ЭСТОНСКИЙ УНИВЕРСИТЕТ ПРЕДПРИНИМАТЕЛЬСТВА МАЙНОР

Учебная программа: Веб-дизайн и цифровая графика

Специализация: Веб-технологии

Ivan Ireev

**Определение наиболее часто упоминаемых в вакансиях технологий и инструментов для iOS и Android разработчиков на основе анализа вакансий из стран Европы и Северной Америки**

Заключительная работа

Руководитель: Boris Shvartsman, PhD

Таллинн 2025

ДЕКЛАРАЦИЯ ОБ АВТОРСКИХ ПРАВАХ

Декларирую, что составил/составила настоящую работу самостоятельно и что ранее она не представлялась к защите кем-либо другим. На все использованные при составлении работы исследования других авторов, принципиальные точки зрения, данные, полученные из литературных источников, имеются ссылки.

Автор: Ivan Ireev

06. 09. 2025

РЕЗЮМЕ

Работа посвящена выявлению наиболее часто упоминаемых в вакансиях технологий и инструментов для мобильных разработчиков *iOS* и *Android* в Европе и Северной Америке. Проблема — быстрые изменения стеков и требований рынка, затрудняющие выбор «ядра» навыков для обучения и найма. Цель — на основе анализа вакансий определить устойчивое платформенное ядро и сформировать практико-ориентированные рекомендации для специалистов и образовательных программ.

Данные собраны из *Google* *Jobs* через *SerpApi*; проведены очистка и дедупликация, определение языка (*Langdetect*), перевод неанглоязычных описаний (*GPT-4o*), разметка вакансий и извлечение технологий с последующей нормализацией/канонизацией и удалением «галлюцинаций». Точность подтверждена на подвыборках: *Token Set Ratio* составил 95,6 % для разметки секций и 98,71 % для извлечения технологий. Итоговый корпус — 2178 релевантных вакансий по *iOS* и *Android*; результаты сведены по 28 категориям; выводы оформлены в таблицах и интерактивном дашборде *Tableau* ([Ireev, 2025r](#Ireev_I_2025r_tableau_dashboard)).

Ключевые результаты: ядро требований устойчиво и платформенно-специфично (***iOS***: *Swift/Objective-C*, *Git*, *Xcode*, *iOS SDK*, *REST*, *SwiftUI/UIKit*, *MVVM*, *Core Data*; ***Android***: *Kotlin/Java*, *Git*, *Android* *Studio*, *Android SDK*, *REST*, *Jetpack* *Compose*, *MVVM*, *Room*), а межрегиональные различия проявляются преимущественно в «длинном хвосте». На основе сводных частот сформированы реплицируемые рекомендации «Основа/Дополнение» для обеих платформ.

Ограничения связаны с возможными неточностями машинного перевода, эффектами нормализации и исключением чрезмерно общих терминов; эти факторы могут умеренно влиять на частоты редких технологий, но не меняют структуру ядра требований.

Заключительная работа написана на русском языке, содержит 5 глав, 0 рисунков и 54 таблицы.

RESÜMEE

Töö keskendub töökuulutustes sagedamini mainitud tehnoloogiate ja tööriistade tuvastamisele iOS-i ja Androidi arendajatele Euroopas ja Põhja-Ameerikas. Probleem: tehnoloogiapakkide ja turunõuete kiire muutumine raskendab koolituse ja värbamise jaoks vajaliku „oskuste tuuma“ valikut. Eesmärk: kuulutuste analüüsi alusel määratleda stabiilne platvormipõhine tuum ning koostada praktikakesksed soovitused spetsialistidele ja haridusprogrammidele.

Andmed koguti Google Jobsi kaudu SerpApi abil; tehti puhastamine ja deduplitseerimine, keele tuvastus (Langdetect), mitteingliskeelsete kirjelduste tõlge (GPT-4o), kuulutuste märgendamine ja tehnoloogiate väljatõmbamine koos järgneva normaliseerimise/kanoniseerimisega ning „hallutsinatsioonide“ eemaldamisega. Täpsust kontrolliti alamhulkadel: Token Set Ratio oli 95,6% sektsioonide märgendamisel ja 98,71% tehnoloogiate väljatõmbamisel. Lõplik korpus hõlmab 2178 asjakohast kuulutust; tulemused koondati 28 kategooriasse; järeldused esitatakse tabelitena ja interaktiivsel Tableau juhtpaneelil ([Ireev, 2025r](#Ireev_I_2025r_tableau_dashboard)).

Põhitulemused: nõuete tuum on stabiilne ja platvormispetsiifiline (**iOS**: Swift/Objective-C, Git, Xcode, iOS SDK, REST, SwiftUI/UIKit, MVVM, Core Data; **Android**: Kotlin/Java, Git, Android Studio, Android SDK, REST, Jetpack Compose, MVVM, Room) ning piirkondadevahelised erinevused avalduvad peamiselt „pika saba“ osas. Koondatud sageduste alusel koostati mõlemale platvormile replitseeritavad soovitused „Alus/Täiendus“.

Piirangud on seotud võimalike masintõlke ebatäpsustega, normaliseerimise mõjudega ja ülemäära üldiste terminite välistamisega; need tegurid võivad mõõdukalt mõjutada haruldaste tehnoloogiate sagedusi, kuid ei muuda nõuete tuuma struktuuri.

Lõputöö on kirjutatud vene keeles; töös on 5 peatükki, 0 joonist ja 54 tabelit.

ABSTRACT

The study identifies the technologies and tools most frequently mentioned in job postings for iOS and Android developers in Europe and North America. The problem is the rapid evolution of stacks and market requirements, which complicates the selection of a “core” skill set for training and hiring. The objective is to determine a stable, platform-specific core based on job-posting analysis and to develop practice-oriented recommendations for professionals and educational programs.

Data were collected from Google Jobs via SerpApi; subsequent procedures included cleaning and deduplication, language identification (Langdetect), translation of non-English descriptions (GPT-4o), annotation of postings, and technology extraction followed by normalization/canonicalization and removal of “hallucinations”. Accuracy was validated on subsamples: the Token Set Ratio reached 95.6% for section annotation and 98.71% for technology extraction. The final corpus comprises 2,178 relevant iOS and Android postings; results are aggregated across 28 categories; findings are presented in tables and an interactive Tableau dashboard ([Ireev, 2025r](#Ireev_I_2025r_tableau_dashboard)).

Key results: the requirement core is stable and platform-specific (**iOS**: Swift/Objective-C, Git, Xcode, iOS SDK, REST, SwiftUI/UIKit, MVVM, Core Data; **Android**: Kotlin/Java, Git, Android Studio, Android SDK, REST, Jetpack Compose, MVVM, Room), while cross-regional differences appear primarily in the “long tail”. Based on pooled frequencies, replicable Core/Addition recommendations are proposed for both platforms.

Limitations concern potential machine-translation inaccuracies, normalization effects, and the exclusion of overly general terms; these factors may moderately affect the frequencies of rare technologies but do not alter the structure of the requirement core.

The final thesis is written in Russian and contains 5 chapters, 0 figures, and 54 tables.

ОГЛАВЛЕНИЕ

[ДЕКЛАРАЦИЯ ОБ АВТОРСКИХ ПРАВАХ 1](#_Toc208061387)

[РЕЗЮМЕ 2](#_Toc208061388)

[RESÜMEE 3](#_Toc208061389)

[ABSTRACT 4](#_Toc208061390)

[СОКРАЩЕНИЯ И ТЕРМИНЫ 9](#_Toc208061391)

[ВВЕДЕНИЕ 10](#_Toc208061392)

[Актуальность темы 11](#_Toc208061393)

[Проблема и цель исследования 12](#_Toc208061394)

[Задачи исследования 12](#_Toc208061395)

[Границы и допущения исследования 14](#_Toc208061396)

[1. АНАЛИЗ СУЩЕСТВУЮЩИХ ИССЛЕДОВАНИЙ 16](#_Toc208061397)

[1.1. Обзор исследований по рынку труда в *IT*-сфере 16](#_Toc208061398)

[1.2. Исследования, фокусирующиеся на *iOS-* и *Android-*разработчиках 18](#_Toc208061399)

[1.3. Выводы и пробелы в исследованиях 21](#_Toc208061400)

[2. ВЫБОР ИНСТРУМЕНТОВ И ПОДХОДОВ 23](#_Toc208061401)

[2.1. Выбор языка программирования 23](#_Toc208061402)

[2.2. Ключевые библиотеки *Python* и их роль 23](#_Toc208061403)

[2.3. Сбор вакансий с помощью *SerpApi* 23](#_Toc208061404)

[2.4. Определение языка вакансий с помощью *Langdetect* 24](#_Toc208061405)

[2.5. Выбор большой языковой модели (*LLM*) 25](#_Toc208061406)

[2.6. Нечёткое сопоставление 25](#_Toc208061407)

[3. МЕТОДИКА ИССЛЕДОВАНИЯ 27](#_Toc208061408)

[3.1. Подготовка и сбор данных 27](#_Toc208061409)

[3.2. Создание набора данных 28](#_Toc208061410)

[3.3. Первичная очистка данных 28](#_Toc208061411)

[3.4. Определение языка вакансий 28](#_Toc208061412)

[3.5. Перевод неанглоязычных вакансий на английский 29](#_Toc208061413)

[3.6. Разделение вакансий на логические части 30](#_Toc208061414)

[3.7. Отбор только *iOS*- и *Android*-вакансий 32](#_Toc208061415)

[3.8. Извлечение технологий и инструментов 33](#_Toc208061416)

[3.9. Очистка данных 34](#_Toc208061417)

[3.10. Группировка технологий по категориям 36](#_Toc208061418)

[3.11. Финальное преобразование данных 36](#_Toc208061419)

[3.12. Формат представления результатов 37](#_Toc208061420)

[4. РЕЗУЛЬТАТЫ 38](#_Toc208061421)

[4.1. Архитектурные шаблоны программного обеспечения 38](#_Toc208061422)

[4.2. Инструменты командного взаимодействия и управления разработкой 39](#_Toc208061423)

[4.3. Интегрированные среды разработки (*IDE*) 40](#_Toc208061424)

[4.4. Интероперабельность и взаимодействие с нативным кодом 41](#_Toc208061425)

[4.5. Информационная безопасность и нормативное соответствие 42](#_Toc208061426)

[4.6. Исполняемые среды и мультиплатформенные рантаймы 43](#_Toc208061427)

[4.7. Кроссплатформенные *SDK* для устройств и сервисов 44](#_Toc208061428)

[4.8. Кроссплатформенные *UI*-фреймворки 46](#_Toc208061429)

[4.9. Методологии и практики управления разработкой 47](#_Toc208061430)

[4.10. Мониторинг и аналитика 48](#_Toc208061431)

[4.11. Нативные *UI*-фреймворки и компоненты 49](#_Toc208061432)

[4.12. Непрерывная интеграция и доставка (*CI/CD*) 50](#_Toc208061433)

[4.13. Отладка, профилирование и реверс-инжиниринг 51](#_Toc208061434)

[4.14. Параллелизм и реактивное программирование 52](#_Toc208061435)

[4.15. Публикация и распространение приложений 53](#_Toc208061436)

[4.16. Разработка игр 54](#_Toc208061437)

[4.17. Руководства по дизайну интерфейсов 55](#_Toc208061438)

[4.18. Сборка и управление зависимостями 55](#_Toc208061439)

[4.19. Серверные платформы и облачные сервисы 56](#_Toc208061440)

[4.20. Сетевые протоколы и клиентские библиотеки 58](#_Toc208061441)

[4.21. Системы контроля версий и платформы 59](#_Toc208061442)

[4.22. Спецификация и документация *API* 60](#_Toc208061443)

[4.23. Тестирование и обеспечение качества 61](#_Toc208061444)

[4.24. Фреймворки внедрения зависимостей (*DI*) 62](#_Toc208061445)

[4.25. Хранилища данных и СУБД 63](#_Toc208061446)

[4.26. Экосистема *Android* 64](#_Toc208061447)

[4.27. Экосистема *Apple* 65](#_Toc208061448)

[4.28. Языки программирования и разметки 67](#_Toc208061449)

[4.29. Итоговый блок 68](#_Toc208061450)

[5. ОБСУЖДЕНИЕ И ПРЕДЛОЖЕНИЯ 75](#_Toc208061451)

[5.1. Связь результатов с теорией и предшествующими исследованиями 75](#_Toc208061452)

[5.2. Ограничения 76](#_Toc208061453)

[5.3. Предложения 78](#_Toc208061454)

[ЗАКЛЮЧЕНИЕ 80](#_Toc208061455)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 82](#_Toc208061456)

[ПРИЛОЖЕНИЯ 90](#_Toc208061457)

[Приложение 1. Полный список стран 90](#_Toc208061458)

[Приложение 2. Директории с вакансиями 91](#_Toc208061459)

[Приложение 3. Единый набор данных 92](#_Toc208061460)

[Приложение 4. Промпт для разбиения вакансий на логические части 93](#_Toc208061461)

[Приложение 5. Пример вакансии после преобразования 94](#_Toc208061462)

[Приложение 6. Промпт для извлечения технологий из вакансий 95](#_Toc208061463)

СОКРАЩЕНИЯ И ТЕРМИНЫ

* *CD* — непрерывная доставка/развёртывание (*Continuous Delivery/Deployment*).
* *CI* — непрерывная интеграция (*Continuous Integration*).
* *CPU* — центральный процессор (*Central Processing Unit*).
* *CSV* — текстовый формат данных с разделителем-запятой (*Comma-Separated Values*).
* *Git* — система контроля версий.
* *HAL* — *Hardware Abstraction Layer*, слой абстракции оборудования (*Android*).
* *HTTP* — протокол передачи гипертекста (*Hypertext Transfer Protocol*).
* *JSON* — формат обмена данными *JavaScript Object Notation*.
* *NDK* — (*Android*) *Native Development Kit*, инструментарий для нативного *C/C++*-кода.
* *NFC* — *Near Field Communication*, ближняя бесконтактная связь.
* *NLU* — *Natural* *Language* *Understanding*, «понимание» естественного языка (подобласть *NLP*).
* *REST* — архитектурный стиль взаимодействия клиент–сервер (*Representational* *State* *Transfer*).
* *SPM* — *Swift* *Package* *Manager*, менеджер пакетов для экосистемы *Swift*.
* *SQL* — *Structured* *Query* *Language*, язык структурированных запросов.
* *TF-IDF* — *Term Frequency–Inverse Document Frequency*, взвешивание важности терминов.
* *TLS* — *Transport* *Layer* *Security*, криптопротокол защиты транспортного уровня.
* *UI* — пользовательский интерфейс (*User Interface*).
* *UTC* — *Coordinated Universal Time*, координированное всемирное время.
* *XML* — *eXtensible Markup Language*, расширяемый язык разметки.

ВВЕДЕНИЕ

Мобильная разработка представляет собой процесс создания программного обеспечения, предназначенного для работы на мобильных устройствах, таких как смартфоны и планшеты. Рынок мобильных приложений динамично растёт ([Grand View Research, n.d.](#Grand_View_Research_n_d_Global_Mbl_app); [Precedence Research, 2025](#Precedence_Research_2025_Mobile_App_mrkt); [Statista, 2023](#Statista_2023_App_Worldwide); [Statista, 2025](#Statista_2025_App_Europe)) по мере роста числа пользователей мобильных устройств ([Howarth, 18 июня 2025](#Howarth_2025_18_июня_How_Mn_Ppl_own_smrt); [IDC, 2024](#IDC_2024_Smartphone_Market_Insights); [Statista, 2021](#Statista_2021_Forecast_num_of_mibile_use)), скачиваний ([Qadir, 15 апреля 2025](#Qadir_2025_15_апреля_Mobile_App_downlds); [Statista, 2024](#Statista_2024_Number_of_mobile_app)) и потребительских расходов в приложениях ([Briskman, февраль 2025](#Briskman_2025_Q4_2024_digital_mrket_Indx)), а также углубляющейся цифровизации. В совокупности это усиливает спрос на мобильных разработчиков.

За последнее десятилетие общее число разработчиков по всему миру увеличилось почти в десять раз — с 2,5 млн человек в 2010 г. до 28 млн человек в 2024 г. ([Human Labs, 2025](#Human_Labs_2025_Top_Mobile_App_Dev_sats)). В ЕС в 2019–2023 гг. количество рабочих мест в «экономике приложений» выросло на 53 % ([Mandel, 2023](#Mandel_2023_EU_App_Economy)), а в США за период 2018–2028 гг. прогнозируется рост числа разработчиков на 21 % ([Zippia, 2024](#Zippia_2024_Job_outlook_for_mbl_app_dev)) — что в совокупности формирует ощутимый кадровый спрос.

По оценкам, только в США в 2024 г. было открыто примерно 200–400 тыс. вакансий в сфере мобильной разработки ([Crudu & MoldStud Research Team, 4 января 2025](#Crudu_MoldStud_Research_Team_2025_4_янв); [Zippia, 2024](#Zippia_2024_Job_outlook_for_mbl_app_dev)). Многие отрасли — финансы, здравоохранение, государственный сектор и другие — ищут таких специалистов как в США, так и в Европе ([Mandel, 2023](#Mandel_2023_EU_App_Economy)). На фоне этого спроса особенно важно оценить влияние нового фактора — генеративного ИИ.

Быстрый прогресс генеративного ИИ в 2023–2024 гг. не снизил, а усилил спрос на мобильных разработчиков, поскольку рынок приложений на основе генеративного ИИ (чат-боты, генеративные изображения/контент и др.) породил новые продуктовые ниши, требующие интуитивных мобильных интерфейсов и глубокой интеграции ИИ ([Briskman, январь 2024](#Briskman_2024_янв_state_of_mobile)). Согласно опросу *Stack Overflow* ([2024](#Stack_Overflow_2024_Annual_Dev_Survey)), 70 % профессиональных разработчиков не считают ИИ угрозой своей работе, а 76 % уже применяют или планируют применять ИИ-инструменты в процессе разработки. Повышенная продуктивность, обеспеченная такими инструментами ([Brown, 2024](#Brown_2024_Research_AI_productivity_boos)), расширяет обязанности инженеров: им приходится не только встраивать модули искусственного интеллекта прямо на устройство (*SDK* для локального *(on-device)* ИИ), но и адаптировать решения к новым типам устройств и форм-факторам. Тем самым профессия разработчика не исчезает: она эволюционирует и ставит перед специалистами новые вызовы.

Ключевой вызов — необходимость постоянно адаптироваться к быстрой эволюции технологий: платформы *iOS* и *Android* регулярно выпускают обновления, внедряют новые языки программирования, фреймворки и подходы к разработке. Это создаёт ситуацию, в которой разработчикам может становиться трудно своевременно ориентироваться в меняющихся требованиях рынка и выбирать наиболее актуальные и востребованные инструменты; что, в свою очередь, приводит к быстрому устареванию учебных программ и снижению конкурентоспособности выпускников.

## Актуальность темы

Актуальность темы обусловлена масштабом и высокой динамикой мобильной экономики (рекордные расходы пользователей и время в приложениях), ускоряющимися технологическими сдвигами (*5G/eSIM*) и постоянными изменениями в инструментах и навыках разработчиков; всё это делает необходимым системный обзор ключевых тенденций в мобильной разработке ([GSMA, 2024](#GSMA_2024_The_Mobile_Economy_2024); [JetBrains, 2024](#JetBrains_2024_State_of_Developer_Ecosys); [Sensor Tower, 2025](#Sensor_Tower_2025_State_of_Mobile_econom)[; Stack Overflow, 2024](#Stack_Overflow_2024_Annual_Dev_Survey)).

Внимание сосредоточивается именно на Северной Америке и Европе, поскольку, несмотря на то, что Азиатско-Тихоокеанский регион в настоящее время лидирует по совокупному объёму рынка, эти два региона объединяют наивысший в мире уровень проникновения смартфонов (≈ 80–91 %) и самый высокий средний доход на пользователя, делая их одними из наиболее прибыльных направлений для монетизации мобильных приложений ([Global Growth Insights, 2025](#Global_Growth_Insights_2025_mob_app_m_s); [Grand View Research, n.d.](#Grand_View_Research_n_d_Global_Mbl_app)). Северная Америка и Европа продолжают оставаться основными центрами притяжения для разработчиков мобильных приложений, доля которых составляет около 40–49 % мировой популяции разработчиков ([Georgakopoulos, 11 апреля 2024](#Georgakopoulos_2024_11_апр_SlashData); [Human Labs, 2025](#Human_Labs_2025_Top_Mobile_App_Dev_sats)). Кроме того, в этих регионах наблюдаются самые высокие в мире уровни заработной платы мобильных разработчиков — около 120–200 тыс. долл. США в год в Северной Америке и 35–100 тыс. долл. США в Европе, что значительно превышает показатели Азиатско-Тихоокеанского региона (15–50 тыс. долл. США), Латинской Америки (30–50 тыс. долл. США) и Африки (20–40 тыс. долл. США) ([Crudu, 6 мая 2025](#Crudu_2025_6_мая_cost_in_hiring_MoldStud); [Khmelevska, n.d.](#Khmelevska_n_d_mbl_dev_salary_Bridge); [TechRound, 2024](#TechRound_2024_The_ultimt_guide_to_hire)). Такая значительная концентрация кадров, охватывающая почти половину мировой популяции разработчиков, в сочетании с привлекательным уровнем вознаграждения, вероятно, будет усиливать конкуренцию за позиции разработчиков.

Подводя итог, выявление наиболее востребованных технологий и инструментов для мобильных разработчиков на рынках Европы и Северной Америки позволит специалистам более эффективно ориентироваться в текущих тенденциях и выбирать оптимальное направление профессионального развития. Для образовательных учреждений подобные исследования служат основой для корректировки и обновления учебных программ, обеспечивая их актуальность и соответствие современным реалиям рынка.

## Проблема и цель исследования

Проблема исследования заключается в сложности определения наиболее востребованных навыков и инструментов для мобильных разработчиков в условиях стремительного развития технологий.

Цель исследования — выявить наиболее часто упоминаемые в вакансиях технологии для *iOS* и *Android* разработчиков и составить рекомендации по изучению, которые помогут специалистам адаптироваться к текущим тенденциям рынка труда, а образовательным учреждениям — формировать программы, соответствующие современным требованиям индустрии.

