ЭСТОНСКИЙ УНИВЕРСИТЕТ ПРЕДПРИНИМАТЕЛЬСТВА МАЙНОР

Учебная программа: Веб-дизайн и цифровая графика

Специализация: Веб-технологии

Ivan Ireev

**Определение наиболее востребованных 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: [Имя Фамилия]

[дд. мм. гггг]

РЕЗЮМЕ

Текст.

RESÜMEE

Текст.

ABSTRACT

Текст.

ОГЛАВЛЕНИЕ

[ДЕКЛАРАЦИЯ ОБ АВТОРСКИХ ПРАВАХ 1](#_Toc193715130)

[РЕЗЮМЕ 2](#_Toc193715131)

[RESÜMEE 3](#_Toc193715132)

[ABSTRACT 4](#_Toc193715133)

[ВВЕДЕНИЕ 7](#_Toc193715134)

[1. АНАЛИЗ СУЩЕСТВУЮЩИХ ИССЛЕДОВАНИЙ 10](#_Toc193715135)

[1.1. Обзор исследований по рынку труда в *IT*-сфере 10](#_Toc193715136)

[1.2. Исследования, фокусирующиеся на *iOS* и *Android* разработчиках 12](#_Toc193715137)

[1.3. Выводы и пробелы в исследованиях 15](#_Toc193715138)

[2. ВЫБОР ИНСТРУМЕНТОВ И ПОДХОДОВ 17](#_Toc193715139)

[2.1. *SerpApi* 17](#_Toc193715140)

[2.2. *Langdetect* 17](#_Toc193715141)

[2.3. *ChatGPT-4o* 18](#_Toc193715142)

[2.4. *Fuzzy matching* 19](#_Toc193715143)

[3. МЕТОДИКА ИССЛЕДОВАНИЯ 21](#_Toc193715144)

[3.1. Подготовка к сбору данных 21](#_Toc193715145)

[3.2. Сбор данных 21](#_Toc193715146)

[3.3. Создание набора данных 22](#_Toc193715147)

[3.4. Первичная очистка данных 22](#_Toc193715148)

[3.5. Определение языка вакансий 22](#_Toc193715149)

[3.6. Перевод не англоязычных вакансий на английский 23](#_Toc193715150)

[3.7. Разделение вакансий на логические части 24](#_Toc193715151)

[ОГРАНИЧЕНИЯ 26](#_Toc193715152)

[РЕЗУЛЬТАТЫ 27](#_Toc193715153)

[ЗАКЛЮЧЕНИЕ 28](#_Toc193715154)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc193715155)

[ПРИЛОЖЕНИЯ 31](#_Toc193715156)

[Приложение 1. Полный список стран 31](#_Toc193715157)

[Приложение 2. Директории с вакансиями 32](#_Toc193715158)

[Приложение 3. Единый набор данных 33](#_Toc193715159)

[Приложение 4. Промпт для разбиения вакансий на логические части 34](#_Toc193715160)

ВВЕДЕНИЕ

Мобильная разработка представляет собой процесс создания программного обеспечения, предназначенного для работы на мобильных устройствах, таких как смартфоны и планшеты. Рынок мобильных приложений активно развивается и 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)) в ответ на увеличение 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)), рост количества скачиваний приложений ([Qadir, 15 апреля 2025](#Qadir_2025_15_апреля_Mobile_App_downlds); [Statista, 2024](#Statista_2024_Number_of_mobile_app)), рост потребительских расходов в приложениях ([Briskman, февраль 2025](#Briskman_2025_Q4_2024_digital_mrket_Indx)) и всё более глубокое проникновение цифровых технологий во все сферы человеческой жизни. В совокупности это повышает спрос на мобильных разработчиков.

В ответ на растущий спрос увеличивается и число разработчиков мобильных приложений. 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)) — тем самым формируя ощутимый кадровый спрос.

Соответственно, работодатели активно нанимают мобильных разработчиков. Известно, что только в США в 2024 году 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)). Таким образом, важно понять, как на этот спрос влияет стремительно развивающийся генеративный ИИ.

На этом фоне волна генеративного ИИ, охватившая индустрию в 2023–2024 гг., не снизила, а усилила спрос на мобильных разработчиков, поскольку рынок «*Gen-AI*»-приложений (чат-боты, *AI art* 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 *SDK*-модули искусственного интеллекта прямо на устройство (*on-device AI SDK*), 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.

Одним из первых таких вызовов становится постоянная адаптация к стремительной эволюции технологий: платформы *iOS* и *Android* регулярно выпускают обновления, внедряют новые языки программирования, фреймворки и подходы к разработке. Это создаёт ситуацию, в которой разработчикам становится трудно своевременно ориентироваться в меняющихся требованиях рынка и выбирать наиболее актуальные и востребованные инструменты, а университетские программы теряют актуальность и выпускают менее конкурентоспособных выпускников.

## Актуальность темы

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.

Актуальность данного исследования обусловлена потребностью в комплексном анализе ключевых тенденций мобильной разработки. 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 востребованные технологии и составить рекомендации, 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 (*ChatGPT-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: кроссплатформенные и смешанные вакансии удаляются после полной разметки, поскольку именно она надёжно показывает истинную платформу.
9. На случайной подвыборке объявлений (*n* ≥ 30) извлечь упомянутые технологии и инструменты (из *Requirements*, *Nice to Have*, *Responsibilities*) при помощи *ChatGPT-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.
    * удалить платформенно несоответствующие упоминания (например, *Swift*— из пула *Android*, *Kotlin*— из пула *iOS*), чтобы не искажать результаты (точность определения блока *Platform* не будет 100 %; часть неверно классифицированных вакансий может остаться).
12. Вручную сгруппировать технологии и инструменты по категориям (языки программирования, системы контроля версий и др.).
13. Представить полученные результаты в интерактивном дашборде *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

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

В выборку включены вакансии всех уровней — *junior*, *middle* и *senior*; далее разделения на эти уровни не будет. Базовый набор языков программирования и инструментов по уровню должности практически не меняется; различаются преимущественно глубина владения ими, масштаб решаемых задач и объём ответственности (Источник). А инструменты сверх базы больше зависят от конкретного места работы и отрасли, а также устоявшихся на этом месте традиций и legacy и специфики работы. Поэтому в исследовании требования будут рассматриваться для обобщённого условного «среднего» мобильного разработчика.

