должен содержать валидный токен.
- **BCrypt** применяется для одностороннего хеширования паролей перед их сохранением в базе данных, исключая хранение чувствительных данных в открытом виде.
- **CORS (Cross-Origin Resource Sharing)** политики ограничивают домены, с которых разрешены запросы к API.
- **Rate Limiting** ограничивает частоту запросов с одного клиентского адреса для обеспечения стабильности работы сервиса.

Подсистема безопасности, в частности JwtAuthFilter, функционирует как сквозной (cross-cutting) модуль, осуществляя проверку до того, как запрос достигнет слоя бизнес-логики, что соответствует принципу defense in depth.

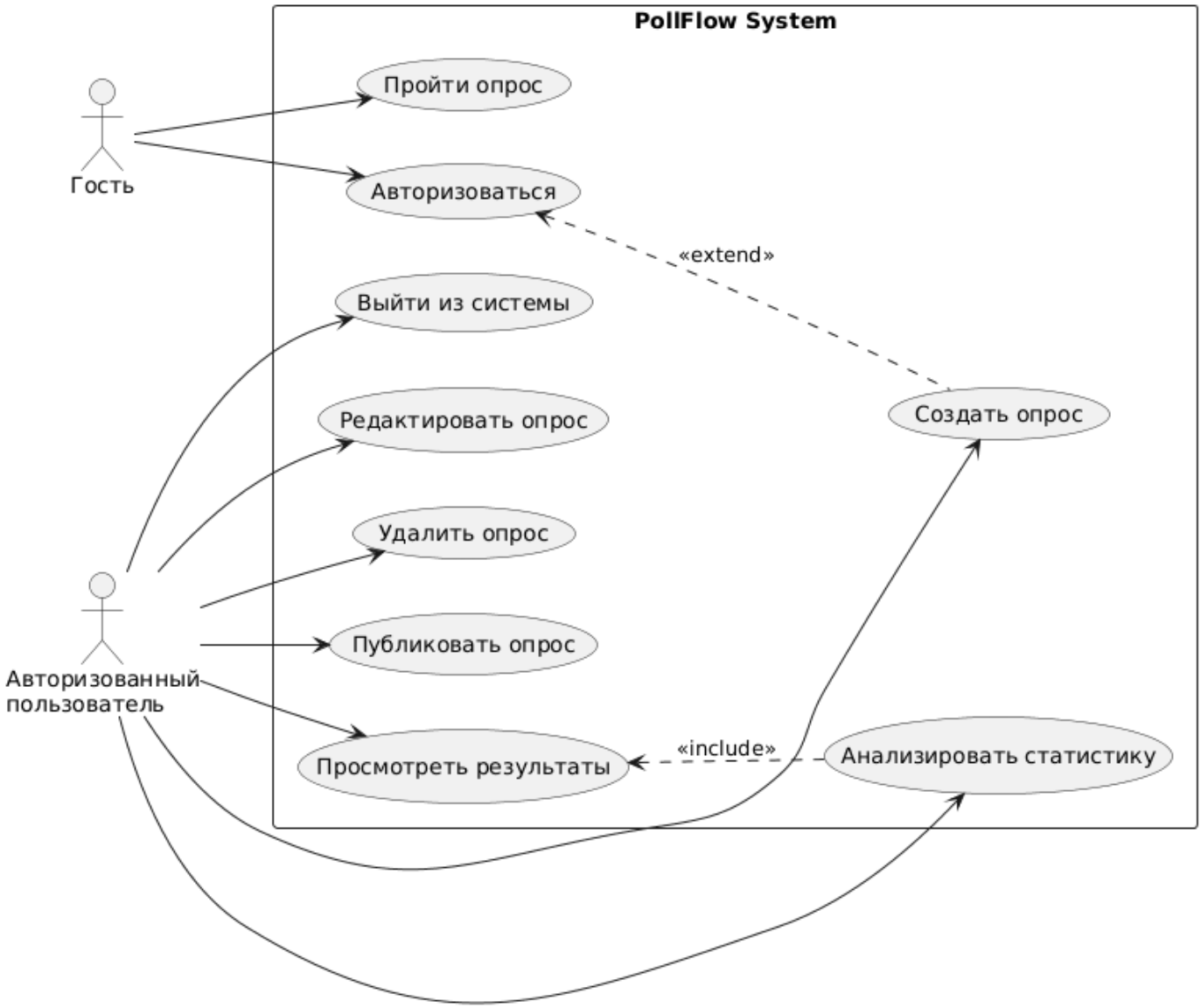


Схема представляет собой диаграмму вариантов использования, описывающую распределение функциональных возможностей системы между двумя категориями пользователей.

- **Система реализует модель ролевого доступа (RBAC),** где права четко разделены между неавторизованными (Гость) и авторизованными пользователями. Это обеспечивает минимально необходимый доступ для выполнения задач.
- **Гость обладает единственной привилегией — «Пройти опрос»,** что соответствует принципу анонимного участия и не требует аутентификации или хранения личных данных респондента.
- **Авторизованный пользователь получает полный цикл управления опросом,** включая операции создания, модификации, публикации, удаления, а также доступ к аналитическим функциям по просмотру и анализу результатов.
- **Процедура авторизации выступает обязательным шлюзом** для доступа к расширенному функционалу, что является базовым элементом управления доступом в системе.

