МОБИЛЬНАЯ РАЗРАБОТКА

ВВЕДЕНИЕ В СПЕЦИАЛЬНОСТЬ

СОДЕРЖАНИЕ

РЫНОК МОБИЛЬНЫХ ОПЕРАЦИОННЫХ СИСТЕМ

- Android: ~70-72% мирового рынка
- iOS: ~28-30% мирового рынка
- Совокупно: 99%+ рынка
- Google Play Store: ~3.5+ млн приложений
- App Store: ~1.6+ млн приложений
- Новые игроки: HarmonyOS, Аврора (РФ)
- Linux-решения: PinePhone, Ubuntu Touch

КЛЮЧЕВЫЕ ТРЕНДЫ

- Импортозамещение и рост значимости ОС Аврора
- AI/ML на устройстве (On-device AI)
- Foldable и складные устройства
- 5G и edge computing
- Супер-приложения (Super Apps)
- Усиление требований к безопасности

ОСНОВНЫЕ ПОДХОДЫ К РАЗРАБОТКЕ

- Нативная разработка — под конкретную платформу
- Кроссплатформенная — один код для всех платформ
- Гибридная (WebView) — веб-технологии в нативной оболочке
- Прогрессивные Web-приложения (PWA)

НАТИВНАЯ РАЗРАБОТКА

ПЛАТФОРМА ANDROID

ANDROID: ОСОБЕННОСТИ ПЛАТФОРМЫ

- Открытая экосистема (AOSP — Android Open Source Project)
- Множество производителей: Samsung, Xiaomi, Honor и др.
- Фрагментация версий и устройств
- Основные языки: Java, Kotlin
- Современный стандарт: Kotlin + Jetpack Compose

ANDROID: АРХИТЕКТУРА ПРИЛОЖЕНИЯ

- Activity — экран приложения
- Fragment — модульная часть экрана
- Service — фоновые задачи
- Broadcast Receiver — системные события
- Content Provider — обмен данными
- Intent — навигация и коммуникация

ANDROID: ИНСТРУМЕНТАРИЙ РАЗРАБОТЧИКА

- Android Studio (на базе IntelliJ IDEA)
- Android SDK, Platform Tools
- Эмулятор Android (AVD)
- Система сборки: Gradle (Kotlin DSL)
- Профилировщик производительности

KOTLIN — СОВРЕМЕННЫЙ СТАНДАРТ ANDROID

- Полная совместимость с Java
- Лаконичный и безопасный синтаксис
- Null-безопасность
- Coroutines для асинхронности
- Data classes, extension functions
- Поддержка Jetpack Compose

JETPACK COMPOSE

- Современный UI toolkit (декларативный подход)
- Аналог SwiftUI и Flutter
- Меньше кода, нагляднее
- Реактивное обновление интерфейса
- Пример:

```
@Composable
fun Greeting(name: String) {
    Text(text = "Hello, $name!")
}
```

НАТИВНАЯ РАЗРАБОТКА

ПЛАТФОРМА IOS

iOS: ОСОБЕННОСТИ ПЛАТФОРМЫ

- Закрытая экосистема (только устройства Apple)
- Строгая модерация приложений (App Store)
- Минимальная фрагментация
- Высокая безопасность и конфиденциальность
- Основные языки: Objective-C, Swift
- Современный стандарт: Swift + SwiftUI

iOS: АРХИТЕКТУРА ПРИЛОЖЕНИЯ

- UIKit (классический) / SwiftUI (современный)
- ViewController — контроллер экрана
- AppDelegate / SceneDelegate — жизненный цикл
- Storyboards / XIB (устаревают)
- MVC, MVVM, Coordinator паттерны

SWIFT

- Безопасный и быстрый
- Современный синтаксис
- Optionals для безопасной работы с nil
- Protocol-Oriented Programming
- Value types (structs)

SWIFTUI

- Декларативный фреймворк интерфейсов
- Живой превью в Xcode
- Единый код для всех платформ Apple
- Автоматическая поддержка темной темы
- Пример:

```
struct ContentView: View {
    var body: some View {
        Text("Hello, World!")
    }
}
```

Xcode — среда разработки для iOS

- Официальная IDE от Apple
- Включает симулятор устройств
- Инструменты: Interface Builder, Instruments
- Система сборки: Xcodebuild
- Требует macOS для разработки

ОСНОВЫ JAVA

JAVA: СОВРЕМЕННОЕ СОСТОЯНИЕ

- Текущая LTS версия: Java 21 (сентябрь 2023)
- Релизный цикл: каждые 6 месяцев
- Java Runtime Environment (JRE) — для запуска
- Java Development Kit (JDK) — для разработки
- OpenJDK — эталонная реализация