В данном исследовании под «востребованностью навыков» понимается их относительная частота упоминаний в вакансиях: чем выше доля объявлений с упоминанием конкретного навыка, тем выше считается его востребованность.

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 остается 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 отчеты регулярно фиксируют глобальные тенденции. К примеру, *Stack Overflow Developer Survey* ([2024](#Stack_Overflow_2024_Annual_Dev_Survey)) представляет широкий срез *IT*-специалистов по всему миру (более 50 тысяч респондентов) и содержит данные о наиболее распространённых языках программирования, технологиях и профессиональных ролях. Подобные исследования дают ценную макрокартину, но не фокусируются на отдельных платформах: мобильные разработчики выступают в них лишь малой частью выборки (лишь около 3,4% от всех респондентов составляют *mobile*-разработчики ([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 более 500 специалистов в области информационных систем и технологий, включая работодателей, разработчиков и рекрутеров. 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 ([Data Nerds, n.d.](#Data_Nerds_n_d_)), разработанный *Luke Barousse* ([2023](#Barousse_Luke_2023)), представляет собой аналитическую платформу, исследующую востребованные навыки и уровень заработных плат в сфере информационных технологий (*IT*). Для сбора данных о вакансиях сервис использует *SerpApi*, получая информацию из результатов поиска *Google*. В среднем ежедневно обрабатывается около 6500 вакансий, преимущественно относящихся к *data science* и смежным направлениям. Полученные данные проходят предобработку и анализ с использованием *NLP*-пайплайна, после чего агрегируются для выявления наиболее востребованных 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 по наиболее часто упоминаемым в вакансиях языкам программирования, инструментам, базам данных, облачным сервисам, библиотекам и фреймворкам, а также анализ динамики популярности различных технологий с течением времени. 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*, проведенные командой Яндекс Практикума ([2024, 3 июня](#Яндекс_Практикум_2024_03_06_Android); [2024, 10 июня](#Яндекс_Практикум_2024_10_06_iOS)), 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, 2024, 10 июня](#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, 2024, 3 июня](#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 проведен 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); [Cummings et al., 2023](#Cummings_Janicki_Matthews_2023); [Data Nerds, n.d.](#Data_Nerds_n_d_); [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 разработчиков программного обеспечения, однако в них мобильные специалисты не выделяются как отдельная категория, что затрудняет выявление специфических требований именно к *iOS*- и *Android*-разработчикам.

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

Исследования Яндекс Практикума ([2024, 3 июня](#Яндекс_Практикум_2024_03_06_Android); [2024, 10 июня](#Яндекс_Практикум_2024_10_06_iOS)) 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. *SerpApi*

*SerpApi* — это сервис, предоставляющий *API* для получения результатов поисковых систем в реальном времени, включая *Google*, *Bing*, *Yahoo* и другие. Он обеспечивает доступ к структурированным данным поисковой выдачи, облегчая извлечение и обработку информации для различных целей. ([SerpApi, n.d. -a](#SerpApi_n_d_Google_Search_API))

В данном исследовании *SerpApi* используется для сбора вакансий *iOS*- и *Android*-разработчиков в Европе и Северной Америке. *SerpApi* позволяет автоматизировать процесс сбора данных о вакансиях, обеспечивая доступ к актуальной информации с различных поисковых систем и сайтов.

Преимущества использования *SerpApi* в исследовании:

* Автоматизированное решение для обхода *CAPTCHA*. В отличие от стандартных методов веб-скрейпинга, *SerpApi* автоматически обходит *CAPTCHA*, что значительно упрощает процесс сбора вакансий ([SerpApi, n.d. -a](#SerpApi_n_d_Google_Search_API));
* Доступ к структурированным данным. *SerpApi* предоставляет структурированные данные о вакансиях, включая описание должностей, требования, зарплаты (если указаны) и местоположение ([SerpApi, n.d. -b](#SerpApi_n_d_Google_Jobs_API));
* Гибкость в настройке поисковых параметров. *SerpApi* поддерживает использование различных локальных доменов *Google* (например, *google*.*de*, *google*.*fr*), что позволяет получать результаты, специфичные для конкретных стран или регионов ([SerpApi, n.d. -b](#SerpApi_n_d_Google_Jobs_API));
* Точный поиск по ключевым словам. *SerpApi* позволяет формулировать точные поисковые запросы, такие как *«iOS developer»* или *«Android developer»*, что помогает находить наиболее релевантные вакансии ([SerpApi, n.d. -b](#SerpApi_n_d_Google_Jobs_API)).

Благодаря перечисленным преимуществам *SerpApi* был выбран в качестве оптимального инструмента для сбора данных о вакансиях в данном исследовании.

## 2.2. *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 работы, в условиях офлайн-анализа и умеренного объёма данных её производительности достаточно. Простота интеграции делают её удобным инструментом для применения в исследованиях. Дополнительным преимуществом является наличие у автора опыта работы с *Langdetect*, что позволило быстрее приступить к анализу.

## 2.3. *ChatGPT-4o*

​*ChatGPT-4o* — это мультимодальная большая языковая модель, разработанная *OpenAI* и выпущенная 13 мая 2024 года. Она способна обрабатывать и генерировать текст, изображения и аудио в реальном времени, обеспечивая более быстрые и эффективные результаты по сравнению с предыдущими моделями. ([GPT-4o, n.d.](#Wikipedia_n_d_GPT_4o))

В данном исследовании используется *API ChatGPT-4o* версии *gpt-4o-2024-11-20*.

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

Структура промпта была выбрана на основе исследования ([Gao, Wang & Hou, 2023](#Gao_Wang_Hou_2023)), где эмпирически тестировались различные формулировки для машинного перевода с использованием *ChatGPT* (*GPT-3.5*). В рамках исследования авторы сравнивали *ChatGPT* с *Google* *Translate* и *DeepL* — двумя наиболее мощными коммерческими системами машинного перевода. Они использовали их в качестве эталонных систем, чтобы понять, насколько *ChatGPT* приближается (или уступает) им в качестве перевода.