## Задачи исследования

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести обзор литературы, посвящённой анализу рынка труда мобильных разработчиков.
2. Определить географию анализа и критерии отбора стран.
3. Собрать данные о вакансиях с помощью *Python* и *SerpApi*.
4. Выполнить первичную очистку данных (удалить дубликаты по *ID* и отфильтровать нерелевантные позиции).
5. Определить язык текста вакансий с помощью *Langdetect*, в спорных случаях выполнить ручную проверку.
6. Перевести все неанглоязычные вакансии на английский с помощью модели *GPT-4o*.
   * Комментарий: единый язык гарантирует, что следующие модели обрабатывают объявления в одинаковых условиях.
7. На случайной подвыборке объявлений (n ≥ 30) определить и разметить логические блоки — *Platform*, *Salary*, *Requirements*, *Nice to Have*, *Responsibilities*, *Benefits* — c помощью *ChatGPT‑4o*; выполнить валидацию качества разметки и, при удовлетворительных результатах, распространить процедуру на весь набор данных.
8. Оставить только подлинные вакансии *iOS‑* и *Android‑*разработчиков, отфильтровав их по блоку *Platform*.
   * Комментарий: после полной разметки кроссплатформенные/смешанные вакансии будут исключены; ожидается высокая, но не 100% точность.
9. На случайной подвыборке вакансий (*n* ≥ 30) извлечь упомянутые технологии и инструменты (из *Requirements*, *Nice to Have*, *Responsibilities*) при помощи *GPT-4o*; провести валидацию качества извлечения и, при удовлетворительных результатах, масштабировать обработку на весь набор данных.
10. Очистить, нормализовать и отфильтровать списки технологий и инструментов (объединить синонимы, выровнять регистр и т. п.).
11. Дополнительно очистить словарь технологий:
    * удалить искусственно сгенерированные моделью термины.
    * минимизировать влияние платформенных несоответствий: по возможности удалять явные ошибки (например, *Swift* в пуле *Android*; *Kotlin* в пуле *iOS*), а пограничные случаи сохранять и интерпретировать отдельно; точность блока *Platform*, вероятно, не будет 100 %.
12. Вручную сгруппировать технологии и инструменты по категориям (языки программирования, системы контроля версий и др.).
13. Представить результаты в табличной форме по категориям и в виде интерактивного дашборда в *Tableau*.
14. Сопоставить требования работодателей к мобильным разработчикам в Европе и Северной Америке, выявив сходства и различия.
15. Сформулировать выводы и рекомендации, которые помогут:
    * мобильным разработчикам выбрать оптимальный технологический стек.
    * образовательным учреждениям адаптировать учебные программы в соответствии с актуальными требованиями рынка труда.

## Границы и допущения исследования

В данном исследовании под «востребованностью технологий» понимается их относительная частота упоминаний в вакансиях: чем выше доля вакансий с упоминанием конкретной технологии, тем выше считается её востребованность.

Регион «Европа» включает страны — члены ЕС и/или Шенгенской зоны, а также Великобританию, что обеспечивает охват ключевых рынков мобильной разработки в Европе. В состав региона «Северная Америка» входят США, Канада и Мексика. Полный список стран см. в [Приложении 1](#Приложение_1_Полный_список_стран).

**Отсутствие разделения на грейды**

В выборку включены вакансии всех уровней — *junior*, *middle* и *senior*; далее разделения на эти уровни не будет по следующим причинам:

Во-первых, отсутствует единый, общепринятый стандарт грейдинга: компании используют собственные карьерные лестницы и терминологию, что затрудняет сопоставимость между работодателями и регионами. Даже отраслевые рамки (например, *SFIA*) описывают уровни ответственности и намеренно не предписывают роли/должности; практики их внедрения в компаниях различаются ([SFIA Foundation, n.d. -a](#SFIA_Foundation_n_d_a_About_SFIA); [SFIA Foundation, n.d. -b](#SFIA_Foundation_n_d_b_how_SFIA_works)). Иллюстрацией служат расхождения в грейдах и названиях уровней у разных компаний на примере *Microsoft*, *Bolt (EU)* и *Twilio* ([Levels.fyi, n.d.](#Levels_fyi_n_d_Compare_career_levels)). Отсюда следует, что прямое сопоставление *«junior/middle/senior»* по текстам вакансий может вносить систематические искажения.

Во-вторых, тексты вакансий часто содержат неоднозначные формулировки и неполные сведения о «старшинстве» (*seniority*) ([Tzimas, Zotos, Mourelatos, Giotopoulos, & Zervas, 2024](#Tzimas_Zotos_Mourelatos_et_al_2024)): исследовательская и прикладная литература по обработке вакансий указывает на высокую вариативность и «длинный хвост» названий должностей и ролей ([Dixon, Goggins, Ho, Howison, Long, Northcott, Shen & Yeats, 2023](#Dixon_Goggins_Ho_Howison_et_al_2023); [Yamashita, Shen, Tran, Ekhtiari & Lee, 2023](#Yamashita_Shen_Tran_et_al_2023)), что требует сложной нормализации и всё равно оставляет погрешности ([Tzimas](#Tzimas_Zotos_Mourelatos_et_al_2024) *[et al.](#Tzimas_Zotos_Mourelatos_et_al_2024)*[, 2024](#Tzimas_Zotos_Mourelatos_et_al_2024)). Это касается как нормализации названий ([Dixon](#Dixon_Goggins_Ho_Howison_et_al_2023) *[et al.](#Dixon_Goggins_Ho_Howison_et_al_2023)*[, 2023](#Dixon_Goggins_Ho_Howison_et_al_2023); [Yamashita *et al.*, 2023](#Yamashita_Shen_Tran_et_al_2023)), так и извлечения признаков «старшинства» из описаний — задачи, которые считаются нетривиальными даже при использовании современных методов *NLP* ([Tzimas](#Tzimas_Zotos_Mourelatos_et_al_2024) *[et al.](#Tzimas_Zotos_Mourelatos_et_al_2024)*[, 2024](#Tzimas_Zotos_Mourelatos_et_al_2024); [Yamashita *et al.*, 2023](#Yamashita_Shen_Tran_et_al_2023)). В таких условиях автоматическое и объективное определение грейда для тысяч объявлений становится ненадёжным.

В-третьих, переход от *junior* к *senior* в основном связан с углублением экспертизы и ростом масштаба/ответственности, а не с радикальной сменой инструментов: открытые рамки *Spotify* и *Monzo* описывают развитие как рост вклада и расширение сферы влияния; уровни не привязаны к конкретным языкам/стекам ([Goldsmith, 8 февраля 2016](#Goldsmith_2016_8_февраля); [Goldsmith, 15 февраля 2016](#Goldsmith_2016_15_февраля); [Monzo, n.d.](#Monzo_n_d_Engineering_Progression_Framew)).

Наконец, с точки зрения валидности выводов по рынку, агрегирование вакансий по «условному среднему» мобильному разработчику снижает риск искажения из-за произвольной маркировки грейдов работодателями и позволяет сфокусироваться на сопоставимых, платформенно-зависимых технологиях.

1. АНАЛИЗ СУЩЕСТВУЮЩИХ ИССЛЕДОВАНИЙ

В современных условиях рынок труда в *IT*-сфере привлекает значительное внимание исследователей, однако анализ рынка труда для *iOS*- и *Android*-разработчиков остаётся недостаточно изученным. Существующие работы, затрагивающие смежные вопросы, чаще всего ограничиваются либо общими тенденциями *IT*-рынка, либо рассматривают более широкую категорию «разработчиков программного обеспечения» в целом, либо фокусируются на анализе рынка труда в рамках отдельных стран.

## 1.1. Обзор исследований по рынку труда в *IT*-сфере

Популярные отраслевые опросы и аналитические отчёты — например, *Stack Overflow Developer Survey* и ежегодный *State of Developer Ecosystem* от *JetBrains* — регулярно фиксируют глобальные тенденции ([JetBrains, 2024](#JetBrains_2024_State_of_Developer_Ecosys); [Stack Overflow, 2024](#Stack_Overflow_2024_Annual_Dev_Survey)). К примеру, *Stack Overflow Developer Survey* ([2024](#Stack_Overflow_2024_Annual_Dev_Survey)) представляет широкий срез *IT*-специалистов по всему миру (65 437 респондентов) и содержит данные о наиболее распространённых языках программирования, технологиях и профессиональных ролях. Аналогично, отчёт *JetBrains* ([2024](#JetBrains_2024_State_of_Developer_Ecosys)) опирается на ответы 23 262 разработчиков и охватывает языки, инструменты и практики разработки на глобальном уровне. Подобные исследования дают ценную макрокартину, но не фокусируются на отдельных платформах: мобильные разработчики выступают в них лишь малой частью выборки (лишь около 3,4 % мобильных разработчиков среди всех респондентов ([Stack Overflow, 2024, Developer Profile](#Stack_Overflow_2024_Annual_Dev_Survey))).​

Исследование ([Cummings, Janicki, & Matthews, 2023](#Cummings_Janicki_Matthews_2023)) анализирует востребованные технологии и навыки на основе опроса 566 *IS/IT*-профессионалов разных ролей (в т.ч. разработчиков, аналитиков, менеджеров проектов). В работе рассматриваются популярные инструменты, тенденции их развития и прогнозы по изменению спроса на навыки в *IT*-сфере. Однако исследование фокусируется на индустрии в целом и не предоставляет детальной информации о технологиях, специфичных для мобильной разработки.

В исследовании ([Apatsidis, Georgiou, Mittas, & Angelis, 2021](#Apatsidis_Georgiou_Mittas_Angelis_2021)) проведён анализ 8514 объявлений о вакансиях с платформы *Stack Overflow* для выявления различий между удалёнными и офисными позициями в сфере информационно-коммуникационных технологий (*ICT*). Авторы применили методы текстового анализа и теорию графов, чтобы определить востребованные технологии и типы должностей. Хотя исследование даёт представление о ключевых технологиях в разработке программного обеспечения, оно не фокусируется на мобильной разработке.

Исследование ([Kang, Park, & Shin, 2020](#Kang_Park_Shin_2020)) посвящено анализу современных тенденций и ключевых технических навыков, востребованных в сфере разработки программного обеспечения. Основное внимание авторы уделяют выявлению разрыва между академической подготовкой и требованиями индустрии, который особенно актуален в *IT*-сфере из-за её динамичного и быстро меняющегося характера. В рамках исследования был проведён анализ более 120 000 вакансий, собранных с различных платформ для поиска работы по всей территории США. Это позволило определить наиболее востребованные навыки, включая языки программирования, интегрированные среды разработки (*IDE*), инструменты управления проектами, а также «мягкие навыки» (*soft skills*), которые активно запрашиваются работодателями. Однако отсутствие фокуса на мобильной разработке в исследовании ограничивает его применимость для специалистов в этой области.

Исследование ([Ternikov & Aleksandrova, 2020](#Ternikov_Aleksandrova_2020)) посвящено анализу востребованности навыков на рынке труда в *IT*-секторе. Авторы предложили алгоритм, который позволяет извлекать и унифицировать наборы ключевых навыков, востребованных различными профессиональными группами в этой сфере. Для выполнения задачи использовалась база данных с описанием вакансий из онлайн-ресурсов российского рынка труда. Анализ данных проводился с применением методов обработки текста, таких как *TF-IDF* и *n-grams*. В результате исследования были выявлены основные навыки, актуальные для различных *IT*-профессий, а также популярные комбинации навыков, пользующиеся наибольшим спросом у работодателей. Важно отметить, что мобильные разработчики не выделялись как отдельная профессиональная группа. Их навыки рассматривались в контексте более широких категорий, таких как высококвалифицированные *IT*-специалисты или разработчики программного обеспечения.

Проект ([Barousse, n.d.](#Barousse_n_d_Data_Job_Pipeline_Airflow); [Data Nerds, n.d.](#Data_Nerds_n_d_)), разработанный *Luke Barousse* ([17 февраля 2023](#Barousse_Luke_2023)), представляет собой аналитическую платформу, исследующую востребованные навыки и уровень заработных плат в сфере *IT*. Для сбора вакансий используется *SerpApi* (*Google Jobs*/поиск), с последующей загрузкой в *BigQuery*. В среднем ежедневно обрабатываются тысячи вакансий, преимущественно относящихся к *data science* и смежным направлениям. Полученные данные проходят предобработку и извлекаются с использованием *NLP*-пайплайна, после чего агрегируются для выявления наиболее часто упоминаемых в вакансиях компетенций и расчёта медианных значений заработных плат. Основной целью сервиса является предоставление обоснованной информации специалистам в области данных (*Data Analysts*, *Data Engineers*, *Data Scientists* и т. д.), однако платформа может быть полезна и для разработчиков программного обеспечения. На веб-сайте представлена статистика о наиболее часто упоминаемых в вакансиях языкам, инструментам, БД, облачным сервисам, библиотекам и фреймворкам, а также динамика их популярности. Несмотря на то, что проект даёт общее представление о ключевых технологиях в сферах разработки программного обеспечения и *data science*, он не фокусируется на мобильной разработке.

## 1.2. Исследования, фокусирующиеся на *iOS-* и *Android-*разработчиках

**1.2.1. Исследования команды Яндекс Практикума**

Отдельного упоминания заслуживают исследования, опубликованные на *Habr*, проведённые командой Яндекс Практикума ([2024, 10 июня](#Яндекс_Практикум_2024_10_06_iOS); [2024, 3 июля](#Яндекс_Практикум_2024_03_06_Android);), специально посвящённые рынку труда *iOS*- и *Android*-разработчиков. В рамках этих исследований анализировалась ситуация с наймом младших специалистов (*junior* и *junior plus*) в сфере мобильной разработки на российском рынке.

Методология включала комбинированный подход: анализ вакансий на *job*-портале *HH.ru* и экспертные интервью. В частности, эксперты опросили технических руководителей команд и ведущих разработчиков ряда компаний (в основном российских, а также некоторых зарубежных) для выяснения требований к кандидатам. Параллельно были собраны и проанализированы актуальные вакансии уровня *junior* и *junior plus* по *iOS*- и *Android*-разработке, опубликованные в мае 2024 г. на *HH.ru*​. Такой подход позволил выявить перечень ключевых навыков и знаний, которые работодатели ожидают от начинающих мобильных разработчиков, а также понять особенности процесса найма.

***iOS*** — В исследовании ([Яндекс Практикум, 10 июня 2024](#Яндекс_Практикум_2024_10_06_iOS)) было установлено, что работодатели чаще всего требуют от начинающих *iOS*-разработчиков следующие компетенции:

* Владение языком *Swift*.
* Опыт работы с системой контроля версий *Git*.
* Знание принципов объектно-ориентированного программирования (ООП).
* Навыки работы в *Xcode*.
* Знакомство с фреймворком *UIKit*.
* Знание архитектурного паттерна *MVVM*.

Помимо этого, от кандидатов ожидается понимание концепции разделения кода на слои и общее представление о шаблонах проектирования, таких как *MVVM*, *VIPER*, *MVI*, *Singleton*, *Delegate* и *Factory*. Знание *SwiftUI* и *Combine* является желательным, однако их необходимость зависит от специфики компании и проекта: одни активно внедряют *SwiftUI* в новых разработках, в то время как другие продолжают использовать *UIKit*.

Также для начинающих специалистов *soft skills* имеют значение, сопоставимое с техническими знаниями. Отсутствие способности к логическому рассуждению или проявление агрессивного поведения во время интервью существенно снижает шансы на успешное трудоустройство.

***Android*** — В исследовании ([Яндекс Практикум, 3 июля 2024](#Яндекс_Практикум_2024_03_06_Android)) было установлено, что работодатели чаще всего требуют от начинающих *Android*-разработчиков следующие компетенции:

* Владение языками *Kotlin* и *Java*.
* Знание *Android* *SDK*.
* Умение работать с системой контроля версий *Git*.
* Понимание принципов ООП.
* Знание архитектурных паттернов *MVVM* и *MVI*.

Кроме того, ценятся навыки работы со *Spring* *Framework*, понимание клиент-серверного взаимодействия, опыт работы с базами данных (например, *SQLite* и *Room*), а также базовые знания алгоритмов, кэширования и управления памятью.

Особое внимание уделяется умению находить и исправлять ошибки, работе с версткой (*XML*, *Jetpack* *Compose*, *HTML*) и использованию *Compound* *View*. При этом в современных проектах всё чаще применяется *Jetpack* *Compose*, хотя знание *XML* остаётся актуальным. Среди архитектурных требований выделяются умения работы с *ViewModel* и внедрения зависимостей посредством *Koin* или *Dagger*. Работодатели также ожидают базовых знаний многопоточности и функционального программирования, что подчёркивает важность владения *Coroutines* или *RxJava*.

Также для начинающих специалистов *soft skills* имеют значение, сопоставимое с техническими знаниями. Работодатели отдают предпочтение кандидатам, которые проявляют инициативу, проявляют искренний интерес к работе, умеют анализировать информацию и подходят к задачам внимательно. В то же время наиболее нежелательными качествами считаются агрессивность и раздражительность.

**1.2.2. Исследование *Hanna Samer Odeh***

В исследовании ([Hanna Samer Odeh, 2024](#Hanna_Samer_Odeh_2024_Android_Jobs)) был проведён анализ 300 вакансий *Android*-разработчиков из шести стран: США, Канады, Великобритании, Австралии, ОАЭ и Индии (по 50 вакансий из каждой страны). Вакансии были собраны с 44 сайтов по поиску работы, включая такие платформы, как *LinkedIn*, *Indeed* и *Seek*. Методология исследования основывалась на текстовом анализе объявлений о вакансиях, что позволило выделить наиболее часто упоминаемые требования к кандидатам. Исследование фокусировалось на выявлении как технических, так и мягких навыков (*soft skills*), востребованных на рынке труда *Android*-разработчиков.

Среди наиболее востребованных технических навыков, упоминаемых в более чем 30 % проанализированных объявлений, были выявлены:

* Владение языками программирования *Kotlin* и *Java*;
* Навыки обеспечения производительности *Android*-приложений;
* Опыт работы с *RESTful* *API* и взаимодействием с сетевыми сервисами;
* Владение системой контроля версий *Git*;
* Знание *Android* *SDK*;
* Навыки модульного тестирования (*Unit* *Testing*);
* Проектирование интерфейсов *Android*-приложений, в том числе использование *XML* для разработки активностей;
* Владение архитектурным шаблоном *MVVM*;
* Знание *Agile*-методологий разработки программного обеспечения;
* Опыт публикации приложений в *Google* *Play* *Store*.

Помимо технических компетенций, значительное внимание уделяется и мягким навыкам. В частности, наибольшее значение придается:

* Умению работать в команде (*Collaboration* — 47 %, *Teamwork* — 28 % вакансий);
* Коммуникативным навыкам (*Communication* — 29 %);
* Способности к решению проблем (*Problem*-*Solving* — 21 %).

## 1.3. Выводы и пробелы в исследованиях

Анализ существующих исследований показал, что, несмотря на высокий интерес к рынку труда в *IT*-сфере, детальное изучение наиболее востребованных технологий и инструментов для *iOS*- и *Android*-разработчиков до сих пор остаётся ограниченным.

Так, в целом ряде работ ([Apatsidis *et al.*, 2021](#Apatsidis_Georgiou_Mittas_Angelis_2021); [Barousse, n.d.](#Barousse_n_d_Data_Job_Pipeline_Airflow); [Cummings *et al.*, 2023](#Cummings_Janicki_Matthews_2023); [JetBrains, 2024](#JetBrains_2024_State_of_Developer_Ecosys); [Kang *et al.*, 2020](#Kang_Park_Shin_2020); [Stack Overflow, 2024](#Stack_Overflow_2024_Annual_Dev_Survey); [Ternikov *et al.*, 2020](#Ternikov_Aleksandrova_2020)) представлена ценная информация о глобальных потребностях рынка, однако мобильные разработчики не выделены в отдельный срез и рассматриваются как часть общей выборки, из-за чего сложно выделить их специфические требования.

Исследование, проведённое *Hanna* *Samer* *Odeh* ([2024](#Hanna_Samer_Odeh_2024_Android_Jobs)), предоставляет детализированное представление о востребованных навыках и технологиях для *Android*-разработчиков на межрегиональном уровне (6 стран), однако аналогичного комплексного исследования, охватывающего *iOS*-разработчиков, не существует на момент написания работы.

Исследования Яндекс Практикума ([2024, 10 июня](#Яндекс_Практикум_2024_10_06_iOS); [2024, 3 июля](#Яндекс_Практикум_2024_03_06_Android)) обладают высокой практической ценностью, однако они ограничены анализом российского рынка труда. С одной стороны, можно предположить, что требования к *iOS*- и *Android*-разработчикам между различными странами могут быть схожи — подтверждением этому служит обнаруженная в исследовании ([Hanna Samer Odeh, 2024](#Hanna_Samer_Odeh_2024_Android_Jobs)) сильная положительная корреляция (≥ 0,7) между некоторыми навыками *Android*-разработчиков в разных странах. С другой стороны, не исключено, что локальные особенности и различия в корпоративных культурах могут приводить к вариациям в наборе необходимых навыков.

Таким образом, данное исследование направлено на устранение пробелов в существующих знаниях, позволяя более детально изучить и сопоставить востребованные навыки и инструменты мобильной разработки в регионах Европы и Северной Америки.

2. ВЫБОР ИНСТРУМЕНТОВ И ПОДХОДОВ

## 2.1. Выбор языка программирования

В качестве основного языка выбран *Python*, поскольку он обеспечивает быструю прикладную разработку для задач обработки текстов вакансий, очистки и нормализации данных, базовой статистической проверки гипотез и интеграции с внешними *API*. Богатая экосистема (*pandas/NumPy/SciPy*, инструменты для работы с *JSON/CSV/Parquet*, удобные клиенты к *API*) позволяет собрать весь конвейер — от загрузки данных до выгрузки табличных результатов и визуализаций — в едином стеке, что упрощает воспроизводимость. Дополнительно, *Python* имеет низкий порог вхождения и широкую поддержку сообщества, а используемые инструменты — открытые и бесплатные.

## 2.2. Ключевые библиотеки *Python* и их роль

Ниже перечислены библиотеки, составляющие ядро инструментария исследования:

* *Pandas* — базовый каркас табличной обработки: загрузка/очистка, объединения, группировки, подготовка финальных таблиц и выгрузок;
* *SerpApi* — сбор вакансий из *Google* *Jobs*;
* *Langdetect* — определение языка текста вакансий;
* *OpenAI* — перевод текстов вакансий на английский и автоматизированное извлечение технологий с помощью *API* *GPT-4o*;
* *RapidFuzz* — валидация качества извлечения с использованием метрики *Token* *Set* *Ratio*;
* *OpenPyXL* — оформление *Excel*-выгрузок (ширины/шрифты/выравнивание, служебные утилиты колонок);
* *NumPy* — векторные и численные операции, вспомогательные агрегации.

## 2.3. Сбор вакансий с помощью *SerpApi*

Для получения широкой и диверсифицированной выборки объявлений использован интерфейс *Jobs on Google Search* (далее — *Google Jobs*), который агрегирует вакансии с сайтов работодателей и крупных джоб-бордов и отображает их в едином поисковом интерфейсе ([Google, 2025](#Google_2025_Job_posting); [Google, n.d.](#Google_n_d_have_your_job_postings)). Это обеспечивает широкое покрытие источников при единой точке входа. Однако *Google* не предоставляет публичного *API* для извлечения агрегированных результатов из этого интерфейса; официальные решения ориентированы на публикацию и индексацию вакансий (структурированная разметка *JobPosting*, *Indexing* *API*) или на поиск по собственным вакансиям организаций через *Cloud* *Talent* *Solution* ([Clifford, 26 июня 2018](#Clifford_2018_26_июня_Introducin_the_ind); [Google Cloud, n.d.](#Google_cloud_n_d_Job_search); [Google, 2025](#Google_2025_Job_posting)).

