# Архитектурные решения (ADR) и функциональные требования (FR)

16 марта 2025 г.

## Содержание

# 1 Архитектурные решения (ADR)

## 1.1 ADR-001: Выбор мобильной платформы и фреймворка

**Контекст:**
Приложение предназначено для геологов, проводящих полевые исследования. Требуется мобильное приложение, способное работать в офлайн-режиме, обеспечивать высокую отзывчивость интерфейса и синхронизироваться с серверной частью.

**Варианты решения:**

- **Нативная разработка (Android/iOS):** Максимальное использование возможностей платформ, однако требуется поддержка двух отдельных кодовых баз, что увеличивает временные затраты.
- **Кроссплатформенные фреймворки (React Native, Xamarin):** Единая кодовая база снижает затраты, но могут возникнуть ограничения по производительности и доступу к некоторым нативным API.
- **Flutter:** Обеспечивает высокую производительность благодаря собственному рендеринговому движку, позволяет создавать кроссплатформенные приложения с единой кодовой базой и имеет широкие возможности для кастомизации интерфейса, а также встроенную поддержку офлайн-режима.

**Решение:** Использовать **Flutter**.
**Обоснование:**

- Экономия ресурсов за счёт единой кодовой базы.
- Высокая производительность и отзывчивость интерфейса.
- Гибкость в разработке и возможность быстрой итерации.

**Последствия:**

- Быстрая разработка и выпуск обновлений.
- Возможность интеграции с нативными модулями при необходимости.

## 1.2 ADR-002: Выбор серверной платформы

**Контекст:**
Серверная часть отвечает за обработку запросов мобильного приложения, построение профилей местности по изолиниям, аутентификацию пользователей и интеграцию с внешними сервисами. Требуются высокая производительность, масштабируемость и безопасность.

**Варианты решения:**

- **ASP.NET Core:** Высокая производительность, современные стандарты безопасности, богатый набор библиотек и средств для масштабирования, а также простая интеграция с внешними системами.

- **Node.js (Express/Koa):** Хорош для быстрого прототипирования и эффективен при I/O-операциях, однако может быть менее эффективен при вычислительно затратных операциях и требует дополнительных мер безопасности.
- **Java (Spring Boot):** Надёжная и масштабируемая платформа, широко используемая в корпоративных системах, но требует более сложной настройки и сопровождения.

**Решение:** Использовать **ASP.NET Core**.
**Обоснование:**

- Оптимизирован для работы с большими нагрузками и масштабирования.
- Встроенные средства для обеспечения безопасности (аутентификация, авторизация).
- Лёгкость интеграции с внешними сервисами и базами данных.

**Последствия:**

- Формирование единого REST API для мобильного и потенциальных веб-клиентов.
- Упрощение поддержки и модернизации серверной части.
- Возможность быстрого реагирования на изменяющиеся требования.

## 1.3 ADR-003: Выбор базы данных

**Контекст:**
Приложению необходимо хранить данные о рельефе, профили местности, геоданные и информацию о пользователях. Важен вопрос хранения профилей местности, включая возможность объектного представления данных.

**Варианты решения:**
**PostgreSQL с расширением PostGIS**

- **Структурированность:** Реляционная модель обеспечивает строгую схему и целостность данных.
- **Геопространственные возможности:** PostGIS предоставляет мощный инструментарий для сложных пространственных запросов и анализа (например, построение профилей по изолиниям).
- **Гибкость хранения:** Поддержка JSON/JSONB позволяет хранить объектные данные в сочетании с фиксированной схемой для критичных данных.
- **Надёжность:** Полная ACID-совместимость гарантирует корректность транзакций.

**MongoDB**

- **Гибкость схемы:** Документная база не требует фиксированной схемы, что удобно для хранения сложных, вложенных структур в формате JSON.

- **Объектное хранение:** Естественное представление данных в виде JSON-объектов.
- **Масштабируемость:** Хорошая поддержка горизонтального масштабирования.
- **Геопространственная поддержка:** Обеспечивает базовые геопространственные операции, но не обладает таким же уровнем аналитических возможностей, как PostGIS.

**Решение:** Использовать **PostgreSQL с расширением PostGIS**.
**Обоснование:**

- Требуется мощный инструментарий для проведения сложных геопространственных запросов и анализа.
- Необходима высокая надёжность и целостность данных.
- Поддержка JSONB обеспечивает достаточную гибкость для хранения динамичных данных без перехода на полностью документную СУБД.