Результаты экспериментов показали, что оптимальная формулировка промпта может существенно повысить точность перевода. *ChatGPT* (*GPT-3.5*) превзошёл *Google* *Translate* и *DeepL* в переводе на английский, согласно метрикам *BLEU* и *ChrF++*.

В данной работе была адаптирована наиболее эффективная версия промпта из исследования, но с использованием *GPT-4o* — более совершенной модели.

## 2.4. *Fuzzy matching*

*Fuzzy matching* (нечёткое сопоставление) — это метод, используемый для сравнения текстовых данных с учётом возможных орфографических вариаций, перестановок слов и незначительных расхождений в формулировках. В отличие от строгого (точного) сопоставления строк, *fuzzy matching* позволяет выявлять схожесть текстов, даже если они содержат частичные совпадения или небольшие различия. ([DataCamp, 2025](#DataCamp_2025_Fuzzy_String_Matching))

В данном исследовании метод *fuzzy matching* был выбран для оценки точности автоматического разбиения вакансий на логические части с использованием *API ChatGPT-4o*. Основная причина выбора данного подхода заключается в том, что модель *ChatGPT-4o* не всегда возвращает текст в точности в том же виде, что и исходный, но при этом сохраняет смысл и структуру данных. Поэтому применение нечёткого сопоставления позволяет объективно измерить степень соответствия между автоматической и ручной разметкой, не наказывая за незначительные различия в формулировках.

**2.4.1. Token Set Ratio**

Одной из наиболее эффективных метрик для сравнения текстов в рамках *fuzzy matching* является *Token Set Ratio*. Она основана на классическом методе — расстоянии Левенштейна (*Levenshtein Distance*), которое вычисляет минимальное число операций (вставки, удаления или замены) для преобразования одной строки в другую ([DataCamp, 2025](#DataCamp_2025_Fuzzy_String_Matching)).

Однако *Token Set Ratio* отличается от расстояния Левенштейна следующим ([DataCamp, 2025](#DataCamp_2025_Fuzzy_String_Matching)):

1. Токенизация строк — разбиение текста на отдельные слова (токены);
2. Удаление дубликатов — одинаковые слова внутри каждой строки не учитываются повторно;
3. Приведение к одному регистру — сравнение проводится без учёта регистра букв;
4. Расчёт сходства — после нормализации строк вычисляется расстояние Левенштейна между отсортированными множествами токенов.

Эта метрика особенно эффективна при анализе текстов вакансий, где одна и та же информация может быть представлена разными формулировками, но при этом иметь одинаковый смысл.

**2.4.2. 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 [Приложение 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*), заменённые 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* *ChatGPT-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 *ChatGPT-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 *ChatGPT-4o*.

Для перевода вакансий на английский язык применялся следующий промпт (подробнее об обосновании выбора промпта см. [раздел 2.3.1.](#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 ChatGPT-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 ChatGPT-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* *ChatGPT-4o* на основе заранее сформулированного промпта (см. [Приложение 4](#_Приложение_2._Промпт)).

Для оценки степени соответствия двух методов разметки был применён метод *fuzzy matching* с использованием метрики *Token Set Ratio* из библиотеки *rapidfuzz* (подробнее см. [раздел 2.4.](#_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 *ChatGPT-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* *ChatGPT*-*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 *ChatGPT-4o*.

Из 2846 строк набора 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 *ChatGPT*-*4o* обоснованно не смогла определить платформу. Такие строки были исключены как информационно неполные.

Таким образом, после всех этапов фильтрации в итоговый набор данных вошли 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 *ChatGPT-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* *ChatGPT-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* *ChatGPT-4o* на основе заранее сформулированного промпта ([см. Приложение 6](#_Приложение_6._Промпт)).

Для оценки степени соответствия двух методов разметки был применён метод *fuzzy* *matching* с использованием метрики *Token* *Set* *Ratio* из библиотеки *rapidfuzz* (подробнее см. [раздел 2.4.](#_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 *ChatGPT-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*). Категории упорядочены и представлены в алфавитном порядке; после них следует сводный «Итоговый блок» (*Total*), объединяющий самые популярные технологии вне зависимости от категории. Для каждой платформы формируются отдельные таблицы единого формата:

***Technology | EU count | EU % | Technology | 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*

Таким образом, проценты не сравниваются напрямую между группами, а отражают внутреннюю структуру каждой выборки.

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)). Интерактивные визуализации доступны в дашборде *Tableau* ([Ireev, 2025r](#Ireev_I_2025r_tableau_dashboard)). Набор данных, использованный для создания таблиц и визуализаций, представлен в виде *CSV*-файла ([Ireev, 2025s](#Ireev_I_2025s_df_filtered_long_csv)).

ОГРАНИЧЕНИЯ

## 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.

## 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. Это сделано сознательно — чтобы не дробить классификацию и не создавать отдельную категорию ради одной позиции. Указанное ограничение следует учитывать при интерпретации результатов по категориям.

РЕЗУЛЬТАТЫ

Перед прочтением данного раздела автор рекомендует ознакомиться с разделами «[3.12. Формат представления результатов](#_3.12._Формат_представления)» и «[Ограничения](#_Ограничения_классификации)», в которых подробно описаны принципы формирования и представления данных, а также указаны нюансы, которые необходимо учитывать при интерпретации. Результаты исследования представлены в виде таблиц; интерактивные визуализации для детального самостоятельного изучения данных доступны в дашборде *Tableau* ([Ireev, 2025r](#Ireev_I_2025r_tableau_dashboard)).

## Архитектурные шаблоны программного обеспечения

Категория «*architecture*» охватывает архитектурные подходы к структурированию клиентских приложений, включая принципы разделения ответственности, управление состоянием и потоком данных. В таблицах 1–2 представлены результаты по платформам *Android* и *iOS*.