В этой конфигурации оптимальным решением стал специализированный сервис *SerpApi*, предоставляющий программный доступ к результатам *Google Jobs* в формате *JSON* с поддержкой пагинации по *«next\_page\_token»*, геопараметров *«location»/«gl»* и языкового параметра *«hl»*. Ключевую роль играет параметр *«q»*: он позволяет формулировать точные поисковые запросы (например, *«iOS developer»* или *«Android developer»*), что концентрирует выборку на более релевантных объявлениях. Дополнительно *SerpApi* берёт на себя инфраструктурные сложности сбора данных (браузерный/динамический рендеринг, прокси-инфраструктура, обработка *CAPTCHA*) и публикует открытый статус-монитор доступности и задержки ответа (*latency*) *API*, что упрощает контроль качества сбора данных. ([SerpApi, n.d. -a](#SerpApi_n_d_Google_Search_API); [SerpApi, n.d. -b](#SerpApi_n_d_Google_Jobs_API))

С точки зрения альтернативных каналов данных, прямой сбор с частных платформ (например, *LinkedIn*) ограничен условиями пользовательских соглашений, которые, как правило, запрещают автоматизированный сбор данных (*scraping*, *crawling*, *automated* *collection*) и подтверждаются соответствующей судебной практикой (характерный пример — *hiQ Labs, Inc. v. LinkedIn Corp.*). Поэтому опора на агрегатор *Google Jobs* через специализированный посредник, работающий на уровне результатов поиска (*SerpApi*), позволяет снизить правовые и технические риски, присущие прямому скрейпингу частных площадок. ([Fried, 15 июня 2021](#Fried_2021_15_июня_Supreme_Court); [Haller, 28 марта 2025](#Haller_2025_28_марта_Scraping_Isnt_sourc))

## 2.4. Определение языка вакансий с помощью *Langdetect*

Для определения языка вакансий была выбрана библиотека *Python Langdetect*. Хотя существуют более современные решения, такие как *fastText*, *Langdetect* показывает достаточно высокую точность, что подтверждено в сравнительном исследовании ([*ModelPredict*, 2021](#ModelPredict_2021_Comparison_of_language)). Несмотря на более низкую скорость, в условиях офлайн-анализа и умеренного объёма данных её производительности достаточно. Простота интеграции делает её удобным инструментом для исследовательских задач; дополнительным плюсом стало предыдущее знакомство с библиотекой, что сократило время внедрения.

## 2.5. Выбор большой языковой модели (*LLM*)

​Для задач исследования требовались качественный перевод вакансий на английский и точное извлечение/нормализация упоминаний технологий при строгом соблюдении форматам вывода. В качестве основной модели использована *gpt-4o-2024-11-20* от *OpenAI* — последняя стабильная ревизия модели *GPT-4o* на момент проведения работы. Сводная оценка независимых источников (пользовательские арены и *capability*-бенчмарки) подтверждает, что *GPT-4o* стабильно входит в фронтир-класс и демонстрирует конкурентоспособность относительно *Gemini 1.5 Pro* и других флагманов; относительные преимущества распределяются по наборам, но в целом *GPT-4o* удерживает уровень универсальной модели, подходящей для прикладных задач *NLU*, перевода и структурированного извлечения информации. ([Artificial Analysis, n.d](#Artificial_Analysis_n_d_GPT_4o); [Chou, Dunlap, Chiang, Sheng, Zheng, Angelopoulos, Darrell, Stoica, & Gonzalez, 27 июня 2024](#Chou_Dunlap_Chiang_et_al_2024_27_июня); [LLM Stats, n.d.](#LLM_Stats_n_d_GPT_4o_vs_Gemini); [LMArena, 2025](#LMArena_2025_Vision_Arena); [Xu, Mai, & Liang, 20 марта 2025](#Xu_Mai_Liang_20_марта_2025_HELM))

**2.5.1. Формирование промпта для перевода вакансий на английский**

Структура промпта опирается на результаты ([Gao, Wang & Hou, 2023](#Gao_Wang_Hou_2023)), где эмпирически сравнивались разные формулировки для машинного перевода с использованием *ChatGPT (GPT-3.5)* относительно *Google Translate* и *DeepL*. Показано, что корректная формулировка промпта может заметно повышать качество перевода; в данной работе адаптирована наиболее эффективная версия промпта из исследования, но с использованием *GPT-4o*.

## 2.6. Нечёткое сопоставление

Нечёткое сопоставление (*fuzzy matching*) применяется для сравнения текстов с учётом орфографических вариаций, перестановок и незначительных расхождений. В отличие от строгого сопоставления строк, этот подход позволяет выявлять смысловую близость при частичных совпадениях. ([DataCamp, 2025](#DataCamp_2025_Fuzzy_String_Matching))

**2.6.1. *Token Set Ratio***

Одной из наиболее эффективных метрик для сравнения текстов в рамках нечёткого сопоставления является *Token Set Ratio*. Она основана на классическом методе — расстоянии Левенштейна (минимальное число операций вставки/удаления/замены для преобразования одной строки в другую), но отличается тем, что ([DataCamp, 2025](#DataCamp_2025_Fuzzy_String_Matching)):

1. Выполняется токенизация — разбиение текста на отдельные слова (токены);
2. Удаляются дубликаты токенов;
3. Сравнение проводится без учёта регистра;
4. Расстояние вычисляется по нормализованным множествам токенов.

Метрика уместна для вакансий, где одна и та же информация формулируется по-разному при сохранении смысла.

**2.6.2. *Rapidfuzz***

Для реализации *Token Set Ratio* в *Python* была выбрана библиотека *rapidfuzz*. Она представляет собой высокопроизводительную альтернативу *fuzzywuzzy*, обеспечивая большую точность и скорость вычислений за счёт оптимизированных алгоритмов на языке *C++* ([RapidFuzz, n.d.](#RapidFuzz_n_d_GitHub)).

3. МЕТОДИКА ИССЛЕДОВАНИЯ

## 3.1. Подготовка и сбор данных

Чтобы упростить последующий сбор и обработку, данные о странах (полный список стран см. в [Приложении 1](#Приложение_1_Полный_список_стран).) были оформлены в виде *CSV*-файла «*location\_domain\_table.csv*» ([Ireev, 2025a](#Ireev_I_2025_a_location_domain_table_csv)). В этот файл были включены следующие столбцы:

* *Location* — название страны;
* *Region* — регион, к которому относится страна (*Europe* или *Northern* *America*);
* *EU Member* — флаг, указывающий на членство страны в Европейском Союзе (*True/False*);
* *Schengen Agreement* — флаг, указывающий на участие страны в Шенгенском соглашении (*True/False*);
* *Google Domain* — локальный домен *Google*, соответствующий данной стране (например, *google.ee* для Эстонии, *google.co.uk* для Великобритании).

Сбор данных осуществлялся с использованием *Python*-функции «*collect\_jobs\_data»* ([Ireev, 2025b](#Ireev_I_2025b_data_collection_ipynb); [Ireev, 2025c](#Ireev_I_2025с_jobs_helpers_py)), которая выполняла автоматизированные *API*-запросы к *SerpApi.* Все вакансии были собраны в период с «13 января 2025 года 12:08:48 *UTC*» по «13 января 2025 года 12:45:14 *UTC*».

Для максимального охвата уникальных вакансий сбор данных выполнялся дважды для каждой страны с использованием двух различных доменов *Google*:

1. Домен по умолчанию (*google.com*) — для получения результатов глобального поиска;
2. Локальный домен (например, *google.ee* для Эстонии) — для учета локальных особенностей поисковой выдачи.

Вакансии были собраны с помощью следующих поисковых запросов:

* *«iOS developer»* — для поиска вакансий *iOS*-разработчиков.
* *«Android developer»* — для поиска вакансий *Android*-разработчиков.

Все собранные вакансии сохранялись в соответствующие директории в зависимости от домена и типа вакансии (см. [Приложение 2](#_Приложение_2._Директории)). Такое разбиение упростило последующее формирование набора данных.

## 3.2. Создание набора данных

Перед созданием набора данных была выполнена предобработка: удалены нестандартные символы-разделители строк (*\u2028\u2029*), заменены пробелами. После этого, данные были объединены в единый набор (см. [Приложение 3](#_Приложение_3._Итоговый)) с помощью *Python*-функции *«process\_json\_to\_csv»* ([Ireev, 2025b](#Ireev_I_2025b_data_collection_ipynb); [Ireev, 2025c](#Ireev_I_2025с_jobs_helpers_py)). Функция принимала в качестве входных данных пути к директориям с вакансиями (см. [Приложение 2](#_Приложение_2._Директории)), а также данные из файла *«location\_domain\_table.csv»* ([Ireev, 2025a](#Ireev_I_2025_a_location_domain_table_csv)), содержащего информацию о странах.

## 3.3. Первичная очистка данных

Поскольку вакансии собирались через два разных домена, ожидалось, что в данных будет значительное количество дубликатов, что и было подтверждено анализом: при 5 834 строках в наборе данных содержалось всего 2 863 уникальных *Job ID*, что подтверждает наличие дубликатов.

Для удаления дубликатов было установлено следующее правило: если одна и та же вакансия (с одинаковым *Job ID*) встречалась более одного раза в рамках одной страны, приоритет отдавался варианту, полученному через *google.com* (*Python*-функция *«remove\_job\_id\_duplicates»* ([Ireev, 2025d](#Ireev_I_2025d_data_preparation_ipynb); [Ireev, 2025e](#Ireev_I_2025e_data_cleaning_py))).

В ходе очистки была удалена 2 971 строка, и итоговый набор данных уменьшился до 2 863 вакансий. Также были удалены вакансии, содержащие 14 слов или менее.

## 3.4. Определение языка вакансий

Для определения языка вакансий была разработана *Python*-функция *«detect\_language\_with\_confidence»* ([Ireev, 2025d](#Ireev_I_2025d_data_preparation_ipynb)), использующая библиотеку *Langdetect*. Для воспроизводимости результатов был задан параметр *«DetectorFactory.seed = 0»*. Функция выполняла следующие задачи:

* Определяла язык текста для каждой вакансии;
* Возвращала код языка в формате *ISO* (например, *en* — *English*, *fr* — *French* и т. д.);
* Возвращала уровень уверенности в определённом языке (например, 0,999).

Если уровень уверенности модели *Langdetect* в определении языка был ниже 0,99 (что наблюдалось только в 16 случаях), то язык вакансии определялся повторно с помощью *Python*-функции *«chatgpt\_async»* ([Ireev, 2025d](#Ireev_I_2025d_data_preparation_ipynb); [Ireev, 2025f](#Ireev_I_2025f_chat_gpt_py)), которая определяла язык вакансии с помощью *API* *GPT-4o*.

Для определения языка использовался следующий промпт:

*«Detect the language of the text and return ONLY the ISO language code* (*e.g., en, fr, de, etc*.). *Text*: {*Job Description*}*»*.

Где {*Job Description*} — описание вакансии.

Окончательное решение принималось на основе ручной проверки по следующему алгоритму:

* Если выводы *Langdetect* и *GPT-4o* совпадали (например, оба определили текст как английский), то данный язык фиксировался как окончательный.
* Если текст содержал примерно равные доли двух языков, такая вакансия не учитывалась как англоязычная.

## 3.5. Перевод неанглоязычных вакансий на английский

В ходе исследования было выявлено, что 86 % всех вакансий были размещены на английском языке.

Перед переводом была составлена языковая разметка ([Ireev, 2025g](#Ireev_I_2025g_language_map_json)), в которой каждому уникальному *ISO*-коду языка было сопоставлено его наименование. Например:

* *en* → *English*;
* *fr* → *French*;
* *de* → *German* и т. д.

Для перевода вакансий на английский была написана *Python*-функция *«translate\_non\_english\_descriptions»* ([Ireev, 2025d](#Ireev_I_2025d_data_preparation_ipynb); [Ireev, 2025f](#Ireev_I_2025f_chat_gpt_py)), которая формировала динамический промпт в зависимости от языка оригинального текста и отправляла *API*-запрос к модели *GPT-4o*.

Для перевода вакансий на английский язык применялся следующий промпт (подробнее об обосновании выбора промпта см. [раздел 2.5.1.](#Раздел_2_3_1_Формирование_промпта)):

*«This is a {Mapped Language} to English translation, please provide the English translation for this job description: {Job Description}»*

Здесь {*Mapped Language*} — язык оригинального текста, а {*Job Description*} — описание вакансии.

## 3.6. Разделение вакансий на логические части

С целью последующей обработки данных вакансии были разделены на следующие логические части:

1. *Platform*;
2. *Salary*;
3. *Requirements*;
4. *Nice to Have*;
5. *Responsibilities*;
6. *Benefits*.

Такое разделение упрощает извлечение навыков и технологий за счёт сокращения объёма обрабатываемого текста, что, в свою очередь, снижает затраты на использование *API GPT-4o*.

**3.6.1. Валидация**

Для оценки качества автоматического разбиения вакансий на логические части была проведена процедура валидации. Целью данного этапа была проверка точности работы метода, использующего *API GPT-4o*, по сравнению с ручной разметкой.

Для проведения валидации была сформирована случайная выборка из 30 вакансий ([Ireev, 2025j](#Ireev_I_2025j_job_descriptions_sample)). Для обеспечения воспроизводимости эксперимента в процессе отбора использовался параметр *«random\_state = 42»*. Далее вакансии были разделены на логические части двумя способами:

1. Ручная разметка ([Ireev, 2025k](#Ireev_I_2025k_job_descriptions_groud_tru)), выполненная автором, который классифицировал информацию в соответствии с установленными категориями: *Platform*, *Salary*, *Requirements*, *Nice to Have*, *Responsibilities* и *Benefits*;
2. Автоматическая разметка ([Ireev, 2025l](#Ireev_I_2025l_job_descriptions_chatgpt)), выполненная при помощи *Python*-функции *«chatgpt\_async»* ([Ireev, 2025f](#Ireev_I_2025f_chat_gpt_py); [Ireev, 2025h](#Ireev_I_2025h_tests_ipynb)) и *API* *GPT-4o* на основе заранее сформулированного промпта (см. [Приложение 4](#_Приложение_2._Промпт)).

Для оценки степени соответствия двух методов разметки был применён метод нечёткого сопоставления (*fuzzy matching*) с использованием метрики *Token Set Ratio* из библиотеки *rapidfuzz* (подробнее см. [раздел 2.6.](#_2.4._Fuzzy_matching)). В качестве порога значимости была установлена граница 85 %, выше которой автоматическая разметка считается удовлетворительно точной. Метрика *Token Set Ratio* рассчитывалась с помощью *Python*-функции *«compare\_fuzzy\_sections»* ([Ireev, 2025h](#Ireev_I_2025h_tests_ipynb); [Ireev, 2025i](#Ireev_I_2025i_tests_helpers_py)), проходя по каждой вакансии и сравнивая секции, размеченные автором и *GPT-4o*. Затем для каждой вакансии рассчитывалось среднее значение *Token Set Ratio* по всем её секциям.

В результате тестирования были получены следующие средние значения *Token Set Ratio* по секциям:

* *Platform*: 98,33 %
* *Salary*: 99,98 %
* *Requirements*: 98,61 %
* *Nice* *to* *Have*: 93,38 %
* *Responsibilities*: 91,42 %
* *Benefits*: 91,85 %

Все значения *Token Set Ratio* по секциям оказались выше 85 %, а итоговое среднее значение *Token Set Ratio* составило 95,60 %, что значительно превышает установленный порог 85 %. Полученные результаты подтверждают высокую степень схожести автоматической разметки с ручной аннотацией и свидетельствуют о надёжности применяемого метода.

После успешной валидации все вакансии были автоматически разделены на логические части с помощью *Python*-функции *«chatgpt\_async»* ([Ireev, 2025d](#Ireev_I_2025d_data_preparation_ipynb); [Ireev, 2025f](#Ireev_I_2025f_chat_gpt_py)) с использованием *API* *GPT*-*4o*.

**3.6.2. Преобразование секций в столбцы**

После автоматического разбиения вакансий на логические части каждая запись представляла собой сплошной блок текста, в котором секции следовали друг за другом по шаблону «Ключ: Значение» (см. пример в [Приложении 5](#_Приложение_5._Пример)). Для последующего анализа требовалось преобразовать эти выжимки в табличный вид, чтобы каждая логическая часть располагалась в отдельном столбце и могла обрабатываться как самостоятельная переменная.

При помощи *Python*-функции *«extract\_section»* ([Ireev, 2025d](#Ireev_I_2025d_data_preparation_ipynb); [Ireev, 2025i](#Ireev_I_2025i_tests_helpers_py)) и регулярных выражений блок текста разбивался на пары «Ключ: Значение». Ключами выступали фиксированные заголовки (*Platform, Salary, Requirements, Nice to Have, Responsibilities, Benefits*), при этом сохранялся исходный порядок строк значений с учётом возможных маркеров списков. Таким образом, на выходе каждая логическая секция представлена в виде отдельного столбца.

## 3.7. Отбор только *iOS*- и *Android*-вакансий

Для достижения целей исследования требовалось сохранить в выборке только вакансии, строго относящиеся к платформам *iOS* и *Android*. Для этого был проведён предварительный анализ столбца *Platform*, содержащего результаты автоматического определения платформы на основе описаний вакансий, полученных с помощью *GPT-4o*.

В наборе данных было выявлено 71 уникальное значение в поле *Platform*. Число вакансий, строго определённых как *iOS* или *Android*, составило: 1094 вакансии для *iOS* и 1080 вакансий для *Android*, что в сумме дало 2174 вакансии, строго соответствующие требованиям.

Дополнительно были обнаружены вакансии, содержащие упоминания сразу нескольких платформ. Эти вакансии разделялись на две группы:

* Кроссплатформенные позиции, ориентированные на разработку сразу под несколько платформ;
* Вакансии, содержащие требования к разработчикам как *iOS*, так и *Android* в рамках одного объявления, вероятно предназначены для оптимизации процесса найма.

Такие позиции не соответствовали целям сравнения и были удалены как методически «шумовые» или чрезмерно ресурсоёмкие для раздельной обработки.

Отдельно были рассмотрены редкие случаи потенциально пригодных вакансий из группы смешанных платформ. В результате ручной проверки были выявлены две группы вакансий (*«iOS, tvOS»* и *«iOS/macOS»*), каждая из которых включала по 2 позиции. Поскольку они явно относились к экосистеме *iOS*, соответствующие записи были отнесены к категории *iOS*.

У 77 объявлений поле *Platform* содержало значение *«Not mentioned»*. Визуальный анализ показал, что это либо крайне краткие публикации, либо общие описания без технических деталей; в обоих случаях модель *GPT*-*4o* обоснованно не смогла определить платформу. Такие строки были исключены как информационно неполные.

Таким образом, после всех этапов фильтрации в итоговый набор данных вошло 2178 вакансии, относящихся строго к разработке под платформы *iOS* или *Android*.

## 3.8. Извлечение технологий и инструментов

Для последующего извлечения технологий и инструментов были использованы три информационных раздела вакансий — *Requirements*, *Nice* *to* *Have* и *Responsibilities*. Содержимое этих секций было конкатенировано в единый текстовый блок, добавленный в новый столбец *Full* *Requirements*. Это было необходимо, чтобы отправлять всю необходимую информацию о вакансии в одном *API*-запросе к модели *GPT-4o*.

**3.8.1. Валидация автоматического извлечения**

Для оценки качества автоматического извлечения технологий и инструментов из вакансий (*Full Requirements*) была проведена процедура валидации. Целью данного этапа была проверка точности работы метода, использующего *API* *GPT-4o*, по сравнению с ручной разметкой.

Для проведения валидации была сформирована случайная выборка из 40 вакансий. Для обеспечения воспроизводимости эксперимента в процессе отбора использовался параметр *«random\_state = 42»*.  Далее технологии и инструменты извлекались из вакансий двумя способами:

1. Ручная разметка, выполненная автором, который извлекал все упомянутые технологии и инструменты из вакансий;
2. Автоматическая разметка, выполненная при помощи *Python*-функции *«chatgpt\_async»* ([Ireev, 2025f](#Ireev_I_2025f_chat_gpt_py); [Ireev, 2025h](#Ireev_I_2025h_tests_ipynb)) и *API* *GPT-4o* на основе заранее сформулированного промпта ([см. Приложение 6](#_Приложение_6._Промпт)).

Для оценки степени соответствия двух методов разметки был применён метод нечеткого сопоставления (*fuzzy* *matching*) с использованием метрики *Token* *Set* *Ratio* из библиотеки *rapidfuzz* (подробнее см. [раздел 2.6.](#_2.4._Fuzzy_matching)). В качестве порога значимости была установлена граница 85 %, выше которой автоматическая разметка считается удовлетворительно точной. Метрика *Token* *Set* *Ratio* рассчитывалась с помощью *Python*-функции *«compare\_fuzzy\_sections»* ([Ireev, 2025h](#Ireev_I_2025h_tests_ipynb); [Ireev, 2025i](#Ireev_I_2025i_tests_helpers_py)),  которая проходила по каждой вакансии и сравнивала результаты, размеченные автором и *GPT-4o*. Затем рассчитывалось среднее значение *Token* *Set* *Ratio* по всем вакансиям в выборке.

В результате тестирования было получено средние значение *Token* *Set* *Ratio* равное 98,71 %, что значительно превышает установленный порог 85 %. Полученные результаты подтверждают высокую степень схожести автоматической разметки с ручной аннотацией и свидетельствуют о надёжности применяемого метода.

После успешной валидации метод автоматического извлечения технологий и инструментов был применён ко всем вакансиям в наборе данных.

## 3.9. Очистка данных

Несмотря на высокую точность извлечения (*Token Set Ratio* = 98,71 %), крупные языковые модели подвержены «галлюцинациям», т. е. включению в ответ названий технологий, фактически отсутствующих в тексте вакансии. Чтобы исключить подобные артефакты и не завысить частоты отдельных инструментов, был добавлен этап нормализации и фильтрации с двумя целями:

1. привести синонимичные названия до канонических форм в нижнем регистре (например, *«Git»* может встречаться как *«GiT»*, *«g-i-t»* или *«GIT»* и т. д.);
2. удалить термины, не встречающиеся в тексте вакансии.

**3.9.1. Нормализация и удаление несуществующих в тексте технологий**

Для достижения целей был вручную составлен *JSON*-словарь синонимов ([Ireev, 2025m](#Ireev_I_2025m_synonyms_json)), который дополнялся по мере появления новых терминов и вариантов написания уже существующих. Структура словаря выглядит как «каноническая форма → список вариантов написания».