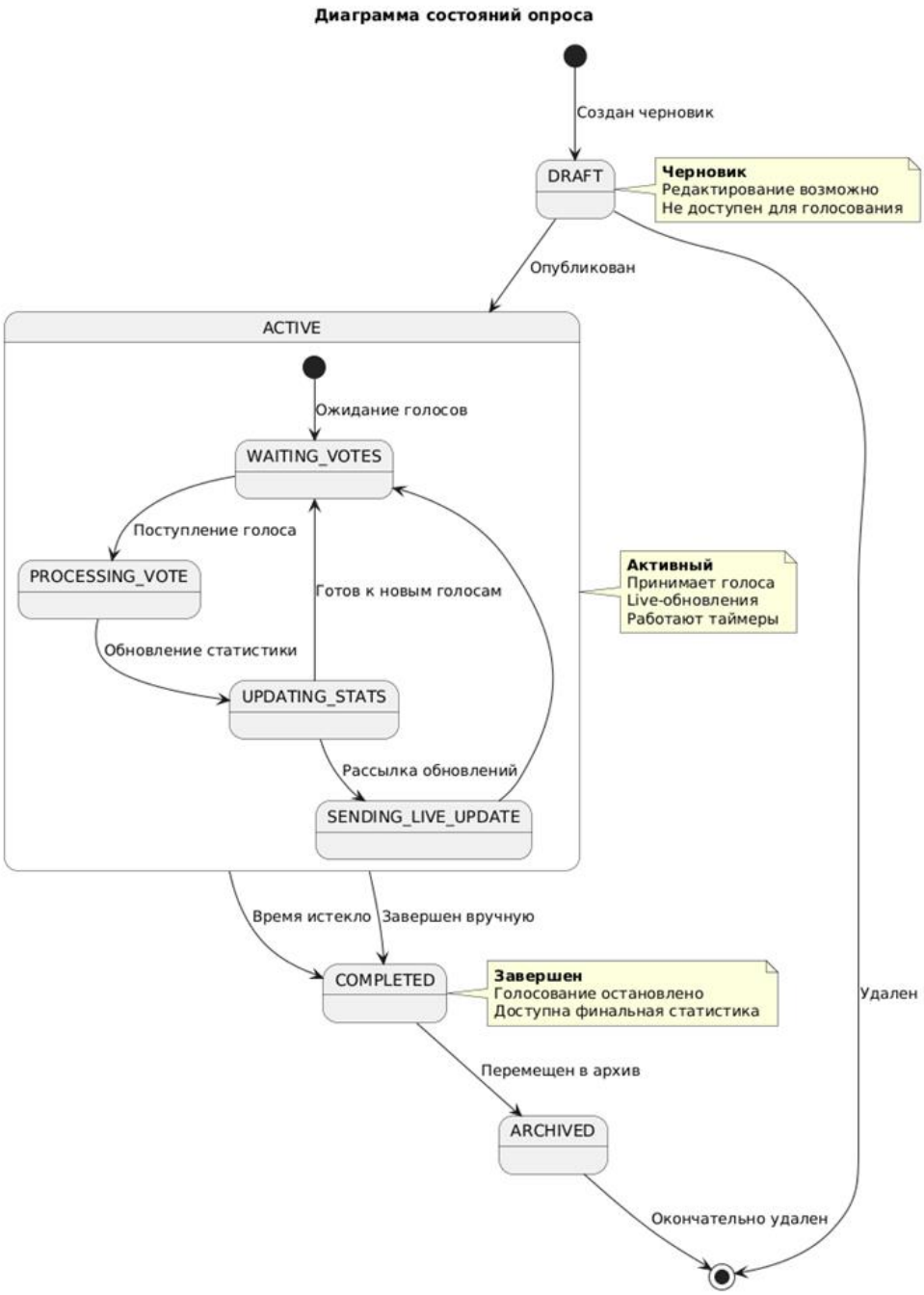


Схема описывает жизненный цикл опроса через диаграмму состояний.

- **Система реализует конечный автомат** с состояниями: DRAFT, ACTIVE, COMPLETED, ARCHIVED.
- **Черновик (**DRAFT**)** доступен только для редактирования создателем.
- **Активное состояние (**ACTIVE**)** является сложным и включает подсостояния для обработки голоса, обновления статистики и live-рассылки.
- **Завершение (**COMPLETED**)** наступает по таймеру или вручную, останавливая сбор ответов.
- **Архивация (**ARCHIVED**)** — финальный этап перед удалением данных.

Диаграмма корректно отображает бизнес-логику, однако смешивает уровни абстракции: состояние ACTIVE раскрыто как суб-автомат, демонстрирующий внутренние технические процессы (обработка голоса, рассылка), в то время как остальные состояния показаны на высоком, пользовательском уровне. Для полной формализации в нотации UML (State Machine Diagram) требуется явное разделение на уровни: верхний уровень — пользовательские статусы (DRAFT, ACTIVE, COMPLETED, ARCHIVED), нижний уровень — детализация технических состояний внутри ACTIVE.