**Таблица 1. 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 2. 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 |
| View-Interactor-Presenter-Entity-Router (VIPER) | 47 | 6,33 | Model-View-Presenter (MVP) | 16 | 5,9 |
| 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*** доминирует *MVVM*, далее следуют *MVP* и *Clean* *Architecture*; *MVI* и *MVC* представлены умеренно, *Redux* и *VIPER* — единично. Для ***iOS*** также лидирует *MVVM*; вторую позицию занимает *MVC*, а *VIPER* и *MVP* формируют следующую группу; *Clean* *Architecture* встречается реже, *VIP* и *MVI* — точечно. В обеих платформах фиксируется высокая концентрация в верхних рангах (лидирующие паттерны), при наличии «длинного хвоста» редких упоминаний. Пересечение лидеров между регионами внутри каждой платформы устойчиво: *MVVM* занимает первую позицию во всех рассмотренных группах, тогда как распределение последующих позиций варьирует без выраженных атипичностей.

## Инструменты командного взаимодействия и управления разработкой

Категория *«team\_collaboration»* охватывает программные средства, используемые для координации командной работы в разработке: системы учёта задач и инцидентов, корпоративные базы знаний и инструменты обмена проектными материалами между дизайном и разработкой. В таблицах 3–4 представлены результаты по платформам *Android* и *iOS*.

**Таблица 3. 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 4. Инструменты командного взаимодействия и управления разработкой для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Jira | 37 | 4,98 | Jira | 16 | 5,9 |
| 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*** структура упоминаний сконцентрирована вокруг решений от *Atlassian* — производителя корпоративных инструментов: *Jira* (управление задачами/проектами) занимает ведущую позицию, за ней следует *Confluence* (база знаний и документации). *Figma* встречается эпизодически как платформа совместной работы над интерфейсами и канал передачи макетов и спецификаций разработчикам; *Trello* фиксируется точечно как лёгкая канбан-система. Для ***iOS*** наблюдается аналогичный профиль: лидирует *Jira*, далее *Confluence*; *Figma* упоминается несколько чаще как средство совместного редактирования макетов и обмена спецификациями, *Zeplin* сохраняет нишевое присутствие как инструмент формализации дизайнерских спецификаций; *Trello* — единичные случаи. В целом результаты указывают на устойчивую стандартизацию практик вокруг пары *Jira*–*Confluence* при ограниченной роли дополнительных средств для обмена дизайнерскими артефактами.

## 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 5–6 представлены результаты по платформам *Android* и *iOS*.

**Таблица 5. Интегрированные среды разработки (*IDE*) для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Android Studio | 124 | 18,02 | Android Studio | 63 | 21,8 |
| Eclipse | 2 | 0,29 | Eclipse | 4 | 1,38 |

**Таблица 6. Интегрированные среды разработки (*IDE*) для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Xcode | 142 | 19,11 | Xcode | 71 | 26,2 |

Для ***Android*** наблюдается выраженная концентрация упоминаний на *Android* *Studio* как официальной среде разработки; альтернативы представлены минимально (*Eclipse* — эпизодически). Для ***iOS*** профиль однозначный: *Xcode* выступает стандартной средой разработки и фактически формирует монокультуру инструментов. В совокупности результаты указывают на высокую степень стандартизации *IDE* внутри каждой платформенной экосистемы и ограниченную вариативность вне нативных решений.

## Интероперабельность и взаимодействие с нативным кодом

Категория *«native\_interop»* охватывает средства взаимодействия мобильных приложений с нативным кодом и системными сервисами: встраивание модулей на *C/C++*, межъязыковые вызовы между *Kotlin/Java* и *C/C++*, а также механизмы межпроцессного обмена данными и вызова удалённых методов. В таблице 7 представлены результаты по платформе *Android*.

**Таблица 7. Интероперабельность и взаимодействие с нативным кодом для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Android NDK | 44 | 6,4 | 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*** профиль устойчив: лидирует *Android* *NDK* (набор инструментов для разработки и подключения *C/C++*-модулей), далее *JNI* (мост между *Java/Kotlin* и нативным кодом); реже встречаются *AIDL* (язык описания интерфейсов для межпроцессного взаимодействия) и *Binder* (базовый *IPC*-механизм). Для ***iOS*** в рассматриваемой выборке явных упоминаний не зафиксировано.

## 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 8–9 представлены результаты по платформам *Android* и *iOS*.

**Таблица 8. 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 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Open Web Application Security Project (OWASP) | 1 | 0.15 |  |  |  |

**Таблица 9. 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 | 0.4 | OAuth | 4 | 1.48 |
| OAuth 2.0 | 3 | 0.4 | 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*** отмечаются единичные упоминания нормативного регламента *GDPR* (защита персональных данных в ЕС), протоколов *OAuth/OAuth 2.0* (авторизация доступа к ресурсам) и рекомендаций *OWASP* (ориентиры безопасной разработки). Для ***iOS*** стабильно лидирует системное хранилище *Keychain* (безопасное хранение ключей и учётных данных); далее следуют *OAuth/OAuth 2.0*. Встречаются точечные ссылки на криптографические механизмы (*AES* — симметричное шифрование; *RSA* — асимметричное шифрование) и контроль транспортной безопасности (*TLS* *pinning* — привязка проверки сертификата/ключа сервера). В целом профиль категории характеризуется низкими абсолютными частотами и концентрацией вокруг нескольких базовых понятий: хранение секретов на уровне платформы, авторизация по отраслевому стандарту и соблюдение регуляторных и методических ориентиров.

## Исполняемые среды и мультиплатформенные рантаймы

Категория *«runtimes\_shared»* охватывает среды выполнения (*runtime*) и технологические стек-решения, используемые для совместного использования кода между платформами и для унификации исполняемого окружения в проектах мобильной разработки. К типичным представителям относятся *Kotlin* *Multiplatform* (общие модули на *Kotlin* для *Android* и *iOS*), *Node.js* (среда выполнения *JavaScript/TypeScript*, часто фигурирует как часть единого технологического стека и инструментов) и *.NET Core* (кроссплатформенная среда выполнения экосистемы *.NET*). 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 %** |
| Kotlin Multiplatform | 29 | 4,22 | Kotlin Multiplatform | 11 | 3,81 |
| Node.js | 4 | 0,58 | Node.js | 4 | 1,38 |

**Таблица 11. Информационная безопасность и нормативное соответствие 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* фиксируется как второстепенное упоминание. Для ***iOS*** картина более разреженная: *Kotlin* *Multiplatform* присутствует в европейской подвыборке, а в североамериканской дополнительно отмечается *.NET* *Core* как нишевый вариант. В целом распределение указывает на узкий набор практик с выраженной концентрацией вокруг *Kotlin* *Multiplatform* и умеренной ролью прочих сред, используемых преимущественно для интеграции и сопутствующей инфраструктуры.