Для нормализации и удаления несуществующих терминов также была разработана *Python*-функция *«remove\_hallucinated»* ([Ireev, 2025d](#Ireev_I_2025d_data_preparation_ipynb); [Ireev, 2025e](#Ireev_I_2025e_data_cleaning_py)), которая использовала *JSON*-словарь синонимов и работает по следующему алгоритму:

1. Приводит термины к нижнему регистру.
2. Канонизирует термины, если находит их вариант в списке альтернативных написаний (например, *«g-i-t»* → *«git»*).
3. Удаляет «несуществующие» термины. Технология считается «реальной», если выполняется хотя бы одно из условий:
   1. Термин (после канонизации) отсутствует среди ключей словаря — в этом случае он сохраняется априори.
   2. Отсутствие в списке проверяемых терминов. Для любых редких или новых слов, которых нет в словаре «проверяемых», допускается априорное сохранение.
   3. Совпадение по *n*-грамме. Хотя бы один вариант написания из *JSON*-словаря синонимов содержится среди n‑грамм текста (n = 1…3).

В результате этапа очистки и нормализации было канонизировано 926 терминов, а 389 несуществующих — удалено.

**3.9.2. Фильтрация и очистка**

Далее был сформирован список уникальных технологий *«tech\_counts»* ([Ireev, 2025d](#Ireev_I_2025d_data_preparation_ipynb)), включающий все технологии, упоминавшиеся два раза и более.

Также был создан *JSON*-список терминов ([Ireev, 2025n](#Ireev_I_2025n_remove_list_json)), исключённых из *«tech\_counts»* ввиду их чрезмерной обобщённости, поскольку их наличие не добавляло ценности (например, *Android*, *CI/CD*, *UI* *tests* и т. д.).

## 3.10. Группировка технологий по категориям

Далее, для повышения информативности результатов, все 213 уникальных технологий были вручную распределены между 28 категориями в виде *JSON*-файла *«key\_values»* ([Ireev, 2025o](#Ireev_I_2025o_key_values_json)), данные в котором представлены в формате «Ключ: Значение». Для группировки технологий по категориям была разработана *Python*-функция *«categorize»* ([Ireev, 2025d](#Ireev_I_2025d_data_preparation_ipynb); [Ireev, 2025e](#Ireev_I_2025e_data_cleaning_py)), которая использовала указанный *JSON*-файл: каждый уникальный ключ становился новой колонкой, содержащей соответствующие значения.

Поскольку на данном этапе все данные находились в нижнем регистре, следующим этапом стало приведение всех значений к эталонному виду, например: *«git → Git»*, *«aws → Amazon Web Services (AWS)»* и т. д. Для этого был вручную составлен *JSON*-словарь *«map»* ([Ireev, 2025p](#Ireev_I_2025p_map_json)). Также была разработана *Python*-функция *«fix\_casing»* ([Ireev, 2025d](#Ireev_I_2025d_data_preparation_ipynb); [Ireev, 2025e](#Ireev_I_2025e_data_cleaning_py)), которая с помощью словаря *«map»* приводила названия технологий к эталонному виду.

## 3.11. Финальное преобразование данных

Для экономии вычислительных ресурсов набор данных был отфильтрован с помощью *Python*‑функции *«filtered\_data»* ([Ireev, 2025e](#Ireev_I_2025e_data_cleaning_py); [Ireev, 2025q](#Ireev_I_2025q_data_analysis_ipynb)), в результате чего остались только колонки, необходимые для анализа и визуализации: 28 колонок с технологиями и инструментами, а также:

* *Location*— название страны;
* *Region*— регион (*Europe* или *Northern* *America*);
* *Job ID* — уникальный идентификатор вакансии;
* *Platform*— тип вакансии (*Android* или *iOS*).

Затем данные были преобразованы с помощью функции *«wide\_long»* ([Ireev, 2025e](#Ireev_I_2025e_data_cleaning_py); [Ireev, 2025q](#Ireev_I_2025q_data_analysis_ipynb)) из «широкого» формата (32 колонки, 2178 строк) в «длинный» (6 колонок, 12081 строк). В результате 28 колонок с технологиями были преобразованы в две новые переменные:

* *Category*— отражает исходные тематические заголовки (например, *«ci\_cd»*, *«languages»*);
* *Technology*— отдельные элементы технологического стека, извлечённые из значений соответствующих колонок (например, *Bamboo*, *Kotlin* и т. д.).

## 3.12. Формат представления результатов

Результаты представлены в табличном виде отдельно по каждой тематической категории переменной *Category* и по платформам (*Android*, *iOS*) в [Приложении 7](#_Приложение_7._Результаты). Категории упорядочены в алфавитном порядке; в основной части приводится сводный «Итоговый блок», объединяющий наиболее часто упоминаемые технологии вне зависимости от категории. В «Итоговый блок» включены 95 % упоминаний; доли < 4,5 % (≈ 5 %) опущены. Для каждой платформы формируются отдельные таблицы единого формата:

***Technology (EU) | EU count | EU % | Technology (NA) | NA count | NA %***

* *EU (Europe) / NA (Northern America)* — регионы Европа и Северная Америка;
* *EU count / NA count* — количество вакансий, в которых упомянута соответствующая технология или инструмент;
* *EU % / NA %* — доля таких вакансий в пределах соответствующей регионально-платформенной группы.

Всего рассматриваются четыре независимые группы, для каждой из которых формируется собственная процентная шкала, нормированная до 100 %:

* *EU + Android*
* *EU + iOS*
* *NA + Android*
* *NA + iOS*

Воспроизводимый код, использованный для генерации таблиц с результатами *«all\_tables.xlsx»* ([Ireev, 2025t](#Ireev_I_2025t_all_tables_xlsx)), представлен в виде *Python*-функций *«make\_table»* и *«to\_excel»* ([Ireev, 2025e](#Ireev_I_2025e_data_cleaning_py); [Ireev, 2025q](#Ireev_I_2025q_data_analysis_ipynb)). Набор данных, использованный для создания таблиц и визуализаций, представлен в виде *CSV*-файла ([Ireev, 2025s](#Ireev_I_2025s_df_filtered_long_csv)).

4. РЕЗУЛЬТАТЫ

Перед прочтением данного раздела автор рекомендует ознакомиться с «3.12. Формат представления результатов» в котором подробно описаны принципы формирования и представления данных. Результаты исследования представлены таблицами; детальные результаты по каждой категории вынесены в [Приложение 7](#_Приложение_7._Результаты), в основной части приведён сводный «Итоговый блок». Интерактивные визуализации для детального самостоятельного изучения данных доступны в дашборде *Tableau* ([Ireev, 2025r](#Ireev_I_2025r_tableau_dashboard)).

## 4.1. Итоговый блок

Категория *«total»* охватывает наиболее часто упоминаемые технологии вне привязки к тематическим категориям и отражает общую структуру востребованности ключевых элементов стека по платформам. В таблицах 1–2 представлены результаты по платформам *Android* и *iOS*.

**Таблица 1.** **Итоговый блок для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Kotlin | 600 | 87,21 | Kotlin | 208 | 71,97 |
| Java | 368 | 53,49 | Java | 186 | 64,36 |
| Git | 207 | 30,09 | Android SDK | 111 | 38,41 |
| Jetpack Compose | 198 | 28,78 | Git | 105 | 36,33 |
| Android SDK | 181 | 26,31 | REST API | 98 | 33,91 |
| REST API | 161 | 23,40 | Jetpack Compose | 82 | 28,37 |
| Agile | 158 | 22,97 | Agile | 82 | 28,37 |
| Model-View-ViewModel (MVVM) | 157 | 22,82 | Model-View-ViewModel (MVVM) | 74 | 25,61 |
| Android Studio | 124 | 18,02 | Android Studio | 63 | 21,80 |
| Kotlin Coroutines | 95 | 13,81 | Dagger | 53 | 18,34 |
| Scrum | 81 | 11,77 | Kotlin Coroutines | 40 | 13,84 |
| C++ | 79 | 11,48 | RxJava | 39 | 13,49 |
| Model-View-Presenter (MVP) | 68 | 9,88 | Retrofit | 37 | 12,80 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Google Play Store / Google Play Console | 66 | 9,59 | Scrum | 35 | 12,11 |
| Gradle | 65 | 9,45 | Gradle | 32 | 11,07 |
| RxJava | 60 | 8,72 | Model-View-Presenter (MVP) | 29 | 10,03 |
| Jira | 57 | 8,28 | C++ | 29 | 10,03 |
| Room | 54 | 7,85 | Clean Architecture | 24 | 8,30 |
| Clean Architecture | 53 | 7,70 | C | 23 | 7,96 |
| Test-Driven Development (TDD) | 52 | 7,56 | Room | 23 | 7,96 |
| Retrofit | 49 | 7,12 | Espresso | 23 | 7,96 |
| Firebase | 45 | 6,54 | Jira | 22 | 7,61 |
| Dagger | 44 | 6,40 | Google Play Store / Google Play Console | 21 | 7,27 |
| Android NDK | 44 | 6,40 | GraphQL | 21 | 7,27 |
| Espresso | 43 | 6,25 | Model-View-Controller (MVC) | 20 | 6,92 |
| Kotlin Flow | 36 | 5,23 | JUnit | 20 | 6,92 |
| JUnit | 35 | 5,09 | Firebase | 20 | 6,92 |
| Hilt | 34 | 4,94 | React Native | 20 | 6,92 |
| Jenkins | 32 | 4,65 | JavaScript | 20 | 6,92 |
| Confluence | 32 | 4,65 | Hilt | 19 | 6,57 |
| SQLite | 31 | 4,51 | Jenkins | 17 | 5,88 |
|  |  |  | Notifications | 15 | 5,19 |
|  |  |  | Model-View-Intent (MVI) | 15 | 5,19 |
|  |  |  | Test-Driven Development (TDD) | 15 | 5,19 |
|  |  |  | Flutter | 15 | 5,19 |
|  |  |  | Mockito | 14 | 4,84 |
|  |  |  | GitLab | 13 | 4,50 |

**Таблица 2.** **Итоговый блок для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Swift | 664 | 89,37 | Swift | 230 | 84,87 |
| SwiftUI | 283 | 38,09 | Objective-C | 148 | 54,61 |
| Objective-C | 276 | 37,15 | Git | 97 | 35,79 |
| Git | 228 | 30,69 | Agile | 90 | 33,21 |
| UIKit | 190 | 25,57 | SwiftUI | 90 | 33,21 |
| REST API | 185 | 24,90 | REST API | 74 | 27,31 |
| Model-View-ViewModel (MVVM) | 158 | 21,27 | Xcode | 71 | 26,20 |
| Agile | 146 | 19,65 | Model-View-ViewModel (MVVM) | 63 | 23,25 |
| Xcode | 142 | 19,11 | Core Data | 53 | 19,56 |
| Core Data | 102 | 13,73 | UIKit | 47 | 17,34 |
| App Store / App Store Connect | 71 | 9,56 | Model-View-Controller (MVC) | 43 | 15,87 |
| Model-View-Controller (MVC) | 70 | 9,42 | Apple Combine | 38 | 14,02 |
| CocoaPods | 70 | 9,42 | Scrum | 34 | 12,55 |
| Scrum | 69 | 9,29 | Core Animation | 30 | 11,07 |
| Core Animation | 67 | 9,02 | JavaScript | 26 | 9,59 |
| Apple Combine | 63 | 8,48 | Notifications | 26 | 9,59 |
| RxSwift | 62 | 8,34 | App Store / App Store Connect | 22 | 8,12 |
| fastlane | 50 | 6,73 | Cocoa Touch | 21 | 7,75 |
| View-Interactor-Presenter-Entity-Router (VIPER) | 47 | 6,33 | iOS SDK | 21 | 7,75 |
| Test-Driven Development (TDD) | 46 | 6,19 | Apache Subversion (SVN) | 18 | 6,64 |
| Jenkins | 44 | 5,92 | GraphQL | 18 | 6,64 |
| Swift Package Manager | 43 | 5,79 | Model-View-Presenter (MVP) | 16 | 5,90 |
| Firebase | 40 | 5,38 | Jira | 16 | 5,90 |
| XCTest | 40 | 5,38 | XCTest | 16 | 5,90 |
| Cocoa Touch | 40 | 5,38 | RxSwift | 15 | 5,54 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Grand Central Dispatch (GCD) | 39 | 5,25 | View-Interactor-Presenter-Entity-Router (VIPER) | 15 | 5,54 |
| Model-View-Presenter (MVP) | 38 | 5,11 | CocoaPods | 14 | 5,17 |
| Jira | 37 | 4,98 | React Native | 14 | 5,17 |
| iOS SDK | 35 | 4,71 | C++ | 13 | 4,80 |
| Core ML | 34 | 4,58 | Test-Driven Development (TDD) | 13 | 4,80 |

Для ***Android*** первый ранг стабильно у *Kotlin*, за ним — *Java*. Далее идёт слой повсеместно используемых инструментов и практик: контроль версий *Git*, базовые платформенные компоненты (*Android* *SDK*), декларативный *UI*-фреймворк *Jetpack* *Compose* и взаимодействие с сервером через *REST* *API*. В верхних позициях также процессные подходы (*Agile*, *Scrum*), архитектурные паттерны (прежде всего *MVVM*; *MVP* — реже), *IDE* (*Android* *Studio*), *DI*-фреймворки (*Dagger*; *Hilt* заметнее в Северной Америке), библиотеки асинхронности (*Kotlin* *Coroutines*; *Flow* встречается точечно, главным образом в Европе), сетевой стек (*Retrofit*), инструменты тестирования (*JUnit*, *Espresso*), система сборки (*Gradle*), хранение данных (*Room*, *SQLite*) и элементы публикации (*Google Play Store / Google Play Console*). Профиль указывает на сосуществование классического (*Java*, *RxJava*, местами *MVC*) и современного стека (*Kotlin*, *Coroutines/Flow*, *Compose*, *Hilt*); низкоуровневые *NDK/C++* — реже.

Для ***iOS*** лидирует *Swift* при заметном сосуществовании с *Objective-C*; в ядре также находятся *Git*, процессные практики (*Agile*, *Scrum*), взаимодействие с сервером через *REST* *API*, *IDE* (*Xcode*), *UI*-фреймворки (*SwiftUI*, *UIKit*), а также долговременное хранилище (*Core* *Data*). Среди архитектурных подходов выделяются *MVVM* и *MVC*; представлены реактивные библиотеки (*Combine*, *RxSwift*), инструменты управления зависимостями (*CocoaPods*, *Swift* *Package* *Manager*), модульное тестирование (*XCTest*, *TDD*), средства автоматизации публикации и сборки (*fastlane*, *Jenkins*), каналы распространения (*App* *Store* *Connect*), асинхронные механизмы выполнения (*GCD*) и межплатформенная интеграция (точечно *React* *Native*). Картина отражает доминирование *Swift* при поддержке легаси-кода с постепенным переходом к *SwiftUI*.

**В целом** по обеим платформам повторяется общий каркас стека:

* язык платформы
* система контроля версий
* *IDE*
* сетевое взаимодействие (*API*)
* методологии разработки и *PM*-инструменты
* *UI*-фреймворки
* хранение данных
* платформенный *SDK*/базовые компоненты
* архитектура и *DI*
* каналы публикации
* асинхронность/реактивность
* сборка и управление зависимостями
* тестирование
* дополнительные языки
* облачные/бэкенд-сервисы (*BaaS*)
* *CI/CD* и автоматизация
* остальное

Различия — в конкретных технологиях и приоритетах; структура уровней схожа. Кроссплатформенные решения отмечаются точечно и не формируют верхние ранги.

Ниже представлены табличные результаты по «каркасу» категорий для *Android* и *iOS*. Проценты трактуются как средняя доля вакансий соответствующей платформы, в которых упоминается технология: % = (*EU%* + *NA%*) / 2; при отсутствии упоминания в регионе использовался 0. Внутри каждой категории технологии перечислены в порядке убывания. Если категория разбита на два подблока, они приводятся последовательно (например, «архитектура» и *«DI»* в *Android*).

***Android***

* язык платформы — *Kotlin* (79,59 %), *Java* (58,92 %);
* система контроля версий — *Git* (33,21 %);
* *IDE* — *Android* *Studio* (19,91 %);
* сетевое взаимодействие (*API*) — *REST* *API* (28,65 %), *Retrofit* (9,96 %), *GraphQL* (3,63 %);
* методологии разработки и *PM*-инструменты — *Agile* (25,67 %), *Scrum* (11,94 %), *Jira* (7,95 %);
* *UI*-фреймворки — *Jetpack* *Compose* (28,58 %), *React* *Native* (3,46 %), *Flutter* (2,60 %);
* хранение данных — *Room* (7,90 %);
* платформенный *SDK*/базовые компоненты — *Android* *SDK* (32,36 %), *Android* *NDK* (3,20 %);
* архитектура и *DI* — архитектура: *MVVM* (24,21 %), *MVP* (9,96 %), *Clean* *Architecture* (8,00 %), *MVC* (3,46 %), *MVI* (2,60 %); *DI*: *Dagger* (12,37 %), *Hilt* (3,29 %);
* каналы публикации — *Google Play Store / Google Play Console* (8,43 %);
* асинхронность/реактивность — *Kotlin* *Coroutines* (13,82 %), *RxJava* (11,11 %), *Kotlin* *Flow* (2,62 %);
* сборка и управление зависимостями — *Gradle* (10,26 %);
* тестирование — *Espresso* (7,11 %), *TDD* (6,38 %), *JUnit* (6,00 %);
* дополнительные языки — *C++* (10,75 %), *C* (3,98 %), *JavaScript* (3,46 %);
* облачные/бэкенд-сервисы (*BaaS*) — *Firebase* (6,73 %);
* *CI/CD* и автоматизация — *Jenkins* (2,94 %);
* остальное — *Notifications* (2,60 %).

***iOS***

* язык платформы — *Swift* (87,12 %), *Objective-C* (45,88 %);
* система контроля версий — *Git* (33,24 %), *SVN* (3,32 %);
* *IDE* — *Xcode* (22,66 %);
* сетевое взаимодействие (*API*) — *REST* *API* (26,10 %), *GraphQL* (3,32 %);
* методологии разработки и PM*-*инструменты — *Agile* (26,43 %), *Scrum* (10,92 %), *Jira* (2,95 %);
* *UI*-фреймворки — *SwiftUI* (35,65 %), *UIKit* (21,45 %), *Core* *Animation* (10,04 %), *Cocoa* *Touch* (6,56 %), *React* *Native* (2,58 %);
* хранение данных — *Core* *Data* (16,64%);
* платформенный *SDK*/базовые компоненты — *iOS* *SDK* (3,88 %);
* архитектура и *DI* — *MVVM* (22,26 %), *MVC* (12,64 %), *VIPER* (5,94 %), *MVP* (5,51 %);
* каналы публикации — *App Store / App Store Connect* (8,84 %);
* асинхронность/реактивность — *Combine* (11,25 %), *RxSwift* (6,94 %), *GCD* (2,62 %);
* сборка и управление зависимостями — *CocoaPods* (7,29 %), *Swift* *Package* *Manager* (2,90 %);
* тестирование — *XCTest* (5,64 %), *TDD* (3,10 %);
* дополнительные языки — *JavaScript* (4,79 %);
* облачные/бэкенд-сервисы (*BaaS*) — *Firebase* (2,69 %);
* *CI/CD* и автоматизация — *fastlane* (3,37 %), *Jenkins* (2,96 %);
* остальное — *Notifications* (4,79 %).

5. ОБСУЖДЕНИЕ И ПРЕДЛОЖЕНИЯ

## 5.1. Связь результатов с теорией и предшествующими исследованиями

Сопоставление «Итогового блока» ([раздел 4.1.](#_4.29._Итоговый_блок)) с выводами обзора литературы, сфокусированного на *iOS*- и *Android*-разработчиках ([раздел 1.2.](#_1.2._Исследования,_фокусирующиеся)), демонстрирует согласованность на концептуальном и технологическом уровнях. Теоретически ожидаемое ядро платформенных навыков — нативный язык + официальный *SDK/IDE* + базовые архитектурные паттерны + контроль версий — в работах по мобильной разработке описывается как устойчивый «каркас» компетенций, слабо зависящий от ситуативных факторов рынка. Эмпирические данные подтверждают модель: на обеих платформах верхние позиции распределения упоминаний в вакансиях формируют эти связки, а вариативность касается главным образом периферийных инструментов.

Сопоставление с предшествующими прикладными исследованиями дополнительно подтверждает валидность картины. Требования, выделенные в работах Яндекс Практикума ([2024, 10 июня](#Яндекс_Практикум_2024_10_06_iOS); [2024, 3 июля](#Яндекс_Практикум_2024_03_06_Android);) на уровне junior (***Android***: *Kotlin/Java*, *Android* *SDK*, *Git*, *MVVM/MVI*, *REST*, *SQLite/Room*, *Coroutines/RxJava*, *Jetpack* *Compose;* ***iOS****: Swift, Git, Xcode, UIKit, MVVM*), напрямую резонируют с верхними позициями «Итогового блока», где те же элементы формируют наблюдаемое ядро требований в описаниях вакансий. Сходную конфигурацию ядра фиксирует и *Hanna* *Samer* *Odeh* ([2024](#Hanna_Samer_Odeh_2024_Android_Jobs)) для *Android*-разработчиков, что дополнительно подтверждает воспроизводимость наблюдаемой структуры. Тем самым результаты настоящего исследования расширяют и уточняют выводы предшествующих работ, предоставляя количественно сопоставимую картину частот упоминаний по ключевым элементам стеков в Европе и Северной Америке.

Региональные различия минимальны и касаются главным образом «длинного хвоста» (например, отдельных компонентов экосистемы *Apple*); лидеры по каждой платформе стабильны, что подтверждает обобщаемость каркаса. Следовательно, выявленные закономерности — не артефакт выборки или периода, а репликация платформенной логики формулирования требований, описанной в литературе и подтверждённой межрегиональной проверкой на данных вакансий.

## 5.2. Ограничения

**5.2.1. Ограничения машинного перевода**

Автор исследования не проводил собственную оценку качества перевода для всего массива текстов, что может приводить к определённым рискам недостоверности результатов перевода. Тем не менее, учитывая ранее упомянутое исследование ([Gao, Wang & Hou, 2023](#Gao_Wang_Hou_2023)), демонстрирующее высокую степень адекватности машинного перевода, автор считает, что возможные погрешности незначительно влияют на понимание ключевых слов, касающихся технологий, и не искажают общую суть переводимых текстов.

**5.2.2. Ограничения классификации**

В рамках исследования 213 уникальных технологий были вручную объединены в 28 тематических категорий (см. [раздел 3.10](#_3.10._Группировка_технологий)). Такой подход неизбежно приводит к появлению «пограничных» случаев: отдельные технологии могут выглядеть несколько инородно в рамках выбранной группы.

**5.2.3. Интерпретация «востребованности» через тексты вакансий**

В настоящем исследовании «востребованность» трактуется как относительная частота упоминаний технологий в описаниях вакансий, а не как прямое измерение рыночного спроса. Следовательно, наблюдаемые доли отражают практику оформления объявлений и коммуникационные стандарты работодателей и могут отличаться от фактической интенсивности использования технологий в проектах. Это ограничение следует учитывать при интерпретации результатов и сопоставлении регионов/платформ.