JAVA: ПОСТАВЩИКИ JDK

- Oracle JDK — для коммерческого использования (платный)
- OpenJDK — открытая версия
- Eclipse Temurin (бывш. AdoptOpenJDK) — бесплатный LTS
- Amazon Corretto — бесплатный LTS
- Azul Zulu — бесплатный LTS

JVM: ВИРТУАЛЬНАЯ МАШИНА JAVA

- Исполняет байт-код (.class)
- Управление памятью (Garbage Collection)
- JIT-компиляция (Just-In-Time)
- Основные реализации: HotSpot, OpenJ9
- Поддерживает Kotlin, Scala, Groovy

КРОССПЛАТФОРМЕННАЯ РАЗРАБОТКА

ОДИН КОД — ВЕЗДЕ

ЗАЧЕМ НУЖНА КРОССПЛАТФОРМА?

- Экономия ресурсов (одна команда)
- Единая кодовая база
- Быстрый выход на рынок (MVP)
- Унификация бизнес-логики
- Недостатки: производительность, доступ к нативным API

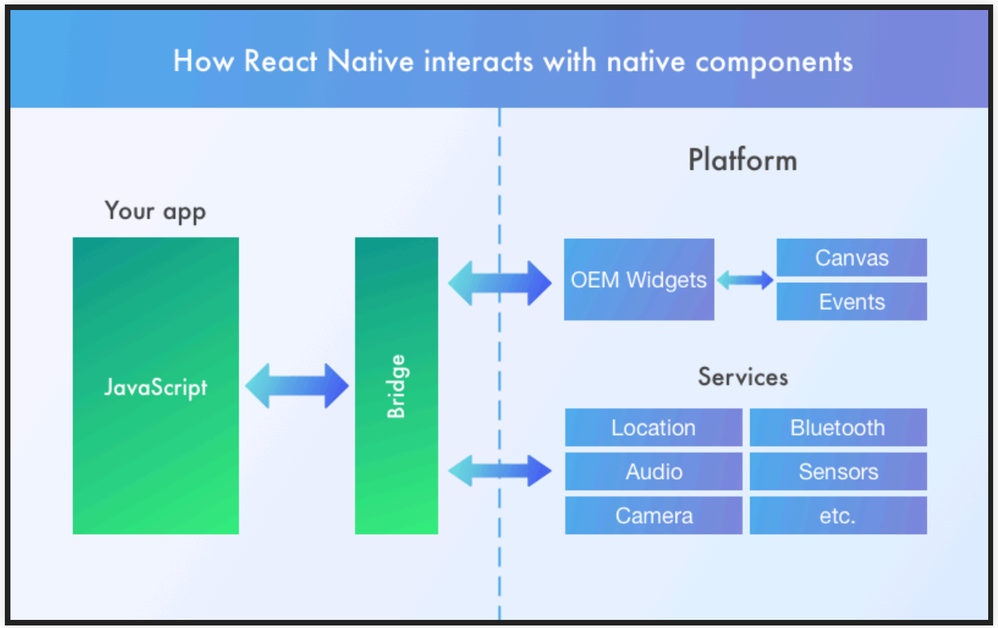
ОСНОВНЫЕ КРОССПЛАТФОРМЕННЫЕ РЕШЕНИЯ

- React Native (JavaScript/TypeScript)
- Flutter (Dart)
- .NET MAUI (C#)
- Kotlin Multiplatform (KMM)

REACT NATIVE

- Создан 2015
- Использует React и JavaScript
- Нативные компоненты моста (Bridge)
- Горячая перезагрузка (Fast Refresh)
- Большое сообщество и экосистема
- Примеры: Instagram, Discord, Shopify

REACT NATIVE: АРХИТЕКТУРА



JavaScript поток + Нативный мост

REACT NATIVE: НОВЫЙ ПОДХОД (FABRIC)

- Новая архитектура (с 0.68)
- JSI (JavaScript Interface) вместо моста
- Синхронный вызов нативных функций
- Улучшение производительности
- Turbomodules и Fabric Renderer