## Кроссплатформенные *SDK* для устройств и сервисов

Категория *«cross\_platform\_sdks»* охватывает наборы средств разработчика (*SDK*), обеспечивающих единый доступ к возможностям устройства и внешним сервисам на разных платформах: системные уведомления, беспроводные интерфейсы, геолокация, камера и датчики, а также интеграции со сторонними сервисами (карты, социальные сети, платежи). В таблицах 12–13 представлены результаты по платформам *Android* и *iOS*.

**Таблица 12. 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 13. 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 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Camera | 3 | 0,4 | Camera | 2 | 0,74 |
| 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 |

Во **всех** рассмотренных группах устойчиво лидируют *Notifications*, что отражает базовую роль *push*-уведомлений в мобильных продуктах. Следующий «пояс» составляют *Bluetooth/BLE* и *Location*; *Camera* и *Sensors* упоминаются реже, а *NFC* — точечно. Набор сторонних *SDK* представлен умеренно: *Facebook* *SDK* и *Google* *Maps* встречаются эпизодически, платёжные решения (*Stripe* *SDK*, *PayPal* *SDK*) — нишево. В целом распределение указывает на приоритет универсальных системных возможностей, тогда как интеграции с внешними сервисами добавляются выборочно под задачи продукта.

## Кроссплатформенные *UI*-фреймворки

Категория *«cross\_platform\_ui»* охватывает фреймворки, позволяющие разрабатывать пользовательские интерфейсы для нескольких платформ на общей кодовой базе, что снижает дублирование реализации и упрощает координацию релизов между экосистемами. В таблицах 14–15 представлены результаты по платформам *Android* и *iOS*.

**Таблица 14. Кроссплатформенные *UI*-фреймворки для *Android***

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

**Таблица 15. 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 | 0,4 | Xamarin | 4 | 1,48 |

Для ***Android*** устойчиво доминирует пара *Flutter* и *React* *Native* с перестановкой лидера между регионами; *Apache* *Cordova*, *Xamarin* и *Ionic* фиксируются заметно реже и образуют «второй эшелон». Для ***iOS*** первой остаётся *React* *Native*, вслед за которой следует *Flutter*; *Xamarin*, *Ionic* и *Apache* *Cordova* представлены ограниченно. В целом структура указывает на высокую концентрацию вокруг пары *React* *Native* и *Flutter* с «длинным хвостом» редких упоминаний альтернативных фреймворков и ограниченной вариативностью за пределами топ-группы.

## Методологии и практики управления разработкой

Категория *«methodologies»* охватывает методологические подходы и организационные практики управления разработкой программного обеспечения: гибкие методы (*Agile*), их процессные фреймворки (*Scrum*, *Kanban*), инженерные практики обеспечения качества (*Test-Driven* *Development*, *TDD*; *Behavior-Driven* *Development*, *BDD*), масштабируемые модели для крупных организаций (*Scaled* *Agile* *Framework*, *SAFe*), а также классические 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 | 4,8 |
| 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*** во всех региональных подвыборках лидирует *Agile*; далее стабильно следует *Scrum*, а третью позицию занимает *TDD*. *Kanban* фиксируется заметно реже; *BDD* и *SAFe* встречаются точечно, в Северной Америке дополнительно присутствуют единичные упоминания *Waterfall*. Для ***iOS*** наблюдается та же иерархия рангов (*Agile* → *Scrum* → *TDD*), с более умеренным представлением *Kanban* и эпизодическим присутствием *BDD*, *SAFe* и *Waterfall*. В целом профиль распределений указывает на высокую концентрацию в верхних позициях и содержательное совпадение лидеров между платформами и регионами, при ограниченной роли альтернативных или узкоспециализированных практик.

## Мониторинг и аналитика

Категория *«monitoring\_analytics»* охватывает программные средства наблюдаемости и аналитики клиентских приложений: сбор и агрегирование отчётов о сбоях, мониторинг ошибок и производительности, а также поведенческую (продуктовую) и базовую веб/мобильную аналитику. 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 | 0,4 |  |  |  |
| Mixpanel | 2 | 0,27 |  |  |  |
| Amplitude | 1 | 0,13 |  |  |  |
| App Center | 1 | 0,13 |  |  |  |
| Datadog | 1 | 0,13 |  |  |  |

Для ***Android*** ведущие позиции занимает *Crashlytics* (сервис экосистемы *Firebase* для отчётов о падениях). Прочие решения представлены эпизодически: *Datadog* (платформа наблюдаемости), *Sentry* (мониторинг ошибок), инструменты продуктовой аналитики (*Amplitude*), а также *Google* *Analytics* и *App* *Center* (диагностика и отчёты о сбоях). Для ***iOS*** профиль схожий: доминирует *Crashlytics*; дополнительно фиксируются упоминания *Google* *Analytics* и *Sentry*; решения поведенческой аналитики (*Mixpanel*, *Amplitude*) встречаются реже; *App* *Center* и *Datadog* — точечно. В целом профиль категории характеризуется преобладанием решений *crash/error* *reporting* при умеренном охвате продуктовой аналитики и единичных инфраструктурных инструментов; распределение упоминаний разрежено, с концентрацией в нескольких наиболее узнаваемых сервисах.

## Нативные *UI*-фреймворки и компоненты

Категория *«native\_ui»* охватывает платформенные средства построения пользовательского интерфейса: фреймворки и базовые компоненты, обеспечивающие разметку экранов, обработку взаимодействий, рендеринг 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 |
| Cocoa Touch | 40 | 5,38 | Cocoa Touch | 21 | 7,75 |
| WatchKit | 29 | 3,9 | WatchKit | 1 | 0,37 |

Для *Android* профиль упоминаний сосредоточен вокруг *Jetpack* *Compose* — современного фреймворка с декларативной моделью описания интерфейсов; дополнительные позиции занимают библиотека *Glide* (загрузка и кэширование изображений в интерфейсе) и системный компонент *WebView* (встраивание веб-контента), причём их присутствие носит эпизодический характер. Для *iOS* лидирует *SwiftUI* — декларативный фреймворк для построения интерфейсов; на второй позиции — *UIKit* как традиционная императивная основа нативных приложений. *Core* *Animation* и *Cocoa* *Touch* представлены как инфраструктурные элементы нативного стека, *WatchKit* фиксируется нишево для сценариев *watchOS*. Общая картина указывает на консолидацию практик вокруг фреймворков текущего поколения при инерционной поддержке унаследованных решений.