**5.2.4. Риски нормализации и канонизации терминов**

Этап приведения терминов к канонической форме на основе словаря синонимов и функции *«remove\_hallucinated»* (см. раздел 3.9.1) снижает шум, но сохраняет вероятность ошибок: возможна некорректная канонизация редких/новых вариантов написания. Этот фактор может как слегка переоценивать частоты отдельных технологий, так и, напротив, занижать их присутствие.

**5.2.5. Исключение слишком общих терминов**

На этапе очистки данных использовался явный перечень «стоп-терминов» — слишком общих или неконкретных обозначений, не позволяющих надёжно идентифицировать конкретные технологии и инструменты; такие упоминания исключались из подсчётов (см. раздел 3.9.2). Этот приём снижает шум и предотвращает искусственное завышение частот по широким категориям, однако сохраняет риск, что часть пограничных терминов, не попавших в перечень, могла остаться в данных.

## 5.3. Предложения

Представленные ниже практико-ориентированные рекомендации для *Android* и *iOS* опираются на сводные результаты «Итогового блока» и реплицируемый «каркас» компетенций, общий для обеих платформ. Используются только технологии, выявленные в результатах; порядок категорий следует каркасу, а внутри категорий элементы упорядочены по убыванию межрегиональной частоты упоминаний. Процентные значения отражают среднее по Европе и Северной Америке.

Классификация «Основа/Дополнение» основана на двух критериях: (1) межрегиональная распространённость требований в вакансиях и (2) статус платформенного канона и его необходимость для полного цикла разработки, включая поддержку легаси-проектов. Поэтому «Основа» фиксирует минимально необходимый набор для найма и базовой учебной программы, тогда как «Дополнение» указывает направления углубления, специализации и повышения конкурентоспособности (включая кроссплатформенные решения, если они встречаются в данных, но не заменяют платформенное ядро).

**5.3.1. Рекомендации для специалистов *Android***

**Основа**

* язык платформы — *Kotlin* (79,59 %), *Java* (58,92 %);
* система контроля версий — *Git* (33,21 %);
* *IDE* — *Android* *Studio* (19,91 %);
* сетевое взаимодействие (*API*) — *REST* *API* (28,65 %), *Retrofit* (9,96 %);
* методологии разработки и *PM*-инструменты — *Agile* (25,67 %);
* *UI*-фреймворки — *Jetpack* *Compose* (28,58 %);
* хранение данных — *Room* (7,90 %);
* платформенный *SDK*/базовые компоненты — *Android* *SDK* (32,36 %);
* архитектура и *DI* — архитектура: *MVVM* (24,21 %); *DI*: *Dagger* (12,37 %), *Hilt* (3,29 %);
* каналы публикации — *Google Play Store / Google Play Console* (8,43 %);
* асинхронность/реактивность — *Kotlin Coroutines* (13,82 %);
* сборка и управление зависимостями — *Gradle* (10,26 %);
* тестирование — *Espresso* (7,11 %), *JUnit* (6,00 %).

**Дополнение**

* сетевое взаимодействие (*API*) — *GraphQL* (3,63 %);
* методологии разработки и *PM*-инструменты — *Scrum* (11,94 %), *Jira* (7,95 %);
* *UI*-фреймворки — *React* *Native* (3,46 %), *Flutter* (2,60 %);
* платформенный *SDK*/базовые компоненты — *Android* *NDK* (3,20 %);
* архитектура и *DI* — архитектура: *MVP* (9,96 %), *Clean* *Architecture* (8,00 %), *MVC* (3,46 %), *MVI* (2,60 %);
* асинхронность/реактивность — *RxJava* (11,11 %), *Kotlin* *Flow* (2,62 %);
* тестирование — *TDD* (6,38 %);
* дополнительные языки — *C++* (10,75 %), *C* (3,98 %), *JavaScript* (3,46 %);
* облачные/бэкенд-сервисы (*BaaS*) — *Firebase* (6,73 %);
* *CI/CD* и автоматизация — *Jenkins* (2,94 %);
* остальное — *Notifications* (2,60 %).

**5.3.2. Рекомендации для специалистов *iOS***

**Основа**

* язык платформы — *Swift* (87,12 %), *Objective-C* (45,88 %);
* система контроля версий — *Git* (33,24 %);
* *IDE* — *Xcode* (22,66 %);
* сетевое взаимодействие (*API*) — *REST* *API* (26,10 %);
* методологии разработки и *PM*-инструменты — *Agile* (26,43 %);
* *UI*-фреймворки — *SwiftUI* (35,65 %), *UIKit* (21,45 %);
* хранение данных — *Core* *Data* (16,64 %);
* платформенный *SDK*/базовые компоненты — *iOS* *SDK* (3,88 %);
* архитектура и *DI* — архитектура: *MVVM* (22,26 %), *MVC* (12,64 %);
* каналы публикации — *App Store / App Store Connect* (8,84 %);
* асинхронность/реактивность — *Combine* (11,25 %), *GCD* (2,62 %);
* сборка и управление зависимостями — *CocoaPods* (7,29 %), *Swift* *Package* *Manager* (2,90 %);
* тестирование — *XCTest* (5,64 %).

**Дополнение**

* сетевое взаимодействие (*API*) — *GraphQL* (3,32 %);
* методологии и *PM*-инструменты — *Scrum* (10,92 %), *Jira* (2,95 %);
* *UI*-фреймворки — *Core* *Animation* (10,04 %), *Cocoa* *Touch* (6,56 %), *React* *Native* (2,58 %);
* архитектура — *VIPER* (5,94 %), *MVP* (5,51 %);
* асинхронность/реактивность —*RxSwift* (6,94 %);
* дополнительные языки — *JavaScript* (4,79 %);
* облачные/бэкенд-сервисы (*BaaS*) — *Firebase* (2,69 %);
* *CI/CD* и автоматизация — *fastlane* (3,37 %), *Jenkins* (2,96 %);
* остальное — *Notifications* (4,79 %).

ЗАКЛЮЧЕНИЕ

В работе решалась задача выявления наиболее часто упоминаемых технологий в вакансиях мобильных разработчиков для *iOS* и *Android* в Европе и Северной Америке, с последующим представлением консолидированных результатов и практико-ориентированных рекомендаций. Корпус данных был сформирован через *Google Jobs/SerpApi*, очищен от дубликатов и нерелевантных записей, после чего тексты были унифицированы по языку, разбиты на логические секции и аннотированы для извлечения технологий с применением *GPT-4o*; итоговый массив составил 2 178 вакансий, строго относящихся к *iOS/Android*.

Качество автоматических процедур подтверждено на независимых подвыборках: разметка секций показала высокие значения *Token Set Ratio* (например, *Requirements* — 98,61 %), а извлечение технологий — среднее 98,71 %, что превышает заданный порог 85 %. Для аналитики и воспроизводимости результаты приведены в табличном формате по 28 тематическим категориям, а также сведены в «Итоговый блок»; подготовлены выгрузки и интерактивный дашборд *Tableau* ([Ireev, 2025r](#Ireev_I_2025r_tableau_dashboard)).

Эмпирическая картина согласуется с теорией и предшествующими прикладными исследованиями: ядро требований устойчиво и платформенно-специфично (***Android***: *Kotlin/Java*, *Git*, *Android* *Studio*, *Android SDK*, *REST*, *Jetpack* *Compose*, *MVVM*, *Room*; ***iOS***: *Swift/Objective-C*, *Git*, *Xcode*, *iOS SDK*, *REST*, *SwiftUI/UIKit*, *MVVM*, *Core Data*), а межрегиональные различия проявляются преимущественно в «длинном хвосте».

Практическим результатом стали реплицируемые рекомендации «Основа/Дополнение» для обеих платформ, ориентированные на разработчиков и образовательные программы; они построены на сводных межрегиональных частотах и фиксируют минимальный «канон» и направления углубления компетенций.

Ограничения касаются возможных погрешностей машинного перевода, рисков нормализации/канонизации терминов и исключения чрезмерно общих обозначений; эти факторы могут умеренно влиять на частоты редких технологий.

Направления дальнейших исследований логично вытекают из обозначенных границ: (1) динамический анализ во времени (лонгитюд) и расширение географии за пределы ЕС/Северной Америки; (2) дополнительная стратификация по грейдам и типам компаний при наличии надёжной маркировки; (3) валидация результатов альтернативными источниками (описания репозиториев, учебные планы, профильные опросы) и углубление словарей синонимов; (4) сопоставление упоминаний с метриками найма и компенсаций. Эти шаги повысят внешнюю валидность и чувствительность инструментария без изменения базовой логики исследования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Apatsidis, I., Georgiou, K., Mittas, N., & Angelis, L. (2021). A Study of Remote and On-site ICT Labor Market Demand using Job Offers from Stack Overflow. *ResearchGate*. <https://www.researchgate.net/publication/355685512_A_Study_of_Remote_and_On-site_ICT_Labor_Market_Demand_using_Job_Offers_from_Stack_Overflow>

Artificial Analysis. (n.d.). *GPT-4o (Nov ’24) vs Gemini 1.5 Pro (Sep ’24)*. <https://artificialanalysis.ai/models/comparisons/gpt-4o-vs-gemini-1-5-pro>

Barousse, L. (2023, 17 февраля). *I analyzed 3,275,597 jobs to solve THIS* [Видео]. <https://www.youtube.com/watch?v=7G_Kz5MOqps&ab_channel=LukeBarousse>

Barousse, L. (n.d.). *Data\_Job\_Pipeline\_Airflow* [Git Hub репозиторий]. <https://github.com/lukebarousse/Data_Job_Pipeline_Airflow>

Briskman, J. (2024, январь). 2024 State of Mobile: The App Market Bounced Back in 2023 and is Primed for a Record-Breaking 2024. *SensorTower*. <https://sensortower.com/blog/state-of-mobile-2024>

Briskman, J. (2025, февраль). Q4 2024 Digital Market Index: The Digital Economy Closes 2024 with New Records Across Mobile and Advertising. *SensorTower*. <https://sensortower.com/blog/q4-2024-digital-market-index-report>

Brown, L. (2024). New Research Reveals AI Coding Assistants Boost Developer Productivity by 26%: What IT Leaders Need to Know. *IT Revolution*. <https://itrevolution.com/articles/new-research-reveals-ai-coding-assistants-boost-developer-productivity-by-26-what-it-leaders-need-to-know/>

Chou, C., Dunlap, L., Chiang, W.-L., Sheng, Y., Zheng, L., Angelopoulos, A., Darrell, T., Stoica, I., & Gonzalez, J. E. (2024, 27 июня). The Multimodal Arena is Here! *LMSYS Org*. <https://lmsys.org/blog/2024-06-27-multimodal/>

Clifford, Z. (2018, 26 июня). Introducing the Indexing API for job posting URLs. *Google Search Central Blog*. <https://developers.google.com/search/blog/2018/06/introducing-indexing-api-for-job>

Crudu, A. (2025, 6 мая). Cost Variations in Hiring Android Developers Across Different Regions - A Comprehensive Analysis. *MoldStud*. <https://moldstud.com/articles/p-cost-variations-in-hiring-android-developers-across-different-regions-a-comprehensive-analysis>

Crudu, V., & MoldStud Research Team. (2025, 4 января). Growing demand for mobile app developers and career growth. *MoldStud*. <https://moldstud.com/articles/p-growing-demand-for-mobile-app-developers-and-career-growth>

Cummings, J., Janicki, T., & Matthews, K. (2023). Invited Paper: Survey of Technology and Skills in Demand: The 2022 Update. *Journal of Information Systems Education*, 34(2), 243-255. <https://aisel.aisnet.org/jise/vol34/iss2/10>

Data Nerds. (n.d.). *Top Skills & Pay of Data Nerds*. <https://datanerd.tech/>

DataCamp. (2025). *Fuzzy String Matching in Python Tutorial*. <https://www.datacamp.com/tutorial/fuzzy-string-python>

Dixon, N., Goggins, M., Ho, E., Howison, M., Long, J., Northcott, E., Shen, K., & Yeats, C. (2023). Occupational models from 42 million unstructured job postings. *Patterns*. <https://doi.org/10.1016/j.patter.2023.100757>

Fried, I. (2021, 15 июня). Supreme Court orders new review of data scraping in LinkedIn case. *Axios*. <https://www.axios.com/2021/06/15/supreme-court-linkedin-data-scraper>

Gao, Y., Wang, R., & Hou, F. (2023). How to Design Translation Prompts for ChatGPT: An Empirical Study. *arXiv*. <https://arxiv.org/pdf/2304.02182>

Georgakopoulos, S. (2024, 11 апреля). Mobile Developer Population Forecast 2024. *SlashData*. <https://www.slashdata.co/post/mobile-developer-population-forecast-2024>

Global Growth Insights. (2025). *Mobile Applications Market Size, Share, Growth, and Industry Analysis, By Types (Android, iOS, Other), Applications (Games, Social Networking, Healthcare, Books, Music, Navigation, Business, Other) and Regional Insights and Forecast to 2033*. <https://www.globalgrowthinsights.com/market-reports/mobile-applications-market-106315>

Goldsmith, K. (2016, 15 февраля). Spotify Technology Career Steps. *Spotify Engineering*. <https://engineering.atspotify.com/2016/02/spotify-technology-career-steps>

Goldsmith, K. (2016, 8 февраля). Building a technical career path at Spotify. *Spotify Engineering*. <https://engineering.atspotify.com/2016/02/technical-career-path>

Google Cloud. (n.d.). *Job search API Basics (v3)*. <https://cloud.google.com/talent-solution/job-search/v3/docs/basics>

Google. (2025). *Job posting (JobPosting) structured data for Job Search*. <https://developers.google.com/search/docs/appearance/structured-data/job-posting>

Google. (n.d.). *Have your job postings found by millions of job seekers who search on Google every day*. <https://jobs.google.com/about/>

Grand View Research. (n.d.). *Global Mobile Application Market Size & Outlook, 2024-2030*. <https://www.grandviewresearch.com/horizon/outlook/mobile-application-market-size/global>

GSMA. (2024). The Mobile Economy 2024. <https://www.gsma.com/mobileeconomy/wp-content/uploads/2024/02/260224-The-Mobile-Economy-2024.pdf>

Haller, S. (2025, 28 марта). Scraping Isn’t Sourcing: The Hidden Risks of Using Data Extraction Tools. *Tracker* [Пост в интернете]. <https://www.tracker-rms.com/blog/scraping-isnt-sourcing-the-hidden-risks-of-using-data-extraction-tools/>

Hanna, Samer Odeh. (2024). Analysis of the Required Skills by Industry for Android Jobs. *SSRN*. [http://dx.doi.org/10.2139/ssrn.5008188](https://dx.doi.org/10.2139/ssrn.5008188)

Howarth, J. (2025, 18 июня). How Many People Own Smartphones? (2025–2029). *Exploding Topics*. <https://explodingtopics.com/blog/smartphone-stats>

Human Labs. (2025). *Top Mobile App Development Stats 2025*. <https://xhumanlabs.com/2025/02/17/top-mobile-app-development-stats-2023/>

IDC. (2024). *Smartphone Market Insights*. <https://www.idc.com/promo/smartphone-market-share>

Ireev, I. (2025a). *location\_domain\_table.csv* [Файл CSV]. <https://github.com/S0la1re/Research-work-new/blob/main/data/csv/location_domain_table.csv>

Ireev, I. (2025b). *data\_collection.ipynb* [Ноутбук Jupyter]. <https://github.com/S0la1re/Research-work-new/blob/main/notebooks/data_collection.ipynb>

Ireev, I. (2025c). *jobs\_helpers.py*[Исходный код Python]. <https://github.com/S0la1re/Research-work-new/blob/main/src/jobs_tools/jobs_helpers.py>

Ireev, I. (2025d). *data\_preparation.ipynb* [Ноутбук Jupyter]. <https://github.com/S0la1re/Research-work-new/blob/main/notebooks/data_preparation.ipynb>

Ireev, I. (2025e). *data\_cleaning.py* [Исходный код Python]. <https://github.com/S0la1re/Research-work-new/blob/main/src/jobs_tools/data_cleaning.py>

Ireev, I. (2025f). *chat\_gpt.py* [Исходный код Python]. <https://github.com/S0la1re/Research-work-new/blob/main/src/jobs_tools/chat_gpt.py>

Ireev, I. (2025g). language\_map.json [Файл JSON]. <https://github.com/S0la1re/Research-work-new/blob/main/data/json/language_map.json>

Ireev, I. (2025h). *tests.ipynb* [Ноутбук Jupyter]. <https://github.com/S0la1re/Research-work-new/blob/main/notebooks/tests.ipynb>

Ireev, I. (2025i). *tests\_helpers.py* [Исходный код Python]. <https://github.com/S0la1re/Research-work-new/blob/main/src/jobs_tools/tests_helpers.py>

Ireev, I. (2025j). *job\_descriptions\_sample.txt* [Текстовый файл]. <https://github.com/S0la1re/Research-work-new/blob/main/data/ground_truth/job_descriptions_sample.txt>

Ireev, I. (2025k). *job\_descriptions\_ground\_truth.txt* [Текстовый файл]. <https://github.com/S0la1re/Research-work-new/blob/main/data/ground_truth/job_descriptions_ground_truth.txt>

Ireev, I. (2025l). *job\_descriptions\_chatgpt.txt* [Текстовый файл]. <https://github.com/S0la1re/Research-work-new/blob/main/data/ground_truth/job_descriptions_chatgpt.txt>

Ireev, I. (2025m). *synonyms.json* [Файл JSON]. <https://github.com/S0la1re/Research-work-new/blob/main/data/json/synonyms.json>

Ireev, I. (2025n). *remove\_list.json* [Файл JSON]. <https://github.com/S0la1re/Research-work-new/blob/main/data/json/remove_list.json>

Ireev, I. (2025o). *key\_values.json* [Файл JSON]. <https://github.com/S0la1re/Research-work-new/blob/main/data/json/key_values.json>

Ireev, I. (2025p). *map.json* [Файл JSON]. <https://github.com/S0la1re/Research-work-new/blob/main/data/json/map.json>

Ireev, I. (2025q). *data\_analysis.ipynb* [Ноутбук Jupyter]. <https://github.com/S0la1re/Research-work-new/blob/main/notebooks/data_analysis.ipynb>

Ireev, I (2025r). Mobile Development Stack Dashboard [Дашборд Tableau]. <https://public.tableau.com/app/profile/ivan.ireev/viz/mobile_development_stack_dashboard/Dashboard1>

Ireev, I. (2025s). *df\_final\_filtered\_long.csv* [Файл CSV]. <https://github.com/S0la1re/Research-work-new/blob/main/data/csv/df_final_filtered_long.csv>

Ireev, I (2025t). *all\_tables.xlsx* [Файл Microsoft Excel]. <https://github.com/S0la1re/Research-work-new/blob/main/data/all_tables.xlsx>

JetBrains. (2024). State of Developer Ecosystem Report 2024. <https://www.jetbrains.com/lp/devecosystem-2024/>

Kang, T., Park, H., & Shin, S. (2020). Searching, Examining, and Exploiting In-demand Technical (SEE IT) Skills using Web Data Mining. *Software Engineering and Knowledge Engineering*. <https://ksiresearch.org/seke/seke20paper/paper058.pdf>

Khmelevska, J. (n.d.). Mobile App Developer Salary: Employers’ Guide. *Bridge*. <https://bridgeteams.com/blog/mobile-app-developer-salary-employers-guide/>

Levels.fyi. (n.d.). *Compare career levels across companies — Software Engineer: Microsoft, Bolt (EU), Twilio*. <https://www.levels.fyi/?compare=Microsoft,Bolt%20(EU),Twilio&track=Software%20Engineer>

LLM Stats. (n.d.). *GPT-4o vs Gemini 1.5 Pro*. <https://llm-stats.com/models/compare/gpt-4o-2024-08-06-vs-gemini-1.5-pro>

LMArena. (2025). *Vision Arena*. <https://lmarena.ai/leaderboard/vision>

Mandel, M. (2023). EU App Economy: Skills for the Digital Age. *Progressive Policy Institute*. <https://www.progressivepolicy.org/wp-content/uploads/2023/11/EU-App-Economy.pdf>

ModelPredict. (2021). *Comparison of language identification models*. <https://modelpredict.com/language-identification-survey>

Monzo. (n.d.). *Engineering Progression Framework v3.0*. <https://monzo.com/documents/engineering-progression-framework-v3-0.pdf>

Precedence Research. (2025). *Mobile Application Market Size, Share, and Trends 2025 to 2034*. <https://www.precedenceresearch.com/mobile-application-market>

Qadir, S. (2025, 15 апреля). Mobile App Download Statistics & Usage Statistics in 2025. *TekRevol*. <https://www.tekrevol.com/blogs/mobile-app-download-statistics/>

RapidFuzz. (n.d.). *RapidFuzz* [GitHub-репозиторий]. <https://github.com/rapidfuzz/RapidFuzz>

Sensor Tower. (2025). 2025 State of Mobile: Consumers’ $150 Billion Spent on Mobile Highlights Another Record-Setting Year. <https://sensortower.com/blog/2025-state-of-mobile-consumers-usd150-billion-spent-on-mobile-highlights>

SerpApi. (n.d. -a). *Google Search API*. <https://serpapi.com>

SerpApi. (n.d. -b). *Google Jobs API*. <https://serpapi.com/google-jobs-api>

SFIA Foundation. (n.d. -a). *About SFIA*. <https://sfia-online.org/en/about-sfia/about-sfia>

SFIA Foundation. (n.d. -b). *How SFIA works - Levels of responsibility and skills*. <https://sfia-online.org/en/about-sfia/how-sfia-works>

Stack Overflow. (2024). *Annual Developer Survey 2024*. <https://survey.stackoverflow.co/2024>

Statista. (2021). *Forecast number of mobile users worldwide from 2020 to 2025*. <https://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010>

Statista. (2023). *App — Worldwide*. <https://www.statista.com/outlook/amo/app/worldwide>

Statista. (2024). *Number of mobile app downloads worldwide from 2016 to 2023*. <https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads>

Statista. (2025). *App – Europe*. <https://www.statista.com/outlook/amo/app/europe?currency=USD#revenueGrowth_656192>

TechRound. (2024). *The Ultimate Guide To Hire iOS Developers and Android Developers For Mobile App Success*. <https://techround.co.uk/other/the-ultimate-guide-to-hire-ios-developers-and-android-developers-for-mobile-app-success/>

Ternikov, A. A., & Aleksandrova, E. A. (2020). Demand for skills on the labor market in the IT sector. *Business Informatics*, 14(2), 64—83. <https://doi.org/10.17323/2587-814X.2020.2.64.83>

Tzimas, G., Zotos, N., Mourelatos, E., Giotopoulos, K. C., & Zervas, P. (2024). From Data to Insight: Transforming Online Job Postings into Labor-Market Intelligence. *Information*, 15(8), 496. <https://doi.org/10.3390/info15080496>

Xu, J., Mai, Y., & Liang, P. (2025, 20 марта). HELM Capabilities: Evaluating LMs Capability by Capability. *Center for Research on Foundation Models, Stanford University*. <https://crfm.stanford.edu/2025/03/20/helm-capabilities.html>

Yamashita, M., Shen, J. T., Tran, T., Ekhtiari, H., & Lee, D. (2023). JAMES: Normalizing Job Titles with Multi-Aspect Graph Embeddings and Reasoning. *IEEE*. <https://doi.org/10.1109/DSAA60987.2023.10302559>

Zippia. (2024). Job outlook for mobile app developers in the United States. <https://www.zippia.com/mobile-app-developer-jobs/trends/>

Яндекс Практикум. (2024, 10 июня). iOS-разработчики в 2024 году: какие нужны скилы и как проходит процесс найма. *Habr*. <https://habr.com/ru/companies/yandex_praktikum/articles/819715/>

Яндекс Практикум. (2024, 3 июля). Android-разработчики в 2024 году: какие нужны скилы и как проходит процесс найма. *Habr*. <https://habr.com/ru/companies/yandex_praktikum/articles/824672/>

ПРИЛОЖЕНИЯ

## Приложение 1. Полный список стран

* *Austria*
* *Belgium*
* *Bulgaria*
* *Canada*
* *Croatia*
* *Cyprus*
* *Czechia*
* *Denmark*
* *Estonia*
* *Finland*
* *France*
* *Germany*
* *Greece*
* *Hungary*
* *Iceland*
* *Ireland*
* *Italy*
* *Latvia*
* *Liechtenstein*
* *Lithuania*
* *Luxembourg*
* *Malta*
* *Mexico*
* *Netherlands*
* *Norway*
* *Poland*
* *Portugal*
* *Romania*
* *Slovakia*
* *Slovenia*
* *Spain*
* *Sweden*
* *Switzerland*
* *United Kingdom*
* *United States*

## Приложение 2. Директории с вакансиями

Директории, в которые собирались вакансии:

* *«data/jobs\_data/data/dotcom\_domain/Android»* — вакансии для *Android*-разработчиков, найденные через *google.com*;
* *«data/jobs\_data/data/dotcom\_domain/iOS»* — вакансии для *iOS*-разработчиков, найденные через google.com;
* *«data/jobs\_data/data/local\_domain/Android»* — вакансии для *Android*-разработчиков, найденные через локальные домены;
* *«data/jobs\_data/data/data/local\_domain/iOS»* — вакансии для *iOS*-разработчиков, найденные через локальные домены.