**Последствия:**

- Максимальное использование геопространственных возможностей для построения профилей.
- Стабильное хранение и обработка критичных данных.
- Возможность адаптации структуры данных за счёт поддержки JSONB.

## 1.4 ADR-004: Интеграция с внешними сервисами

**Контекст:**
Для построения точных профилей местности и корректного отображения карт необходим доступ к внешним данным.

**Варианты решения для интеграций:**
**Геоданные API:**

- **OpenTopoData:** Предоставляет точные высотные данные для построения профилей.
- **Mapbox Terrain API:** Альтернативный сервис с высококачественными данными рельефа.

**Картографический API:**

- **Google Maps:** Широко распространённый сервис с обширной функциональностью и надежностью.
- **OpenStreetMap (OSM):** Бесплатная альтернатива, позволяющая гибко настраивать отображение карт.

**Решение:** Интегрировать **OpenTopoData** для геоданных и **Google Maps** (с возможностью использования OSM в зависимости от лицензионных условий) для картографических слоёв.

**Обоснование:**

- Обеспечение актуальности и точности данных для построения профилей.
- Высокая стабильность и масштабируемость сервиса Google Maps.
- Гибкость выбора альтернативного решения (OSM) для снижения затрат при необходимости.

**Последствия:**

- Повышение точности построения профилей за счёт качественных данных.
- Необходимость управления API-ключами и соблюдения лицензионных требований.
- Резервирование на случай недоступности основного сервиса.

## 1.5 ADR-005: Пользовательская аутентификация и идентификация

**Контекст:**
Для обеспечения безопасности, персонализации опыта и управления доступом к функциям приложения требуется надёжная система аутентификации пользователей.

**Варианты решения:**

- **Identity Server (ASP.NET Core):** Интегрированное решение для реализации OAuth2/OpenID Connect, позволяющее обеспечить высокий уровень безопасности и гибкость в управлении доступом.
- **Альтернативные решения (Auth0, Okta):** Облачные сервисы с богатым функционалом, однако сложны в разработке.
- **Собственная разработка:** Разработка кастомного решения для аутентификации, что может увеличить время разработки и снизить общую безопасность.

**Решение:** Использовать **Identity Server**.

**Обоснование:**

- Прямая интеграция с ASP.NET Core упрощает реализацию и поддержку.
- Поддержка современных стандартов безопасности (OAuth2/OpenID Connect).
- Нет необходимости в дополнительных затратах на сторонние облачные сервисы.

**Последствия:**

- Обеспечение надёжной аутентификации и авторизации пользователей.
- Персонализация пользовательского опыта и управление доступом к функциям приложения.
- Возможность масштабирования системы управления доступом по мере роста нагрузки.

# 2 Функциональные требования (FR)

Ниже приведён перечень функциональных требований, вытекающих из бизнес-задач проекта и архитектурных решений:

### FR-001: Интерактивное построение профилей местности

- Система должна принимать входные данные в виде геоданных и изолиний.
- Автоматически генерировать профиль местности на основе полученных данных.
- Обеспечить возможность ручного редактирования и корректировки профиля.

### FR-002: Кроссплатформенность мобильного приложения

- Мобильное приложение должно работать на платформах Android и iOS.
- Обеспечивать единообразный пользовательский интерфейс и функциональность на всех устройствах.

### FR-003: Офлайн-режим и синхронизация данных

- Приложение должно поддерживать офлайн-режим с локальным кэшированием данных.
- При восстановлении соединения обеспечить автоматическую синхронизацию с сервером.

### FR-004: Безопасность и аутентификация пользователей

- Реализовать надёжную систему аутентификации и авторизации через Identity Server с использованием стандартов OAuth2/OpenID Connect.
- Обеспечить защиту персональных данных и разграничение прав доступа.

### FR-005: Интеграция с внешними сервисами

- Система должна интегрироваться с геоданными API (например, OpenTopoData) для получения высотных данных.
- Обеспечить интеграцию с картографическими API (например, Google Maps или OSM) для отображения картографических слоёв.
- Управлять API-ключами и предусмотреть резервирование в случае недоступности основного сервиса.