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

Категория *«ci\_cd»* охватывает инструменты автоматизации сборки, тестирования и поставки мобильных приложений, а также управление конвейерами автоматизации (*pipeline*) — от подготовки артефактов до публикации и внутреннего распространения. В выборку входят универсальные серверы автоматизации (например, *Jenkins*), специализированные решения для мобильной разработки (например, *fastlane*, *Bitrise*), интегрированные механизмы в экосистемах систем контроля версий (например, *GitHub* *Actions*) и средства контейнеризации (например, *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*** наблюдается устойчивое лидерство *Jenkins*; далее формируется «средний слой» из *Bitrise*, *Docker*, *GitHub* *Actions* и *fastlane*; *CircleCI*, *Azure* *DevOps*, *Bamboo* и *TeamCity* присутствуют эпизодически. Для ***iOS*** структура схожа, однако в европейской подвыборке лидирует *fastlane* с последующим *Jenkins*, тогда как в североамериканской — *Jenkins* опережает *fastlane*; *Bitrise*, *CircleCI* и *GitHub* *Actions* формируют следующий эшелон. Платформенные облачные сервисы (*Xcode* *Cloud*, *Unity* *Cloud* *Build*) встречаются точечно и не меняют общей картины доминирования универсальных систем общего назначения при заметной роли специализированных мобильных инструментов.

## Отладка, профилирование и реверс-инжиниринг

Категория *«debug\_profiling»* охватывает программные средства и практики для анализа исполнения приложений. В рамках данной категории под отладкой понимается диагностика и воспроизведение дефектов, под профилированием — измерение ресурсной нагрузки (время *CPU*, память, ввод-вывод), под реверс-инжинирингом — исследование скомпилированного кода и артефактов сборки для восстановления структуры и поведения. 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*), при точечных ссылках на *Android* *Profiler*. Для ***iOS*** фиксируются лишь *Cydia* *Substrate* и *Frida*, что указывает на эпизодическое использование инструментов динамической модификации/инструментирования. В целом наблюдается низкая частота упоминаний и отсутствие выраженных лидеров; профиль отражает нишевой характер задач отладки на низком уровне и реверс-инжиниринга.

## Параллелизм и реактивное программирование

Категория *«concurrency\_reactive»* охватывает технологии, применяемые для организации одновременного выполнения задач и обработки асинхронных событий в мобильных клиентах. Под параллелизмом понимается координация нескольких вычислительных операций (например, сетевых запросов и вычислений) без блокировки пользовательского интерфейса; под реактивным программированием — моделирование потоков событий и зависимостей между ними. 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* (языковая модель структурированного асинхронного выполнения), далее — *RxJava* из семейства *ReactiveX*; *Kotlin* *Flow* фиксируется как дополнительный инструмент потоковой обработки, *RxKotlin* встречается эпизодически. Для ***iOS*** верхние позиции занимают *Apple* *Combine* (встроенный в экосистему *Apple* фреймворк реактивных потоков) и *RxSwift* (семейство *ReactiveX*); *Grand* *Central* *Dispatch* *(GCD)* присутствует как системный механизм диспетчеризации задач, *Swift* *Concurrency* отмечается единично. В рамках каждой платформы наблюдается устойчивая картина с высокой концентрацией в первых рангах и ограниченным «длинным хвостом» дополнительных упоминаний; пересечения лидеров между регионами внутри платформ сохраняются без выраженных атипичностей.

## 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*** лидируют упоминания *Google Play Store / Google Play Console* — официального портала для загрузки сборок, управления релизами и метаданными приложения; дополнительно точечно встречается *Google Play Billing* как механизм приёма платежей и подписок. Для ***iOS*** на первых позициях *App Store / App Store Connect* — системный контур публикации и сопровождения релизов; *TestFlight* фиксируется как средство распространения предварительных сборок для бета-тестирования. В целом профиль указывает на высокую степенью стандартизации вокруг официальных магазинов платформ при ограниченной роли вспомогательных сервисов (монетизация, бета-дистрибуция).

## Разработка игр

Категория *«game\_dev»* охватывает специализированные средства и технологии, применяемые при создании игровых приложений: игровые движки, фреймворки, инструменты управления контентом и сопутствующие программные компоненты. 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 фрагменте данных упоминания отсутствуют. Для ***iOS*** зафиксированы единичные упоминания *Unity* *Engine* в европейской выборке; в североамериканской группе упоминаний не отмечено. В совокупности результаты указывают на крайне ограниченное присутствие игровых технологий в рассматриваемых вакансиях, при этом единичные случаи для *iOS* сосредоточены вокруг *Unity* как наиболее распространённого решения в мобильной разработке игр.

## Руководства по дизайну интерфейсов

Категория *«design\_guidelines»* охватывает официальные руководства по проектированию пользовательских интерфейсов, определяющие композицию, навигационные паттерны, визуальные и типографские решения, а также требования к 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 от *Google* для проектирования интерфейсов. Для ***iOS*** наблюдается устойчивое присутствие ссылок на *Human* *Interface* *Guidelines* — официальные руководства *Apple*; упоминания представлены в обеих региональных подвыборках. В совокупности результаты указывают на ориентацию работодателей на нормативные источники платформодержателей, при этом низкая доля явных ссылок (особенно в *Android*) может отражать практику подразумеваемого следования руководствам без отдельной фиксации в тексте вакансий.