## Приложение 3. Единый набор данных

Итоговый набор данных включал следующие столбцы:

* ***Location*** — страна, в которой проводился поиск вакансий;
* ***Region*** — регион страны поиска (*Europe* или *Northern* *America*);
* ***EU* *Member*** — флаг, указывающий на членство страны в Европейском Союзе (*True/False*);
* ***Schengen* *Agreement*** — флаг, указывающий на участие страны в Шенгенском соглашении (*True/False*);
* ***Google* *Domain* *Type*** — тип домена, используемого для поиска *(«default»/ «local»*);
* ***Google Domain Used*** — конкретный домен *Google*, использованный при поиске (например, *google.com*);
* ***Job* *Title*** — название вакансии (например, *«Senior Android Developer»*);
* ***Company Name*** — название компании, разместившей вакансию;
* ***Job Location*** — местоположение вакансии (например, *«Vienna, Austria»* или *«Anywhere»* для удалённой работы);
* ***Apply Options*** — список сайтов, где можно подать заявку на вакансию (например, *Indeed, LinkedIn, Startup Jobs* и т. д.);
* ***Job Description*** — описание вакансии, включая требования и обязанности;
* ***Work from home*** — возможность удалённой работы (*True/False* или *NaN*, если не указано);
* ***Salary*** — указанная в вакансии заработная плата (если предоставлена);
* ***Schedule type*** — тип занятости (*«Full-time»*, *«Part-time»* и т. д.);
* ***Qualifications*** — требования к кандидату (например, *«No degree mentioned»*, если степень не указана, *NaN*, если указано);
* ***Job ID*** — уникальный идентификатор вакансии;
* ***Search Date*** — дата и время выполнения поиска (например, «2025-01-13 12:08:48 *UTC*»);
* ***Search Query*** — поисковый запрос, использованный для поиска вакансий (*«Android developer»* или *«iOS developer»*).

## Приложение 4. Промпт для разбиения вакансий на логические части

***System Prompt:*** *"You are an AI assistant. Your role is to extract specific information from job descriptions and format them in a strict structure. "*

***User Prompt:*** *""" I will provide a job description. Please extract and present the information in \*\*this exact order\*\*:*

1. *Platform: (Android/iOS/Cross-platform)*
2. *Salary: (If stated; otherwise "Not mentioned")*
3. *Requirements: (verbatim from the job description or "Not mentioned")*
4. *Nice to have: (verbatim or "Not mentioned")*
5. *Responsibilities: (verbatim or "Not mentioned")*
6. *Benefits: (verbatim or "Not mentioned")*

*\*\*Guidelines\*\*:*

*- \*\*DO NOT reword, paraphrase, or summarize\*\* any part of the job description. Copy the sentences exactly as they appear.*

*- Combine all mandatory or required skill sections (e.g., "Requirements," "Skills," "Key Technologies," "About You") under \*\*Requirements\*\*.*

*- If the job description specifically says something is "a plus," "beneficial," or otherwise indicates it’s optional, place it under "Nice to have" even if it appears under a "Requirements" heading in the job description.*

*- If there is an "About you" or "About Role" section (or similar) that describs duties or tasks, include those under "Responsibilities".*

*- If the information is not in the job description, write "Not mentioned" for that section.*

*- For multiple platforms (e.g., Android, iOS), list them all in \*\*Platform\*\* and use headings under Requirements (and other sections, if needed) like "General Requirements:", "For Android Developers:", "For iOS Developers:".*

*- Present your answer \*\*only\*\* in the format above.*

*---*

*Here is the job description:*

*"""*

## Приложение 5. Пример вакансии после преобразования

*1. Platform: Android*

*2. Salary: Not mentioned*

*3. Requirements:*

*• Passionate about mobile platforms and translating requirements into a user friendly app*

*• Excellent knowledge of Kotlin*

*• Strong knowledge of architecture patterns such as MVVM*

*• Proven experience of developing and publishing native Android apps to the Google Play store in a professional environment*

*• Good knowledge of Kotlin Coroutines, Hilt and Retrofit.*

*4. Nice to have: Not mentioned*

*5. Responsibilities:*

*• Ensure that the app meets our quality standards by creating unit tests and doing code reviews*

*• Design and maintain the architecture of new features and existing ones*

*• Collaborate with cross-functional teams, designers and technical product owners*

*• Stay up to date with programming best practices and new technology trends*

*• Working autonomously you will bring strong communication skills and the ability to mentor colleagues*

*6. Benefits: Not mentioned*

## Приложение 6. Промпт для извлечения технологий из вакансий

*system\_prompt\_1 = """*

*You are a senior extraction assistant.*

*GOAL*

*Given a natural-language job-vacancy text, return one and only one valid, minified JSON object that maps every technology term found in the input to exactly ONE key from ALLOWED\_KEYS.*

*STRICT RULES*

*1. Output = raw JSON only (no Markdown, no comments, no extra text).*

*2. Use keys from ALLOWED\_KEYS exactly as written. Omit any key whose array would be empty.*

*3. Each value is an \*\*array of unique strings\*\*, sorted alphabetically and preserving the term’s original spelling/case from the input.*

*4. Ignore soft skills and vague nouns (e.g. communication, documentation, performance, detail, English, collaboration, problem-solving, university).*

*5. \*\*IGNORE\*\* overly generic technology words/phrases (e.g. "design patterns", "android ui", "json", "xml", "clean code", "API integration", "unit testing", "Continuous integration", "CI", "CD", "CI/CD", "cryptography", "application testing" ect.).*

*6. DO NOT hallucinate. Include a term \*\*only\*\* if it appears verbatim in the input text.*

*7. If a technology term matches \*\*exactly\*\* (case-insensitive) one of the values in ALLOWED\_KEYS, then it MUST be assigned to that exact key — even if the name could hypothetically fit other categories.*

*8. The order of keys in the output JSON MUST strictly follow the order of keys as they appear in the ALLOWED\_KEYS section below. Only include keys that have at least one matched term. If a key has no matched terms, it MUST be completely omitted from the output, even if this breaks the visual continuity of the ALLOWED\_KEYS order.*

*ALLOWED\_KEYS (with examples)*

*{*

*"languages\_and\_runtimes": ["Swift", "Kotlin", "Java", "Dart", "Objective-C", "Coroutines", "RxSwift", "Combine", "GCD", "RxJava", "Flows", "JavaScript", "TypeScript", "C", "C++", "Python", "Golang", "PHP"],*

*"ui\_and\_cross\_platform\_frameworks": ["SwiftUI", "UIKit", "Jetpack Compose", "Flutter", "React Native", "Xamarin", "Ionic", "WatchKit", "Cocoa Touch", "PhoneGap", "Cordova", "Kotlin Multiplatform"],*

*"architectural\_patterns": ["MVVM", "VIPER", "Clean Architecture", "MVI", "MVC", "MVP", "Redux"],*

*"dependency\_injection\_frameworks": ["Dagger", "Hilt", "Koin", "Swinject"],*

*"build\_and\_dependency\_management": ["Gradle", "CocoaPods", "SPM", "Bazel", "Buck", "Xcode", "Android Studio", "CircleCI", "Bamboo", "CocoaPods"],*

*"data\_and\_caching": ["Core Data", "Room", "Realm", "SQLite", "Firestore", "MongoDB", "SAP UltraLite", "MySQL", "NoSQL"],*

*"networking\_and\_api": ["Retrofit", "OkHttp", "URLSession", "Alamofire", "GraphQL", "REST API", "WebSockets", "gRPC", "Protocol Buffers"],*

*"backend\_or\_baas\_integration": ["Firebase", "AWS Amplify", "Azure Mobile", "Parse", "AWS Mobile Hub", "AWS Cognito", "AWS S3"],*

*"device\_and\_platform\_services\_and\_third\_party\_sdks": ["ARKit", "HealthKit", "CoreML", "Core Animation", "Android SDK", "Android NDK", "Push Notifications", "BLE", "NFC", "Camera", "Location", "Sensors", "Stripe SDK", "Facebook SDK", "AdMob", "Google Maps", "ExoPlayer", "Glide", "Stripe", "PayPal SDK", "Binder", "AIDL", "JNI", "CTS", "HAL", "LoRa", "CarPlay", "Android Auto", "CydiaSubstrate", "Frida", "WebViews"],*

*"security\_and\_cryptography": ["Keychain", "TLS pinning", "OAuth2", "Veracode", "Checkmarx", "OWASP"],*

*"testing\_frameworks": ["XCTest", "JUnit", "Espresso", "Mockito", "Robolectric"],*

*"debugging\_or\_profiling": ["Instruments", "Android Profiler"],*

*"version\_control": ["Git", "GIT", "git", "SVN", "Mercurial", "Gerrit", "GitFlow", "SourceTree", "Fork", "Bitbucket"],*

*"ci\_cd\_and\_release\_automation": ["GitHub Actions", "Jenkins", "Bitrise", "fastlane", "CircleCI", "Bamboo", "GitLab CI", "Docker"],*

*"monitoring\_analytics\_and\_crash\_reporting": ["Crashlytics", "Sentry", "Datadog", "Firebase Analytics", "App Center"],*

*"development\_methodologies": ["Scrum", "Kanban", "Agile", "SAFe", "TDD", "BDD", "DevOps"],*

*"testing\_process\_and\_qa": ["test coverage", "regression testing"],*

*"code\_quality\_and\_static\_analysis": ["SonarQube", "SwiftLint", "Veracode", "Checkmarx"],*

*"documentation\_and\_knowledge\_sharing": ["Swagger", "OpenAPI", "Javadoc", "Confluence", "HIG"],*

*"collaboration\_pm\_and\_design\_handoff": ["Jira", "Trello", "Figma", "Zeplin", "Rally/AgileCentral"],*

*"distribution\_and\_store\_operations": ["TestFlight", "App Store Connect", "Google Play Console"],*

*"compliance\_and\_certifications": ["SOC 2", "GDPR"]*

*}*

*EXAMPLE RESPONSE*

*{*

*"languages\_and\_runtimes":["Swift","Java"],*

*"version\_control":["Git"]*

*}*

*"""*

*user\_prompt\_1 = """*

*Extract every technology, tool, framework, library, service or formal methodology mentioned in the text below and output the JSON exactly as specified in the system prompt.*

*---*

*INPUT TEXT:*

*"""*

## Приложение 7. Результаты по категориям

**Архитектурные шаблоны программного обеспечения**

Категория *«architecture»* охватывает архитектурные подходы к структурированию клиентских приложений, включая принципы разделения ответственности, управление состоянием и потоком данных. В таблицах 3–4 представлены результаты по платформам *Android* и *iOS*.

**Таблица 3. Архитектурные шаблоны программного обеспечения для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Model-View-ViewModel (MVVM) | 157 | 22,82 | Model-View-ViewModel (MVVM) | 74 | 25,61 |
| Model-View-Presenter (MVP) | 68 | 9,88 | Model-View-Presenter (MVP) | 29 | 10,03 |
| Clean Architecture | 53 | 7,7 | Clean Architecture | 24 | 8,3 |
| Model-View-Intent (MVI) | 26 | 3,78 | Model-View-Controller (MVC) | 20 | 6,92 |
| Model-View-Controller (MVC) | 25 | 3,63 | Model-View-Intent (MVI) | 15 | 5,19 |
| Redux | 2 | 0,29 | View-Interactor-Presenter-Entity-Router (VIPER) | 5 | 1,73 |
| View-Interactor-Presenter-Entity-Router (VIPER) | 2 | 0,29 | Redux | 1 | 0,35 |

**Таблица 4. Архитектурные шаблоны программного обеспечения для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Model-View-ViewModel (MVVM) | 158 | 21,27 | Model-View-ViewModel (MVVM) | 63 | 23,25 |
| Model-View-Controller (MVC) | 70 | 9,42 | Model-View-Controller (MVC) | 43 | 15,87 |
|  | | | | | |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| View-Interactor-Presenter-Entity-Router (VIPER) | 47 | 6,33 | Model-View-Presenter (MVP) | 16 | 5,90 |
| Model-View-Presenter (MVP) | 38 | 5,11 | View-Interactor-Presenter-Entity-Router (VIPER) | 15 | 5,54 |
| Clean Architecture | 24 | 3,23 | Clean Architecture | 9 | 3,32 |
| Redux | 5 | 0,67 | Redux | 2 | 0,74 |
| View-Interactor-Presenter (VIP) | 5 | 0,67 | Model-View-Intent (MVI) | 1 | 0,37 |
| Model-View-Intent (MVI) | 2 | 0,27 |  |  |  |

Для ***Android*** лидирует *MVVM*; далее — *MVP* и *Clean* *Architecture*; *MVI* и *MVC* — реже; *VIPER* и *Redux* — точечно. Для ***iOS*** лидирует *MVVM*; далее — *MVC*; *VIPER* и *MVP* — реже; *Clean* *Architecture* — реже; *VIP* и *MVI* — точечно. **Сопоставляя регионы**, порядок лидеров внутри каждой платформы сохраняется между Европой и Северной Америкой: *MVVM* занимает первое место во всех четырёх группах.

**Инструменты командного взаимодействия и управления разработкой**

Категория *«team\_collaboration»* включает программные средства для координации командной работы: учёт задач и инцидентов, корпоративные базы знаний и обмен проектными материалами между дизайном и разработкой. В таблицах 5–6 представлены результаты по *Android* и *iOS*.

**Таблица 5. Инструменты командного взаимодействия и управления разработкой для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Jira | 57 | 8,28 | Jira | 22 | 7,61 |
| Confluence | 32 | 4,65 | Confluence | 2 | 0,69 |
| Figma | 7 | 1,02 | Figma | 1 | 0,35 |
| Trello | 1 | 0,15 |  |  |  |

**Таблица 6. Инструменты командного взаимодействия и управления разработкой для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Jira | 37 | 4,98 | Jira | 16 | 5,90 |
| Confluence | 20 | 2,69 | Confluence | 8 | 2,95 |
| Figma | 9 | 1,21 | Figma | 7 | 2,58 |
| Zeplin | 5 | 0,67 | Zeplin | 1 | 0,37 |
| Trello | 1 | 0,13 |  |  |  |

Для ***Android*** лидирует *Jira*; далее — *Confluence*; *Figma* — эпизодически; *Trello* — точечно. Для ***iOS*** лидирует *Jira*; далее — *Confluence*; *Figma* упоминается чаще, чем в *Android*; *Zeplin* — эпизодически; *Trello* — точечно. **Сопоставляя регионы**, профиль схож на обеих платформах: доминирует связка *Jira–Confluence*; инструменты дизайнерского обмена *(Figma/Zeplin*) встречаются относительно чаще в *iOS*.

**Интегрированные среды разработки (*IDE*)**

Категория *«ides»* охватывает интегрированные среды разработки (*IDE*), объединяющие инструменты редактирования кода, сборки, отладки, анализа и эмуляции для мобильных приложений. В таблицах 7–8 представлены результаты по платформам *Android* и *iOS*.

**Таблица 7. Интегрированные среды разработки (*IDE*) для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Android Studio | 124 | 18,02 | Android Studio | 63 | 21,80 |
| Eclipse | 2 | 0,29 | Eclipse | 4 | 1,38 |

**Таблица 8. Интегрированные среды разработки (*IDE*) для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Xcode | 142 | 19,11 | Xcode | 71 | 26,20 |

Для ***Android*** лидирует *Android* *Studio*; *Eclipse* — эпизодически. Для ***iOS*** фиксируется только *Xcode*. **Сопоставляя** **регионы**, лидеры одинаковы в Европе и Северной Америке: для *Android* — *Android* *Studio*; для *iOS* — *Xcode*; вариативность вне нативных решений минимальна.

**Интероперабельность и взаимодействие с нативным кодом**

Категория *«native\_interop»* охватывает средства взаимодействия мобильных приложений с нативным кодом и системными сервисами: подключение модулей на *C/C++*, межъязыковые вызовы (*Kotlin/Java ↔ C/C++*), а также механизмы межпроцессного обмена данными. В таблице 9 представлены результаты по платформе *Android*.

**Таблица 9. Интероперабельность и взаимодействие с нативным кодом для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Android NDK | 44 | 6,40 | Android NDK | 12 | 4,15 |
| Java Native Interface (JNI) | 18 | 2,62 | Java Native Interface (JNI) | 7 | 2,42 |
| Android Interface Definition Language (AIDL) | 13 | 1,89 | Android Interface Definition Language (AIDL) | 4 | 1,38 |
| Binder | 8 | 1,16 | Binder | 1 | 0,35 |

Для ***Android*** лидирует *Android* *NDK*; далее следует *JNI*; реже встречаются *AIDL* (описание интерфейсов для *IPC*) и *Binder* (механизм *IPC*). Для ***iOS*** в рассматриваемой выборке явных упоминаний по категории не зафиксировано. **Сопоставляя** **регионы**, порядок лидеров внутри *Android* сохраняется между Европой и Северной Америкой: *Android* *NDK* → *JNI* → *AIDL* → *Binder*.

**Информационная безопасность и нормативное соответствие**

Категория *«security\_compliance»* включает требования и практики, связанные с защитой данных и соблюдением нормативов в мобильной разработке: механизмы аутентификации и авторизации, безопасное хранение секретов, криптографические методы и ориентиры безопасного кодирования. В таблицах 10–11 представлены результаты по платформам *Android* и *iOS*.

**Таблица 10. Информационная безопасность и нормативное соответствие для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| General Data Protection Regulation (GDPR) | 1 | 0,15 | OAuth 2.0 | 3 | 1,04 |
| OAuth | 1 | 0,15 | General Data Protection Regulation (GDPR) | 1 | 0,35 |
| OAuth 2.0 | 1 | 0,15 | OAuth | 1 | 0,35 |
| Open Web Application Security Project (OWASP) | 1 | 0,15 |  |  |  |

**Таблица 11. Информационная безопасность и нормативное соответствие для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Keychain | 5 | 0,67 | Keychain | 6 | 2,21 |
| OAuth | 3 | 0,40 | OAuth | 4 | 1,48 |
| OAuth 2.0 | 3 | 0,40 | Advanced Encryption Standard (AES) | 1 | 0,37 |
| Open Web Application Security Project (OWASP) | 2 | 0,27 | Rivest–Shamir–Adleman (RSA) | 1 | 0,37 |
| Advanced Encryption Standard (AES) | 1 | 0,13 | TLS pinning | 1 | 0,37 |
| Rivest–Shamir–Adleman (RSA) | 1 | 0,13 |  |  |  |
| TLS pinning | 1 | 0,13 |  |  |  |

Для ***Android*** зафиксированы единичные упоминания регламента *GDPR*, протоколов *OAuth/OAuth 2.0* и рекомендаций *OWASP*; общий объём невысокий. Для ***iOS*** стабильно лидирует *Keychain* (платформенное безопасное хранилище); далее отмечены *OAuth/OAuth 2.0*; реже встречаются *AES*, *RSA* и *TLS* *pinning*; *OWASP* — точечно. **Сопоставляя** **регионы**, профиль разрежен на обеих платформах и концентрируется вокруг базовых практик: *iOS* чаще фиксирует платформенное хранение секретов (*Keychain*), тогда как в *Android* единично присутствуют регуляторные и методические ориентиры (*GDPR/OWASP*); *OAuth* упоминается в обоих регионах.

**Исполняемые среды и мультиплатформенные рантаймы**

Категория *«runtimes\_shared»* охватывает среды выполнения и технологические решения для совместного использования кода между платформами и унификации исполняемого окружения. В таблицах 12–13 представлены результаты по платформам *Android* и *iOS*.

**Таблица 12. Исполняемые среды и мультиплатформенные рантаймы для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Kotlin Multiplatform | 29 | 4,22 | Kotlin Multiplatform | 11 | 3,81 |
| Node.js | 4 | 0,58 | Node.js | 4 | 1,38 |

**Таблица 13. Исполняемые среды и мультиплатформенные рантаймы для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Kotlin Multiplatform | 13 | 1,75 | .NET Core | 2 | 0,74 |
| Node.js | 2 | 0,27 | Kotlin Multiplatform | 1 | 0,37 |

Для ***Android*** лидирует *Kotlin* *Multiplatform*; *Node.js* — реже. Для ***iOS*** профиль разрежен: в Европе — *Kotlin* *Multiplatform* и точечно *Node.js*; в Северной Америке — точечно *.NET Core*. **Сопоставляя** **регионы**, наблюдается узкий набор практик с концентрацией вокруг *Kotlin* *Multiplatform*; *.NET Core* появляется только в североамериканской *iOS*-подвыборке, тогда как *Node.js* встречается в обеих регионах на вторых ролях.