### FR-006: Экспорт данных и генерация отчетов

- Обеспечить экспорт построенных профилей и данных в форматы PDF, CSV, DXF.
- Реализовать возможность генерации аналитических отчетов по местности.

### FR-007: Мониторинг и логирование

- Внедрить системы мониторинга, логирования и алертинга для отслеживания производительности и оперативного реагирования на ошибки.

# 3 C4 Диаграмма

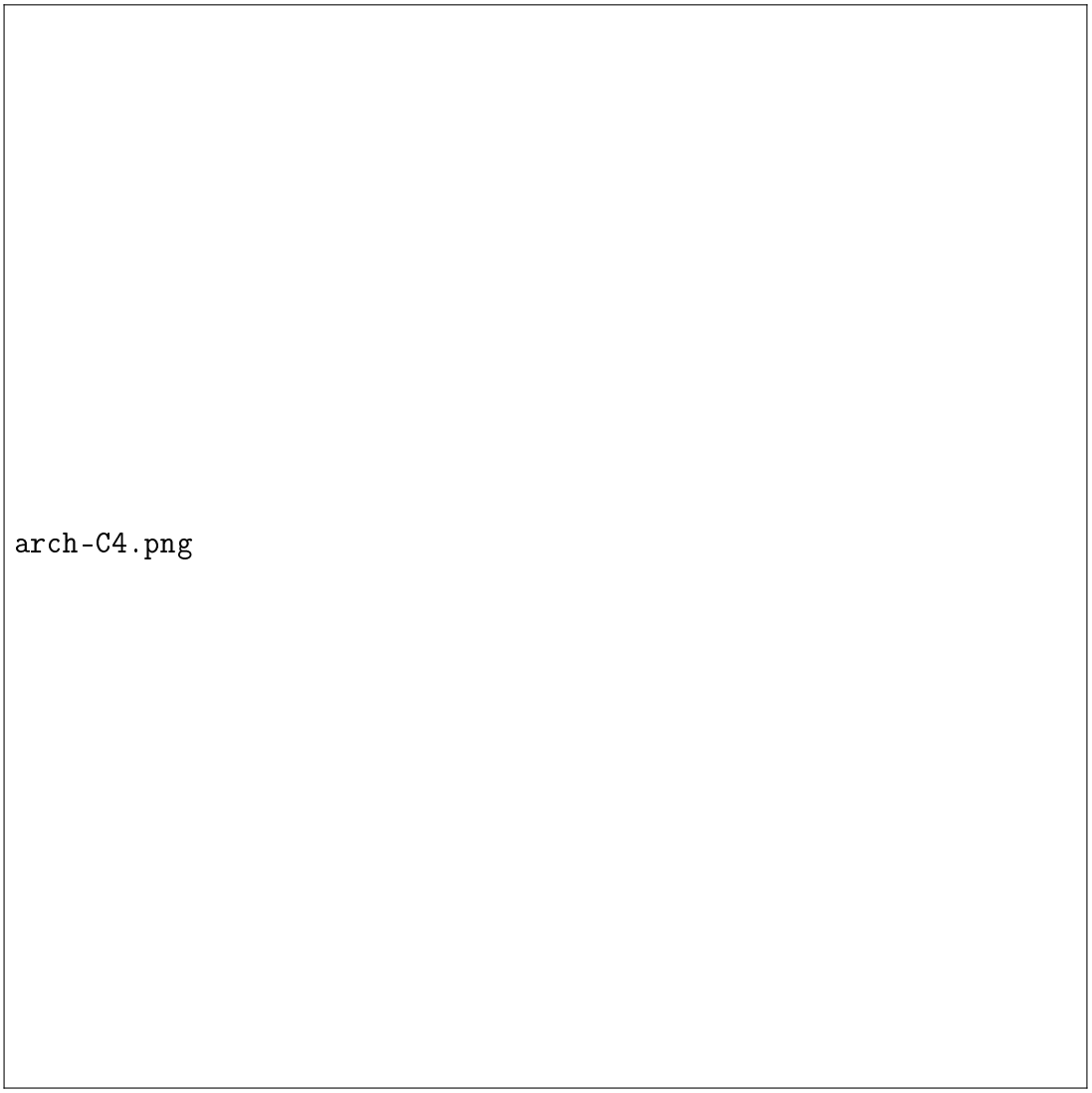


Рис. 1: C4 Диаграмма архитектуры приложения геологов.