## Сборка и управление зависимостями

Категория *«build\_dependency»* охватывает средства для автоматизации сборки и управления зависимостями в проектах мобильной разработки. Под «сборкой» понимается настройка и выполнение процессов компиляции, тестирования и упаковки артефактов; под «управлением зависимостями» — декларативное подключение и обновление 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*** наблюдается выраженная концентрация вокруг *Gradle* как стандартной системы сборки экосистемы; *Apache* *Maven* занимает второстепенные позиции. Единичные упоминания *Yocto* *Project* и *Makefile* указывают на точечные сценарии с элементами встраиваемых систем или низкоуровневых сборочных цепочек. Для ***iOS*** лидирует менеджер зависимостей *CocoaPods*; *Swift* *Package* *Manager* стабильно занимает вторую позицию, что согласуется с его интеграцией в инструментарий языка *Swift*. *Carthage* встречается эпизодически. Упоминания *Bazel*, а также нетипичных для *iOS* инструментов (*Gradle*, *Apache* *Maven*, *npm*) носят ограниченный характер и, вероятно, отражают кросс-платформенные или инфраструктурные задачи, где требуется унификация сборки на уровне организации. В целом профиль категории демонстрирует специализацию по платформам при наличии «длинного хвоста» универсальных инструментов, используемых в отдельных контекстах.

## Серверные платформы и облачные сервисы

Категория *«backend\_baas»* включает управляемые серверные платформы и общие облачные сервисы, используемые мобильными командами для хранения данных, аутентификации, выполнения прикладной логики и интеграций. Сюда относятся решения класса *Backend-as-a-Service* (например, *Firebase*, *AWS* *Amplify*, *Parse* *Platform*), а также универсальные публичные облака и их сервисы (*Amazon* *Web* *Services*, *Google* *Cloud* *Platform*, *Microsoft* *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 | 0 | 0 | 0 |

Для ***Android*** профиль упоминаний концентрируется вокруг *Firebase* как базовой управляемой платформы; далее следуют универсальные облака (*AWS*) и точечные ссылки на отдельные сервисы и компоненты (*Cloud* *Firestore*, *Azure*, *S3*). Единичные упоминания *Spring* *Boot* указывают на сценарии взаимодействия с собственными серверными приложениями помимо *BaaS*. Для ***iOS*** картина схожа: ведущие позиции занимают *Firebase* и *AWS*, присутствуют специализированные сервисы (*AWS* *Lambda*, *S3*) и редкие упоминания альтернатив (*Amplify*, *Parse* *Platform*, *GCP*, *Azure*). В целом наблюдается высокая концентрация вокруг крупных облачных экосистем с ведущей ролью *Firebase*, при ограниченном, но стабильном слое упоминаний отдельных облачных компонентов и серверных стеков.

## Сетевые протоколы и клиентские библиотеки

Категория *«networking»* охватывает средства сетевого взаимодействия мобильных приложений с удалёнными сервисами: модели прикладных интерфейсов (*REST*, *GraphQL*), транспорт и протоколы для обмена данными в реальном времени (*WebSockets*, *WebRTC*), механизмы удалённых вызовов процедур (*gRPC*) и сериализацию сообщений (*Protocol* *Buffers*), а также клиентские библиотеки для выполнения *HTTP*-запросов и управления соединениями (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 | 12.8 |
| 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 |
| Protocol Buffers | 2 | 0.27 | Retrofit | 1 | 0.37 |
| gRPC | 2 | 0.27 | URLSession | 1 | 0.37 |

Для ***Android*** фиксируется устойчивое лидерство *REST* *API*; среди клиентских средств выделяется связка *Retrofit* (высокоуровневый *HTTP*-клиент) и *OkHttp* (низкоуровневая *HTTP*-библиотека). *GraphQL* представлен заметно, но реже, тогда как *WebSockets* используются эпизодически для двунаправленного обмена; *gRPC* и *Protocol* *Buffers* встречаются точечно, что указывает на ограниченное применение *RPC*-подхода на клиенте. Для ***iOS*** наблюдается схожий профиль с доминированием *REST* *API*; нативный *URLSession* фиксируется наряду со сторонними обёртками *Alamofire* и *Moya* для упрощения работы с *HTTP*. *GraphQL* и *WebSockets* присутствуют, *WebRTC* отмечает отдельные сценарии мультимедийного реального времени, а *gRPC* и *Protocol* *Buffers* остаются нишевыми. В целом результаты указывают на стандартизацию интеграций вокруг *REST*, при умеренном, контекстно-зависимом использовании альтернативных парадигм.

## Системы контроля версий и платформы

Категория *«version\_control»* включает системы контроля версий и связанные с ними платформы управления репозиториями и изменениями кода. Под системами контроля версий понимаются распределённые решения (*Git*, *Mercurial*) и централизованные (*Apache Subversion (SVN)*); к платформам относятся сервисы хостинга и совместной работы с репозиториями (*GitHub*, *GitLab*, *Bitbucket*). Также учитываются практики и инструменты, регламентирующие процесс разработки: *GitFlow* (процедуры ветвления и выпуска) и *Gerrit* (система ревью кода и согласования изменений). 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 | 4.5 |
| GitLab | 15 | 2.18 | Apache Subversion (SVN) | 9 | 3.11 |
| Gerrit | 11 | 1.6 | 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 | 3.1 | Apache Subversion (SVN) | 18 | 6.64 |
| GitLab | 21 | 2.83 | Mercurial | 12 | 4.43 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| 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 | 0.4 | GitFlow | 2 | 0.74 |
| Gerrit | 2 | 0.27 |  |  |  |

Для ***Android*** наблюдается выраженное доминирование *Git*; следующую группу формируют платформы хостинга (*GitHub*, *GitLab*), за которыми следуют специализированные инструменты процесса (*Gerrit*, *GitFlow*) и упоминания альтернативных/централизованных систем (*SVN*, *Bitbucket*). Для ***iOS*** картина сопоставима по структуре лидеров: на первом месте *Git*; далее фиксируются практики ветвления (*GitFlow*) и платформы управления репозиториями (*GitLab*, *Bitbucket*, *GitHub*). В обеих платформах, пусть и реже, фиксируются упоминания *SVN* и *Mercurial*, что отражает сохранение практик с использованием централизованных и альтернативных распределённых систем наряду с *Git*. Итоговая картина указывает на выраженный фокус на *Git* при умеренной вариативности сопутствующих платформ (*GitHub*, *GitLab*, *Bitbucket*) и процессных инструментов (*GitFlow*, *Gerrit*).