**Кроссплатформенные *SDK* для устройств и сервисов**

Категория *«cross\_platform\_sdks»* охватывает *SDK*, которые предоставляют единый доступ к возможностям устройства и внешним сервисам на обеих платформах: системные уведомления, *Bluetooth/BLE*, геолокацию, камеру и датчики, *NFC*, а также интеграции со сторонними сервисами (карты, соцсети, платежи). В таблицах 13–14 представлены результаты по платформам *Android* и *iOS*.

**Таблица 13. Кроссплатформенные *SDK* для устройств и сервисов для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Notifications | 18 | 2,62 | Notifications | 15 | 5,19 |
| Bluetooth Low Energy (BLE) | 9 | 1,31 | Bluetooth | 11 | 3,81 |
| Camera | 7 | 1,02 | Location | 7 | 2,42 |
| Location | 5 | 0,73 | Bluetooth Low Energy (BLE) | 5 | 1,73 |
| Sensors | 5 | 0,73 | Camera | 3 | 1,04 |
| Bluetooth | 4 | 0,58 | Facebook SDK | 3 | 1,04 |
| Facebook SDK | 1 | 0,15 | Google Maps | 3 | 1,04 |
| Google Cloud Messaging | 1 | 0,15 | NFC | 2 | 0,69 |
| NFC | 1 | 0,15 | Sensors | 2 | 0,69 |
|  |  |  | Stripe SDK | 1 | 0,35 |

**Таблица 14. Кроссплатформенные *SDK* для устройств и сервисов для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Notifications | 32 | 4,31 | Notifications | 26 | 9,59 |
| Bluetooth | 10 | 1,35 | Facebook SDK | 11 | 4,06 |
| Bluetooth Low Energy (BLE) | 10 | 1,35 | Location | 10 | 3,69 |
| Location | 4 | 0,54 | Bluetooth | 7 | 2,58 |
| Camera | 3 | 0,40 | Camera | 2 | 0,74 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Sensors | 2 | 0,27 | Google Cloud Messaging | 2 | 0,74 |
| Stripe SDK | 2 | 0,27 | Google Maps | 2 | 0,74 |
| Facebook SDK | 1 | 0,13 | Stripe SDK | 2 | 0,74 |
| Google Maps | 1 | 0,13 | NFC | 1 | 0,37 |
| NFC | 1 | 0,13 | PayPal SDK | 1 | 0,37 |
| PayPal SDK | 1 | 0,13 | Sensors | 1 | 0,37 |

Для ***Android*** устойчиво лидируют *Notifications*; далее следуют *Bluetooth/BLE* и *Location*. *Camera* и *Sensors* встречаются реже; *NFC* и сторонние *SDK* (*Facebook SDK*, *Google* *Maps*, *Stripe*) отмечены точечно. Для ***iOS*** аналогично: на первом месте — *Notifications*; далее *Bluetooth/BLE* и *Location*. *Camera* и *Sensors* упоминаются реже; среди сторонних решений фиксируются *Facebook* *SDK*, *Google* *Maps* и платежные *SDK* (*Stripe/PayPal*) — эпизодически. **Сопоставляя** **регионы**, доля *Notifications* выше в Северной Америке для обеих платформ; для *iOS* в североамериканской подвыборке чаще встречается *Facebook* *SDK*, тогда как прочие позиции (*Bluetooth/BLE, Location*) остаются близкими по структуре.

**Кроссплатформенные *UI*-фреймворки**

Категория *«cross\_platform\_ui»* включает фреймворки для разработки пользовательских интерфейсов с единой кодовой базой под несколько платформ. В таблицах 15–16 представлены результаты по платформам *Android* и *iOS*.

**Таблица 15. Кроссплатформенные *UI*-фреймворки для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Flutter | 22 | 3,20 | React Native | 20 | 6,92 |
| React Native | 15 | 2,18 | Flutter | 15 | 5,19 |
| Apache Cordova | 4 | 0,58 | Ionic | 5 | 1,73 |
| Xamarin | 3 | 0,44 | Apache Cordova | 3 | 1,04 |
| Ionic | 2 | 0,29 | Xamarin | 2 | 0,69 |

**Таблица 16. Кроссплатформенные *UI*-фреймворки для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| React Native | 24 | 3,23 | React Native | 14 | 5,17 |
| Flutter | 22 | 2,96 | Flutter | 9 | 3,32 |
| Xamarin | 9 | 1,21 | Ionic | 7 | 2,58 |
| Apache Cordova | 5 | 0,67 | Apache Cordova | 6 | 2,21 |
| Ionic | 3 | 0,40 | Xamarin | 4 | 1,48 |

Для ***Android*** доминирует пара *Flutter* и *React* *Native* с перестановкой лидера между регионами; *Apache* *Cordova*, *Xamarin* и *Ionic* фиксируются заметно реже. Для ***iOS*** первой остаётся *React* *Native*, далее следует *Flutter*; *Xamarin*, *Ionic* и *Apache* *Cordova* представлены ограниченно. **Сопоставляя** **регионы**, структура в целом концентрируется вокруг пары *React* *Native* и *Flutter*; в *Android* наблюдается смена лидера между Европой и Северной Америкой, тогда как в *iOS* сохраняется стабильное первенство *React* *Native*.

## 4.9. Методологии и практики управления разработкой

Категория *«methodologies»* включает методологические подходы и организационные практики: гибкие методы (*Agile*) и их фреймворки (*Scrum*, *Kanban*), инженерные практики качества (*TDD/BDD*), масштабируемые модели (*SAFe*), а также каскадные модели (*Waterfall*). В таблицах 16–17 представлены результаты по платформам *Android* и *iOS*.

**Таблица 16. Методологии и практики управления разработкой для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Agile | 158 | 22,97 | Agile | 82 | 28,37 |
| Scrum | 81 | 11,77 | Scrum | 35 | 12,11 |
| Test-Driven Development (TDD) | 52 | 7,56 | Test-Driven Development (TDD) | 15 | 5,19 |
| Kanban | 13 | 1,89 | Kanban | 5 | 1,73 |
| Behavior-Driven Development (BDD) | 6 | 0,87 | Behavior-Driven Development (BDD) | 2 | 0,69 |
| Scaled Agile Framework (SAFe) | 2 | 0,29 | Waterfall | 2 | 0,69 |
|  |  |  | Scaled Agile Framework (SAFe) | 1 | 0,35 |

**Таблица 17. Методологии и практики управления разработкой для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Agile | 146 | 19,65 | Agile | 90 | 33,21 |
| Scrum | 69 | 9,29 | Scrum | 34 | 12,55 |
| Test-Driven Development (TDD) | 46 | 6,19 | Test-Driven Development (TDD) | 13 | 4,80 |
| Kanban | 24 | 3,23 | Kanban | 10 | 3,69 |
| Behavior-Driven Development (BDD) | 4 | 0,54 | Behavior-Driven Development (BDD) | 2 | 0,74 |
| Scaled Agile Framework (SAFe) | 2 | 0,27 | Waterfall | 2 | 0,74 |

Для ***Android*** лидирует *Agile*; далее следуют *Scrum* и *TDD*. *Kanban* фиксируется реже; *BDD* и *SAFe* — точечно; в Северной Америке дополнительно отмечены единичные упоминания *Waterfall*. Для ***iOS*** сохраняется та же иерархия (*Agile* → *Scrum* → *TDD*); *Kanban* представлен умеренно; *BDD* и *SAFe* — эпизодически; *Waterfall* встречается точечно в Северной Америке. **Сопоставляя** **регионы**, во всех четырёх группах лидирует *Agile*; различия касаются «длинного хвоста»: *Waterfall* фиксируется только в Северной Америке, тогда как *SAFe* чаще присутствует в европейских подвыборках.

## 4.10. Мониторинг и аналитика

Категория *«monitoring\_analytics»* охватывает средства наблюдаемости и аналитики клиентских приложений: краш-репортинг (*Crashlytics*, *App* *Center*), отслеживание ошибок и метрик/логов (*Sentry*, *Datadog*), базовую веб/моб. аналитику (*Google* *Analytics*) и продуктовую аналитику (*Amplitude*, *Mixpanel*). В таблицах 18–19 представлены результаты по платформам *Android* и *iOS*.

**Таблица 18. Мониторинг и аналитика для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Crashlytics | 10 | 1,45 | Crashlytics | 3 | 1,04 |
| Datadog | 3 | 0,44 | Google Analytics | 3 | 1,04 |
| Sentry | 2 | 0,29 | App Center | 1 | 0,35 |
| Amplitude | 1 | 0,15 |  |  |  |
| App Center | 1 | 0,15 |  |  |  |

**Таблица 19. Мониторинг и аналитика для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Crashlytics | 12 | 1,62 | Amplitude | 1 | 0,37 |
| Google Analytics | 4 | 0,54 | Crashlytics | 1 | 0,37 |
| Sentry | 3 | 0,40 |  |  |  |
| Mixpanel | 2 | 0,27 |  |  |  |
| Amplitude | 1 | 0,13 |  |  |  |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| App Center | 1 | 0,13 |  |  |  |
| Datadog | 1 | 0,13 |  |  |  |

Для ***Android*** лидирует *Crashlytics*; прочие решения (*Datadog*, *Sentry*, *Amplitude*, *Google* *Analytics*, *App* *Center*) — эпизодически. Для ***iOS*** лидирует *Crashlytics*; далее — *Google* *Analytics* и *Sentry*; *Mixpanel/Amplitude* — реже; *App Center* и *Datadog* — точечно. **Сопоставляя** **регионы**, на обеих платформах фокус смещён к краш-репортингу; при этом в Северной Америке доли отдельных инструментов сопоставимы (*Android*: *Crashlytics* и *Google* *Analytics*; *iOS*: *Amplitude* и *Crashlytics*), тогда как в Европе доминирование *Crashlytics* выражено сильнее.

## 4.11. Нативные *UI*-фреймворки и компоненты

Категория *«native\_ui»* охватывает платформенные средства построения пользовательского интерфейса: фреймворки и базовые компоненты для разметки экранов, обработки взаимодействий, рендеринга и анимации. В таблицах 20–21 представлены результаты по платформам *Android* и *iOS*.

**Таблица 20. Нативные *UI*-фреймворки и компоненты для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Jetpack Compose | 198 | 28,78 | Jetpack Compose | 82 | 28,37 |
| Glide | 6 | 0,87 | Glide | 3 | 1,04 |
| WebView | 3 | 0,44 |  |  |  |

**Таблица 21. Нативные *UI*-фреймворки и компоненты для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| SwiftUI | 283 | 38,09 | SwiftUI | 90 | 33,21 |
| UIKit | 190 | 25,57 | UIKit | 47 | 17,34 |
| Core Animation | 67 | 9,02 | Core Animation | 30 | 11,07 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Cocoa Touch | 40 | 5,38 | Cocoa Touch | 21 | 7,75 |
| WatchKit | 29 | 3,90 | WatchKit | 1 | 0,37 |

Для ***Android*** лидирует *Jetpack* *Compose*; *Glide* и *WebView* — реже. Для ***iOS*** лидирует *SwiftUI*; далее — *UIKit*; *Core* *Animation* и *Cocoa* *Touch* — реже; *WatchKit* — точечно. **Сопоставляя** **регионы**, состав лидеров сохраняется: декларативные фреймворки (*Jetpack* *Compose*, *SwiftUI*) занимают первые позиции в обеих подвыборках; *UIKit* стабильно второй в *iOS*.

## 4.12. Непрерывная интеграция и доставка (*CI/CD*)

Категория *«ci\_cd»* охватывает средства автоматизации сборки, тестирования и поставки мобильных приложений, а также управление конвейерами (*pipeline*): универсальные серверы (*Jenkins*), специализированные мобильные инструменты (*fastlane*, *Bitrise*), механизмы в экосистемах *VCS* (*GitHub* *Actions*) и контейнеризацию (*Docker*). В таблицах 22–23 представлены результаты по платформам *Android* и *iOS*.

**Таблица 22. Непрерывная интеграция и доставка (*CI/CD*) для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Jenkins | 32 | 4,65 | Jenkins | 17 | 5,88 |
| Bitrise | 20 | 2,91 | CircleCI | 7 | 2,42 |
| Docker | 18 | 2,62 | fastlane | 7 | 2,42 |
| GitHub Actions | 15 | 2,18 | GitHub Actions | 6 | 2,08 |
| fastlane | 15 | 2,18 | Docker | 2 | 0,69 |
| Bamboo | 6 | 0,87 | Azure DevOps | 1 | 0,35 |
| CircleCI | 4 | 0,58 |  |  |  |
| Azure DevOps | 3 | 0,44 |  |  |  |
| TeamCity | 2 | 0,29 |  |  |  |

**Таблица 23. Непрерывная интеграция и доставка (*CI/CD*) для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| fastlane | 50 | 6,73 | Jenkins | 12 | 4,43 |
| Jenkins | 44 | 5,92 | fastlane | 10 | 3,69 |
| Bitrise | 15 | 2,02 | CircleCI | 4 | 1,48 |
| GitHub Actions | 10 | 1,35 | Docker | 4 | 1,48 |
| Docker | 5 | 0,67 | Azure DevOps | 3 | 1,11 |
| TeamCity | 5 | 0,67 | Bitrise | 3 | 1,11 |
| Bamboo | 4 | 0,54 | Bamboo | 1 | 0,37 |
| CircleCI | 4 | 0,54 |  |  |  |
| Unity Cloud Build | 2 | 0,27 |  |  |  |
| Xcode Cloud | 2 | 0,27 |  |  |  |
| Azure DevOps | 1 | 0,13 |  |  |  |

Для ***Android*** лидирует *Jenkins*; далее — *Bitrise*, *Docker*, *GitHub* *Actions* и *fastlane*; *CircleCI*, *Azure* *DevOps*, *Bamboo* и *TeamCity* — эпизодически. Для ***iOS*** в Европе лидирует *fastlane*, далее — *Jenkins*; в Северной Америке — *Jenkins* опережает *fastlane*; далее — *Bitrise*, *GitHub* *Actions* и *Docker*; *Xcode* *Cloud* и *Unity* *Cloud* *Build* — точечно. **Сопоставляя** **регионы**, профиль стабильный: доминируют универсальные серверы (*Jenkins*) при заметной роли *fastlane*; различия сводятся к перестановке лидеров в *iOS*, тогда как в *Android* порядок сохраняется. Платформенные облачные сервисы упоминаются редко и на общую картину не влияют.

## 4.13. Отладка, профилирование и реверс-инжиниринг

Категория *«debug\_profiling»* охватывает программные средства и практики для анализа исполнения мобильных приложений: отладку (диагностика и воспроизведение дефектов), профилирование (нагрузка *CPU*, память, ввод-вывод) и реверс-инжиниринг (исследование исполняемых артефактов и структуры сборки). В таблицах 24–25 представлены результаты по платформам *Android* и *iOS*.

**Таблица 24.** **Отладка, профилирование и реверс-инжиниринг для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Android Profiler | 2 | 0,29 | Android Debug Bridge (ADB) | 4 | 1,38 |
| Frida | 2 | 0,29 | GNU Debugger (GDB) | 3 | 1,04 |
| Ghidra | 1 | 0,15 | WinDbg | 3 | 1,04 |
|  |  |  | Frida | 2 | 0,69 |
|  |  |  | Ghidra | 2 | 0,69 |

**Таблица 25.** **Отладка, профилирование и реверс-инжиниринг для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Cydia Substrate | 1 | 0,13 | Cydia Substrate | 1 | 0,37 |
| Frida | 1 | 0,13 | Frida | 1 | 0,37 |

Для ***Android*** упоминания единичны и распределены между утилитами отладки (*ADB*, *GDB*, *WinDbg*) и инструментами анализа (*Frida*, *Ghidra*); *Android* *Profiler* фиксируется точечно. Для ***iOS*** зафиксированы *Cydia* *Substrate* и *Frida*; профиль носит эпизодический характер. **Сопоставляя** **регионы**, частота упоминаний на обеих платформах низкая; в *Android* спектр наименований шире, тогда как в *iOS* присутствуют только отдельные инструменты (*Cydia* *Substrate*, *Frida*).

## 4.14. Параллелизм и реактивное программирование

Категория *«concurrency\_reactive»* включает средства организации асинхронности и обработки событий: языковые абстракции (*Kotlin* *Coroutines*, *Swift* *Concurrency*), реактивные фреймворки (*RxJava/RxSwift*, *Apple* *Combine*) и системные механизмы диспетчеризации (*GCD*). В таблицах 26–27 представлены результаты по платформам *Android* и *iOS*.

**Таблица 26.** **Параллелизм и реактивное программирование для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Kotlin Coroutines | 95 | 13,81 | Kotlin Coroutines | 40 | 13,84 |
| RxJava | 60 | 8,72 | RxJava | 39 | 13,49 |
| Kotlin Flow | 36 | 5,23 | Kotlin Flow | 8 | 2,77 |
| RxKotlin | 3 | 0,44 |  |  |  |

**Таблица 27.** **Параллелизм и реактивное программирование для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Apple Combine | 63 | 8,48 | Apple Combine | 38 | 14,02 |
| RxSwift | 62 | 8,34 | RxSwift | 15 | 5,54 |
| Grand Central Dispatch (GCD) | 39 | 5,25 | Grand Central Dispatch (GCD) | 4 | 1,48 |
| Swift Concurrency | 2 | 0,27 |  |  |  |

Для ***Android*** лидирует стек на базе *Kotlin* *Coroutines*; далее следует *RxJava*; *Kotlin* *Flow* фиксируется как дополнительный инструмент потоковой обработки; *RxKotlin* — эпизодически. Для ***iOS*** верхние позиции занимают *Apple* *Combine* и *RxSwift*; *Grand* *Central* *Dispatch (GCD)* присутствует как системный механизм диспетчеризации задач; *Swift* *Concurrency* отмечается точечно. **Сопоставляя регионы**, порядок лидеров внутри платформ одинаков, а различия касаются долей: в Северной Америке относительно выше доля *Combine* и *RxJava*, тогда как в Европе выше *RxSwift* и *GCD*.

## 4.15. Публикация и распространение приложений

Категория *«app\_distribution»* описывает инструменты и каналы официальной публикации и распространения мобильных приложений, а также сервисы, связанные с управлением релизами и предварительным тестированием. В таблицах 28–29 представлены результаты по платформам *Android* и *iOS*.

**Таблица 28.** **Публикация и распространение приложений для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Google Play Store / Google Play Console | 66 | 9,59 | Google Play Store / Google Play Console | 21 | 7,27 |
|  |  |  | Google Play Billing | 2 | 0,69 |

**Таблица 29.** **Публикация и распространение приложений для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| App Store / App Store Connect | 71 | 9,56 | App Store / App Store Connect | 22 | 8,12 |
| TestFlight | 8 | 1,08 | TestFlight | 4 | 1,48 |

Для ***Android*** лидирует *Google Play Store / Google Play Console*; *Google Play Billing* — точечно. Для ***iOS*** лидирует *App Store / App Store Connect*; далее — *TestFlight*. **Сопоставляя** **регионы**, профиль стандартизирован: в обоих регионах центром публикации остаются официальные магазины платформ; вспомогательные сервисы упоминаются ограниченно (для *iOS* чаще виден *TestFlight*, для *Android* — единично *Play* *Billing*).

## 4.16. Разработка игр

Категория *«game\_dev»* включает технологии, используемые при создании мобильных игр: игровые движки и сопутствующие инструменты контент-продакшена. В таблице 30 представлены результаты по платформе *iOS*.

**Таблица 30.** **Разработка игр для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Unity Engine | 5 | 0,67 |  |  |  |

Для ***Android*** в предоставленном наборе данных явные упоминания игровых технологий отсутствуют. Для ***iOS*** зафиксированы единичные упоминания *Unity* *Engine* в европейской подвыборке; в североамериканской — не отмечено. **Сопоставляя** **регионы**, профиль категории разрежен: единичные случаи фиксируются только в Европе и только для *Unity*; иных игровых технологий не выявлено.

## 4.17. Руководства по дизайну интерфейсов

Категория *«design\_guidelines»* охватывает официальные руководства по проектированию пользовательских интерфейсов: композицию экранов, навигационные паттерны, визуальные и типографские принципы, а также требования доступности и согласованности поведения компонентов. В таблицах 31–32 представлены результаты по платформам *Android* и *iOS*.

**Таблица 31.** **Руководства по дизайну интерфейсов для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Material Design | 3 | 0,44 |  |  |  |

**Таблица 32.** **Руководства по дизайну интерфейсов для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Human Interface Guidelines | 5 | 2,83 | Human Interface Guidelines | 10 | 3,69 |

Для ***Android*** зафиксированы единичные явные упоминания *Material* *Design* — официальной системы рекомендаций *Google*. Для ***iOS*** устойчиво присутствуют ссылки на *Human* *Interface* *Guidelines* — официальные руководства *Apple*, отмеченные в обеих региональных подвыборках. **Сопоставляя** **регионы**, явные ссылки на руководства чаще встречаются в *iOS*-группах как в Европе, так и в Северной Америке; в *Android* упоминания точечны.

## 4.18. Сборка и управление зависимостями

Категория *«build\_dependency»* охватывает средства автоматизации сборки и управления зависимостями в мобильных проектах: настройку и выполнение компиляции/тестирования/упаковки артефактов и декларативное подключение внешних библиотек. В таблицах 33–34 представлены результаты по платформам *Android* и *iOS*.

**Таблица 33.** **Сборка и управление зависимостями для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Gradle | 65 | 9,45 | Gradle | 32 | 11,07 |
| Apache Maven | 5 | 0,73 | Apache Maven | 4 | 1,38 |
| Yocto Project | 1 | 0,15 | Makefile | 2 | 0,69 |
|  |  |  | Yocto Project | 1 | 0,35 |

**Таблица 34.** **Сборка и управление зависимостями для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| CocoaPods | 70 | 9,42 | CocoaPods | 14 | 5,17 |
| Swift Package Manager | 43 | 5,79 | Swift Package Manager | 7 | 2,58 |
| Carthage | 6 | 0,81 | Bazel | 4 | 1,48 |
| Gradle | 4 | 0,54 | Apache Maven | 2 | 0,74 |
| Apache Maven | 1 | 0,13 | npm | 2 | 0,74 |
| Bazel | 1 | 0,13 |  |  |  |

Для ***Android*** доминирует *Gradle* в обеих региональных подвыборках; *Apache* *Maven* заметно реже; *Yocto* *Project* и *Makefile* фиксируются точечно. Для ***iOS*** лидирует *CocoaPods*; на второй позиции — *Swift* *Package* *Manager*; *Carthage* встречается реже. Отмечены единичные упоминания *Bazel*, а также нетипичных для *iOS* инструментов (*Gradle*, *Apache* *Maven*, npm*).* **Сопоставляя** **регионы**, порядок лидеров сохраняется внутри каждой платформы (*Android* — *Gradle*; *iOS* — *CocoaPods* → *Swift* *Package* *Manager*); различия касаются долей и «длинного хвоста» редких инструментов (например, *Bazel* чаще отмечен в Северной Америке).