## Спецификация и документация *API*

Категория *«api\_documentation»* охватывает формальные средства описания и публикации спецификаций прикладных интерфейсов (*API*), а также сопутствующие инструменты генерации документации и артефактов разработки. В терминологии области *OpenAPI* — это спецификация для машинно-читаемого описания *HTTP-API*, тогда как *Swagger* — набор инструментов вокруг этой спецификации (включая редактор, визуальный просмотр и генераторы кода). 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 выборке 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. В целом наблюдается точечное и взаимодополняющее употребление названий *OpenAPI* и *Swagger* (спецификация и инструменты), без выраженных атипичностей распределения внутри платформ.

## Тестирование и обеспечение качества

Категория *«testing\_quality»* охватывает инструменты и практики, применяемые для обеспечения качества программного обеспечения: 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 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Mockito | 18 | 2.62 | Mockito | 14 | 4.84 |
| Android Compatibility Test Suite (CTS) | 11 | 1.6 | 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 | 5.9 |
| 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* (автоматизация *UI*-тестов), за которым следуют *JUnit* (модульные тесты на *JVM*) и *Mockito* (моделирование зависимостей). Отмечены упоминания *Android* *Compatibility* *Test* *Suite* *(CTS)* как платформенного набора тестов на совместимость, *SonarQube* как системы статического анализа и *Robolectric* как среды запуска *Android*-тестов на *JVM*; *Appium* — единично. Для ***iOS*** основная связка — *XCTest* (модульные/интеграционные тесты) и *XCUITest* (*UI*-тесты); дополнительно встречаются *Appium* (кроссплатформенная *UI*-автоматизация), *SonarQube* (статический анализ) и *SwiftLint* (линтер стиля кода *Swift*). Единичные упоминания *JUnit* и *Mockito* отражают либо включённость *JVM*-компонентов в проект, либо общекомандные подходы к тестированию вне строгой привязки к платформе. В целом обе платформы демонстрируют доминирование нативных средств тестирования при вспомогательной роли инструментов статического анализа и кроссплатформенной автоматизации.

## Фреймворки внедрения зависимостей

Категория *«dependency\_injection»* охватывает программные средства, реализующие внедрение зависимостей (*Dependency Injection*, *DI*) — приём проектирования, при котором необходимые объекту компоненты передаются извне. Такой подход снижает связность модулей, упрощает тестирование и поддерживает модульность архитектуры. В таблицах 45–46 представлены результаты по платформам *Android* и *iOS.*

**Таблица 45.** **Фреймворки внедрения зависимостей для *Android***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Dagger | 44 | 6.4 | 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 для *iOS***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Swinject | 2 | 0.27 | Swinject | 3 | 1.11 |

Для ***Android*** в обеих региональных группах верхние позиции занимают *Dagger* и *Hilt* (*DI*-фреймворк для *Android*, основанный на *Dagger*); *Koin* фиксируется существенно реже. Для ***iOS*** отмечаются редкие упоминания *Swinject*; иных *DI*-фреймворков в представленной выборке не выявлено. В сумме наблюдается высокая концентрация в верхних позициях у *Android* с доминированием стека *Dagger/Hilt* и минимальная вариативность в *iOS*-подвыборках.

## Хранилища данных и СУБД

Категория *«data\_storage»* охватывает средства хранения данных в мобильной разработке: встраиваемые СУБД и объектные хранилища для локального хранения данных 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 клиентских приложений 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* (рекомендуемая *Google* надстройка над *SQLite*), далее следует *SQLite* как базовая встраиваемая СУБД; *Realm* встречается реже как объектное хранилище. Упоминания *NoSQL* и серверных СУБД (*MySQL*, *MongoDB*, *Amazon* *DynamoDB*) носят эпизодический характер. Для ***iOS*** доминирует *Core* *Data* (стандартная технология *Apple* для управления объектной моделью и хранением данных), далее фиксируются *Realm* и *SQLite*; упоминания серверных систем (*MySQL*, *MongoDB*, *Amazon* *DynamoDB*, *Microsoft* *SQL* *Server*) присутствуют точечно. В целом для обеих платформ характерна высокая концентрация вокруг нативных/встраиваемых средств локального хранения данных и «длинный хвост» редких упоминаний серверных СУБД.

## 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 | 1.6 | 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*** отмечается выраженная концентрация на *Android* *SDK*, стабильно занимающем первую позицию во всех группах. Вспомогательные компоненты представлены точечно: платформенные слои (*HAL*, *AOSP*), среды для устройств (*Android* *Auto*, *Android* *TV*, *Fire* *TV*), а также *ExoPlayer*, *ARCore* и *Google* *Play* *Services*. Вне лидера наблюдается «длинный хвост» без атипичных всплесков.

## 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 49.** **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 | 0.4 | Core ML | 1 | 0.37 |
| watchOS | 3 | 0.4 | 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. В целом наблюдается концентрация вокруг базового *SDK* и нескольких прикладных стеков при «длинном хвосте» редких упоминаний.

## Языки программирования и разметки

Категория *«languages»* охватывает языки программирования и разметки, упоминаемые в требованиях к мобильным разработчикам. В таблицах 50–51 представлены результаты по платформам *Android* и *iOS*.

**Таблица 50.** **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 | 3.2 | 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 | 0 | 0 | 0 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Swift | 664 | 89.37 | Swift | 230 | 84.87 |
| **Technology** | **EU count** | **EU %** | **Technology** | **NA count** | **NA %** |
| Objective-C | 276 | 37.15 | Objective-C | 148 | 54.61 |
| C++ | 24 | 3.23 | JavaScript | 26 | 9.59 |
| JavaScript | 18 | 2.42 | C++ | 13 | 4.8 |
| 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 | 0.4 | 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. Внутри каждой платформы лидеры устойчивы в обоих регионах; различия касаются преимущественно состава редких упоминаний.

## Total

ЗАКЛЮЧЕНИЕ

Текст.

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>

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>

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>

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>

GPT-4o. (n.d.). *Wikipedia*. <https://en.wikipedia.org/wiki/GPT-4o>

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)

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

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>

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

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

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

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

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>

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>

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

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

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

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

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

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>

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

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>

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>

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>

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>

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

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/>

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/>

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>

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>

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

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/>

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 X. Список технологий для проверки