## 4.19. Серверные платформы и облачные сервисы

Категория *«backend\_baas»* охватывает управляемые серверные платформы и публичные облака, которые используют мобильные команды для хранения данных, аутентификации, серверной логики и интеграций. Включает решения класса *BaaS* (например, *Firebase*, *AWS* *Amplify*, *Parse* *Platform*) и универсальные облачные провайдеры (*AWS*, *Google Cloud Platform*, *Azure*). В таблицах 35–36 представлены результаты по платформам *Android* и *iOS*.

**Таблица 35.** **Серверные платформы и облачные сервисы для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Firebase | 45 | 6,54 | Firebase | 20 | 6,92 |
| Amazon Web Services (AWS) | 7 | 1,02 | Amazon Web Services (AWS) | 4 | 1,38 |
| Cloud Firestore | 2 | 0,29 | Google Cloud Platform | 2 | 0,69 |
| Microsoft Azure | 2 | 0,29 | Amazon S3 | 1 | 0,35 |
| Amazon S3 | 1 | 0,15 | Cloud Firestore | 1 | 0,35 |
| Spring Boot | 1 | 0,15 | Parse Platform | 1 | 0,35 |
|  |  |  | Spring Boot | 1 | 0,35 |

**Таблица 36.** **Серверные платформы и облачные сервисы для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Firebase | 40 | 5,38 | Amazon Web Services (AWS) | 5 | 1,85 |
| Amazon Web Services (AWS) | 14 | 1,88 | Firebase | 5 | 1,85 |
| Cloud Firestore | 2 | 0,27 | AWS Lambda | 2 | 0,74 |
| Microsoft Azure | 2 | 0,27 | Microsoft Azure | 2 | 0,74 |
| AWS Amplify | 1 | 0,13 | AWS Amplify | 1 | 0,37 |
| Amazon S3 | 1 | 0,13 | Parse Platform | 1 | 0,37 |
| Google Cloud Platform | 1 | 0,13 |  |  |  |

Для ***Android*** лидирует *Firebase*; далее отмечаются *AWS* и точечные упоминания отдельных сервисов и стеков (*Cloud* *Firestore*, *Azure*, *S3*; локально — *Spring* *Boot/Parse*). Для ***iOS*** доминируют *Firebase* и *AWS*; дополнительно встречаются сервисы уровня компонентов (*AWS* *Lambda*, *S3*) и редкие альтернативы (*Amplify*, *Parse*, *GCP*, *Azure*). **Сопоставляя** **регионы**, картина единообразная: на обеих платформах устойчивое ядро формируют *Firebase* и *AWS*; отличия ограничиваются редкими упоминаниями отдельных облачных сервисов и не меняют общий профиль.

## 4.20. Сетевые протоколы и клиентские библиотеки

Категория *«networking»* включает способы сетевого взаимодействия мобильных приложений с удалёнными сервисами: модели прикладных интерфейсов (*REST*, *GraphQL*), обмен в реальном времени (*WebSockets*, *WebRTC*), удалённые вызовы (*gRPC*) и форматы сериализации (*Protocol* *Buffers*), а также клиентские *HTTP*-библиотеки (на *Android* — *Retrofit*, *OkHttp*; на *iOS* — *URLSession*, *Alamofire*, *Moya*). В таблицах 37–38 представлены результаты по платформам *Android* и *iOS*.

**Таблица 37.** **Сетевые протоколы и клиентские библиотеки для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| REST API | 161 | 23,4 | REST API | 98 | 33,91 |
| Retrofit | 49 | 7,12 | Retrofit | 37 | 12,80 |
| GraphQL | 17 | 2,47 | GraphQL | 21 | 7,27 |
| OkHttp | 10 | 1,45 | OkHttp | 12 | 4,15 |
| WebSockets | 3 | 0,44 | WebSockets | 7 | 2,42 |
| gRPC | 3 | 0,44 | Protocol Buffers | 3 | 1,04 |
| Protocol Buffers | 2 | 0,29 | gRPC | 1 | 0,35 |

**Таблица 38.** **Сетевые протоколы и клиентские библиотеки для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| REST API | 185 | 24,9 | REST API | 74 | 27,31 |
| GraphQL | 26 | 3,5 | GraphQL | 18 | 6,64 |
| Alamofire | 15 | 2,02 | Alamofire | 11 | 4,06 |
| WebSockets | 14 | 1,88 | WebRTC | 2 | 0,74 |
| URLSession | 6 | 0,81 | WebSockets | 2 | 0,74 |
| Moya | 2 | 0,27 | Moya | 1 | 0,37 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Protocol Buffers | 2 | 0,27 | Retrofit | 1 | 0,37 |
| gRPC | 2 | 0,27 | URLSession | 1 | 0,37 |

Для ***Android*** лидирует *REST*; основная связка — *Retrofit* + *OkHttp*; *GraphQL* заметно реже; *WebSockets* эпизодически; *gRPC* и *Protocol* *Buffers* — точечно. Для ***iOS*** также лидирует *REST*; фиксируются *URLSession*, а также *Alamofire* и *Moya*; присутствуют *GraphQL* и *WebSockets*; *WebRTC* — точечно; *gRPC/Protocol* *Buffers* — нишево. **Сопоставляя** **регионы**, профиль стабилен: доминирует *REST*, клиентские библиотеки одинаковы; в Северной Америке относительно выше доля *GraphQL* и каналов реального времени (*WebSockets/WebRTC*).

## 4.21. Системы контроля версий и платформы

Категория *«version\_control»* охватывает системы контроля версий (распределённые — *Git*, *Mercurial*; централизованные — *Apache* *Subversion*, *SVN*), а также платформы хостинга и совместной работы с репозиториями (*GitHub*, *GitLab*, *Bitbucket*). Включены и процессные инструменты: практики ветвления (*GitFlow*) и системы ревью кода (*Gerrit*). В таблицах 39–40 представлены результаты по платформам *Android* и *iOS*.

**Таблица 39.** **Системы контроля версий и платформы для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Git | 207 | 30,09 | Git | 105 | 36,33 |
| GitHub | 25 | 3,63 | GitLab | 13 | 4,50 |
| GitLab | 15 | 2,18 | Apache Subversion (SVN) | 9 | 3,11 |
| Gerrit | 11 | 1,60 | GitHub | 9 | 3,11 |
| GitFlow | 9 | 1,31 | Gerrit | 6 | 2,08 |
| Apache Subversion (SVN) | 7 | 1,02 | Bitbucket | 4 | 1,38 |
| Bitbucket | 6 | 0,87 | GitFlow | 2 | 0,69 |

**Таблица 40.** **Системы контроля версий и платформы для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Git | 228 | 30,69 | Git | 97 | 35,79 |
| GitFlow | 23 | 3,10 | Apache Subversion (SVN) | 18 | 6,64 |
| GitLab | 21 | 2,83 | Mercurial | 12 | 4,43 |
| Bitbucket | 16 | 2,15 | GitHub | 10 | 3,69 |
| GitHub | 12 | 1,62 | GitLab | 4 | 1,48 |
| Apache Subversion (SVN) | 11 | 1,48 | Bitbucket | 3 | 1,11 |
| Mercurial | 3 | 0,40 | GitFlow | 2 | 0,74 |
| Gerrit | 2 | 0,27 |  |  |  |

Для ***Android*** лидирует *Git*; далее — платформы хостинга (*GitHub*, *GitLab*), затем — процессные инструменты (*Gerrit*, *GitFlow*). *SVN* и *Bitbucket* — реже. Для ***iOS*** также лидирует *Git*; далее — *GitFlow* и платформы управления репозиториями (*GitLab*, *Bitbucket*, *GitHub*). *SVN* и *Mercurial* — эпизодически. **Сопоставляя** **регионы**, картина стабильна: *Git* — стандарт во всех четырёх группах; при этом в Северной Америке заметнее доля централизованных/альтернативных систем (*SVN*, *Mercurial* — особенно в *iOS*), тогда как в Европе в *iOS*-подвыборке несколько выше удельный вес хостинг-платформ; различия невелики и не меняют общий порядок лидеров.

## 4.22. Спецификация и документация *API*

Категория *«api\_documentation»* охватывает формальные средства описания *HTTP-API* и инструменты публикации/просмотра документации. *OpenAPI* — это спецификация машинно-читаемого описания *API*; *Swagger* — сопутствующий набор инструментов (редактор, просмотр, генераторы). В таблицах 41–42 представлены результаты по платформам *Android* и *iOS*.

**Таблица 41.** **Спецификация и документация *API* для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Swagger | 4 | 0,58 | OpenAPI | 1 | 0,35 |
| OpenAPI | 2 | 0,29 | Swagger | 1 | 0,35 |

**Таблица 42.** **Спецификация и документация *API* для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| OpenAPI | 1 | 0,13 |  |  |  |
| Swagger | 1 | 0,13 |  |  |  |

Для ***Android*** общий объём упоминаний невысок: в европейской группе чаще встречается *Swagger*, тогда как в североамериканской *Swagger* и *OpenAPI* отмечены сопоставимо. Для ***iOS*** зафиксированы единичные упоминания *OpenAPI* и *Swagger* в европейской выборке; в предоставленных данных по Северной Америке упоминания отсутствуют. **Сопоставляя** **регионы**, профиль использования разрежен на обеих платформах: на *Android* упоминания присутствуют в обоих регионах, на *iOS* — только в Европе; *OpenAPI* и *Swagger* употребляются взаимодополняюще.

## 4.23. Тестирование и обеспечение качества

Категория *«testing\_quality»* охватывает инструменты и практики обеспечения качества: модульное и интеграционное тестирование, автоматизацию пользовательских сценариев (*UI*-тесты), моделирование зависимостей (моки/стабы), статический анализ кода и платформенные комплекты проверки совместимости. В таблицах 43–44 представлены результаты по платформам *Android* и *iOS*.

**Таблица 43.** **Тестирование и обеспечение качества для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Espresso | 43 | 6,25 | Espresso | 23 | 7,96 |
| JUnit | 35 | 5,09 | JUnit | 20 | 6,92 |
| Mockito | 18 | 2,62 | Mockito | 14 | 4,84 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Android Compatibility Test Suite (CTS) | 11 | 1,60 | Robolectric | 6 | 2,08 |
| SonarQube | 9 | 1,31 | Android Compatibility Test Suite (CTS) | 2 | 0,69 |
| Robolectric | 5 | 0,73 |  |  |  |
| Appium | 1 | 0,15 |  |  |  |

**Таблица 44.** **Тестирование и обеспечение качества для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| XCTest | 40 | 5,38 | XCTest | 16 | 5,90 |
| XCUITest | 5 | 0,67 | XCUITest | 3 | 1,11 |
| Appium | 2 | 0,27 | Appium | 1 | 0,37 |
| SonarQube | 2 | 0,27 | JUnit | 1 | 0,37 |
| SwiftLint | 2 | 0,27 | SonarQube | 1 | 0,37 |
| JUnit | 1 | 0,13 |  |  |  |
| Mockito | 1 | 0,13 |  |  |  |

Для ***Android*** лидирует *Espresso* (*UI*-тесты), далее *JUnit* (модульные тесты) и *Mockito* (моки/стабы); дополнительно встречаются *Android* *CTS*, *SonarQube* и *Robolectric*; *Appium* — точечно. Для ***iOS*** основная связка — *XCTest* (модульные/интеграционные тесты) и *XCUITest* (*UI*-тесты); дополнительно фиксируются *Appium* и *SonarQube*, *SwiftLint* — реже; единичные упоминания *JUnit/Mockito*. **Сопоставляя регионы**, распределение разрежено и концентрируется вокруг нативных средств тестирования на обеих платформах; инструменты статического анализа (например, *SonarQube*, *SwiftLint*) и кроссплатформенной автоматизации (например, *Appium*) встречаются заметно реже.

## 4.24. Фреймворки внедрения зависимостей (*DI*)

Категория *«dependency\_injection»* включает инструменты внедрения зависимостей (*Dependency* *Injection*, *DI*), применяемые для снижения связности модулей, упрощения тестирования и повышения модульности. В таблицах 45–46 представлены результаты по платформам *Android* и *iOS.*

**Таблица 45.** **Фреймворки внедрения зависимостей (*DI*) для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Dagger | 44 | 6,40 | Dagger | 53 | 18,34 |
| Hilt | 34 | 4,94 | Hilt | 19 | 6,57 |
| Koin | 10 | 1,45 | Koin | 4 | 1,38 |

**Таблица 46.** **Фреймворки внедрения зависимостей (*DI*) для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Swinject | 2 | 0,27 | Swinject | 3 | 1,11 |

Для ***Android*** верхние позиции стабильно занимают *Dagger* и *Hilt*; *Koin* фиксируется заметно реже. Для ***iOS*** отмечены единичные упоминания *Swinject*; других *DI*-фреймворков в выборке не выявлено. **Сопоставляя** **регионы**, структура в *Android* одинакова в Европе и Северной Америке (доминирует связка *Dagger/Hilt*), тогда как в *iOS* профиль разрежен с редкими упоминаниями *Swinject* без выраженных региональных различий.

## 4.25. Хранилища данных и СУБД

Категория *«data\_storage»* охватывает средства хранения данных в мобильной разработке: встраиваемые СУБД и объектные хранилища для локального сохранения на устройстве (например, *SQLite*, *Room*, *Core* *Data*, *Realm*), а также упоминания серверных/облачных СУБД и *NoSQL*-систем (*MySQL*, *MongoDB*, *Amazon* *DynamoDB*, *Microsoft* *SQL* *Server*) в контексте интеграции с бэкендом. В таблицах 47–48 представлены результаты по платформам *Android* и *iOS*.

**Таблица 47.** **Хранилища данных и СУБД для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Room | 54 | 7,85 | Room | 23 | 7,96 |
| SQLite | 31 | 4,51 | SQLite | 8 | 2,77 |
| Realm | 8 | 1,16 | Realm | 5 | 1,73 |
| NoSQL | 5 | 0,73 | MySQL | 4 | 1,38 |
| MySQL | 3 | 0,44 | Amazon DynamoDB | 1 | 0,35 |
| MongoDB | 1 | 0,15 | MongoDB | 1 | 0,35 |

**Таблица 48.** **Хранилища данных и СУБД для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Core Data | 102 | 13,73 | Core Data | 53 | 19,56 |
| Realm | 28 | 3,77 | SQLite | 10 | 3,69 |
| SQLite | 17 | 2,29 | MySQL | 9 | 3,32 |
| MySQL | 4 | 0,54 | Realm | 7 | 2,58 |
| NoSQL | 2 | 0,27 | NoSQL | 6 | 2,21 |
| Microsoft SQL Server | 1 | 0,13 | MongoDB | 5 | 1,85 |
| Room | 1 | 0,13 | Amazon DynamoDB | 2 | 0,74 |
|  |  |  | Microsoft SQL Server | 2 | 0,74 |

Для ***Android*** лидирует *Room*; далее — *SQLite*; *Realm* — реже. *NoSQL* и серверные СУБД (*MySQL*, *MongoDB*, *Amazon* *DynamoDB*) — эпизодически. Для ***iOS*** лидирует *Core* *Data*; далее — *Realm* и *SQLite*; серверные СУБД (*MySQL*, *MongoDB*, Amazon *DynamoDB*, *Microsoft* *SQL* *Server*) — точечно. **Сопоставляя** **регионы**, наблюдается устойчивый профиль: в обеих платформах верхние позиции занимают нативные/встраиваемые средства локального хранения (*Room*, *Core* *Data*), а слой серверных СУБД невелик в Европе и Северной Америке.

## 4.26. Экосистема *Android*

Категория *«platform\_android»* охватывает элементы экосистемы *Android*: базовый комплект разработчика (*Android* *SDK*), слои платформы (*AOSP*, *HAL*), сервисы и каналы распространения (*Google* *Play* *Services*), профильные среды для устройств (*Android* *Auto*, *Android* *TV*, *Fire* *TV*), мультимедийный стек (*ExoPlayer*) и модули дополненной реальности (*ARCore*). В таблице 49 представлены результаты по платформе *Android*.

**Таблица 49.** **Экосистема Android**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Android SDK | 181 | 26,31 | Android SDK | 111 | 38,41 |
| Android HAL | 11 | 1,60 | Android Auto | 7 | 2,42 |
| Android Auto | 7 | 1,02 | Google Play Services | 4 | 1,38 |
| Android Open Source Project (AOSP) | 4 | 0,58 | Android HAL | 3 | 1,04 |
| Android TV | 4 | 0,58 | Android Open Source Project (AOSP) | 1 | 0,35 |
| ExoPlayer | 4 | 0,58 | Android TV | 1 | 0,35 |
| ARCore | 3 | 0,44 | ExoPlayer | 1 | 0,35 |
| Fire TV | 3 | 0,44 |  |  |  |
| Google Play Services | 2 | 0,29 |  |  |  |

Для ***Android*** устойчиво лидирует *Android* *SDK*. Вспомогательные компоненты представлены эпизодически: слои платформы (*HAL*, *AOSP*), среды для устройств (*Android* *Auto*, *Android* *TV*, *Fire* *TV*), мультимедиа (*ExoPlayer*), *AR* (*ARCore*) и сервисы *Google* *Play*. **Сопоставляя** **регионы**, удельный вес *Android* *SDK* выше в Северной Америке, тогда как в Европе спектр вторых позиций шире (чаще встречаются *HAL/AOSP*, *Android* *TV*, *ExoPlayer*, *ARCore* и *Fire* *TV*); *Android* *Auto* присутствует в обеих подвыборках.

## 4.27. Экосистема *Apple*

Категория *«platform\_apple»* системные фреймворки и сервисы *Apple*, используемые при разработке под *iOS*: базовый набор *SDK*, подсистемы мультимедиа и веб-просмотра, геопозиционирование, машинное обучение, здравоохранение, а также интеграции с соплатформами (*watchOS*, *CarPlay*). В таблице 50 представлены результаты по платформе *iOS*.

**Таблица 50.** **Экосистема *Apple***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| iOS SDK | 35 | 4,71 | iOS SDK | 21 | 7,75 |
| Core ML | 34 | 4,58 | AVPlayer | 4 | 1,48 |
| HealthKit | 31 | 4,17 | Core Location | 4 | 1,48 |
| Core Location | 10 | 1,35 | WebKit | 4 | 1,48 |
| ARKit | 9 | 1,21 | AVFoundation | 3 | 1,11 |
| MapKit | 7 | 0,94 | HealthKit | 2 | 0,74 |
| AVFoundation | 4 | 0,54 | CarPlay | 1 | 0,37 |
| Core Bluetooth | 3 | 0,40 | Core ML | 1 | 0,37 |
| watchOS | 3 | 0,40 | MapKit | 1 | 0,37 |
| AVKit | 2 | 0,27 |  |  |  |
| Core Audio | 2 | 0,27 |  |  |  |
| Core Motion | 2 | 0,27 |  |  |  |
| Metal | 2 | 0,27 |  |  |  |
| CarPlay | 1 | 0,13 |  |  |  |

Для ***iOS*** устойчивым лидером в обоих регионах выступает *iOS* *SDK* (базовый набор библиотек и *API*). В Европе далее чаще упоминаются *Core* *ML* (средства машинного обучения) и *HealthKit* (здоровье и фитнесс); далее идут геосервисы (*Core* *Location*, *MapKit*) и *AR*-компоненты (*ARKit*). В Северной Америке после базового *SDK* заметнее мультимедиа и веб-стек — *AVPlayer*, *WebKit*, *AVFoundation* — вместе с *Core* *Location*; *HealthKit* и *Core* *ML* упоминаются реже. Специализированные модули (*watchOS*, *CarPlay*, *Core* *Audio*, *Core* *Motion*, *Metal*, *AVKit*) — точечные упоминания. **Сопоставляя** **регионы**, профиль концентрируется вокруг базового *SDK* с умеренными сдвигами: в Европе акцент на *ML/health*, в Северной Америке — на мультимедиа и веб-компоненты. Существенных перестановок лидеров не наблюдается.

## 4.28. Языки программирования и разметки

Категория *«languages»* охватывает языки программирования и разметки, упоминаемые в требованиях к мобильным разработчикам. В таблицах 51–52 представлены результаты по платформам *Android* и *iOS*.

**Таблица 51.** **Языки программирования и разметки для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Kotlin | 600 | 87,21 | Kotlin | 208 | 71,97 |
| Java | 368 | 53,49 | Java | 186 | 64,36 |
| C++ | 79 | 11,48 | C++ | 29 | 10,03 |
| C | 23 | 3,34 | C | 23 | 7,96 |
| Python | 22 | 3,20 | JavaScript | 20 | 6,92 |
| JavaScript | 15 | 2,18 | Python | 12 | 4,15 |
| C# | 8 | 1,16 | C# | 6 | 2,08 |
| SQL | 8 | 1,16 | PHP | 3 | 1,04 |
| TypeScript | 6 | 0,87 | SQL | 3 | 1,04 |
| Dart | 4 | 0,58 | CSS | 1 | 0,35 |
| Go | 4 | 0,58 | Dart | 1 | 0,35 |
| Rust | 3 | 0,44 | Go | 1 | 0,35 |
| Apache Groovy | 2 | 0,29 | HTML | 1 | 0,35 |
| PHP | 2 | 0,29 | Ruby | 1 | 0,35 |
| Scala | 2 | 0,29 | TypeScript | 1 | 0,35 |
| Ruby | 1 | 0,15 |  |  |  |

**Таблица 52.** **Языки программирования и разметки для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Swift | 664 | 89,37 | Swift | 230 | 84,87 |
| Objective-C | 276 | 37,15 | Objective-C | 148 | 54,61 |
| C++ | 24 | 3,23 | JavaScript | 26 | 9,59 |
| JavaScript | 18 | 2,42 | C++ | 13 | 4,80 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Python | 13 | 1,75 | Python | 9 | 3,32 |
| TypeScript | 10 | 1,35 | SQL | 8 | 2,95 |
| C | 7 | 0,94 | C | 8 | 2,95 |
| C# | 6 | 0,81 | C# | 5 | 1,85 |
| PHP | 5 | 0,67 | Go | 4 | 1,48 |
| SQL | 5 | 0,67 | PHP | 2 | 0,74 |
| Go | 4 | 0,54 | Dart | 2 | 0,74 |
| Ruby | 3 | 0,40 | TypeScript | 2 | 0,74 |
| Dart | 2 | 0,27 | Visual Basic | 2 | 0,74 |
| CSS | 1 | 0,13 | HTML | 1 | 0,37 |
| HTML | 1 | 0,13 | Ruby | 1 | 0,37 |
|  |  |  | Rust | 1 | 0,37 |
|  |  |  | CSS | 1 | 0,37 |

Для ***Android*** базовый стек формируют *Kotlin* и *Java*; *C/C++* встречаются умеренно, прочие языки общего назначения и языки запросов — эпизодически. Для ***iOS*** доминирует *Swift*, при сохранении заметной роли *Objective-C* (поддержка/миграция наследуемого кода); *C/C++* упоминаются умеренно, остальные языки фиксируются единично. **Сопоставляя** **регионы**, внутри каждой платформы состав лидеров стабилен (*Android — Kotlin/Java; iOS — Swift/Objective-C*); различия касаются прежде всего набора редких упоминаний.