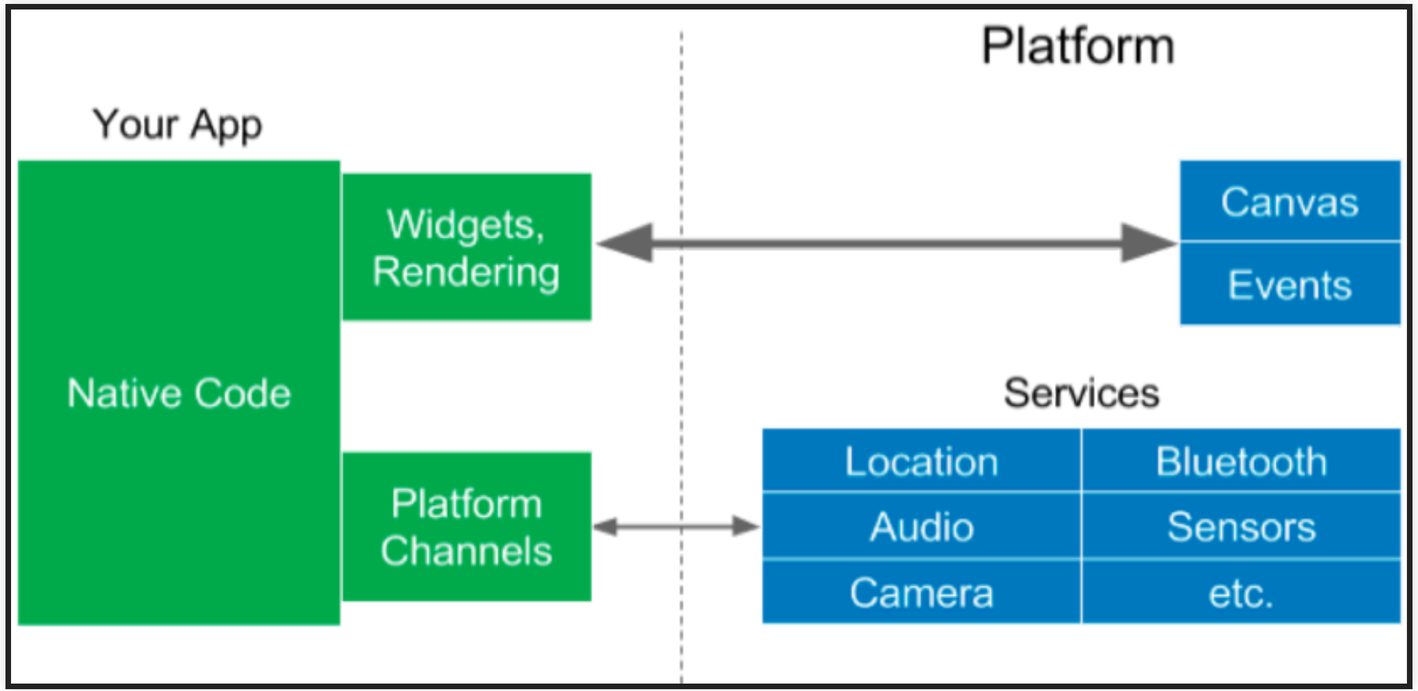
TYPESCRIPT В REACT NATIVE

- Статическая типизация
- Лучшая поддержка IDE
- Улучшенная документация кода
- Стандарт для крупных проектов
- Интеграция с React Native CLI

FLUTTER

- Создан Google (2018)
- Язык Dart
- Собственный движок рендеринга (Skia/Impeller)
- Высокая производительность (близко к нативной)
- Hot Reload за миллисекунды
- Единый UI для всех платформ

FLUTTER: АРХИТЕКТУРА



FLUTTER: ПЛАТФОРМЫ

- iOS, Android
- Web (через WASM)
- Windows, macOS, Linux
- Embedded устройства
- Google Fuchsia

DART: ОСОБЕННОСТИ ЯЗЫКА

- Строгая типизация
- Поддержка AOT и JIT компиляции
- Асинхронность (Future, Stream, async/await)
- Изоляты (Isolates) для многопоточности
- Hullsafety с Dart 2.12
- Сборщик мусора (Generational GC)

DART: AOT VS JIT

Режим	Использование	Преимущества
JIT	Разработка	Hot reload, быстрая итерация
AOT	Релиз	Макс. производительность, быстрый старт

DART: ИЗОЛЯТЫ (ISOLATES)

- Альтернатива потокам
- Нет общей памяти
- Обмен через порты (SendPort/ReceivePort)
- Избегание состояний гонки
- Идеально для параллельных вычислений

FLUTTER: ВИДЖЕТЫ

- Всё — виджет
- StatelessWidget vs StatefulWidget
- Material (Android) и Cupertino (iOS)
- Композиция вместо наследования
- Древовидная структура

FLUTTER: ПРИМЕР ПРИЛОЖЕНИЯ

```dart
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Flutter Demo')),
        body: const Center(
          child: Text('Hello, World!'),
```

FLUTTER: **ЛОКАЛИЗАЦИЯ** (L10N)

- Использование ARB файлов
- Генерация классов локализации
- Поддержка множественных чисел
- Плюрализация и интерполяция
- Пример app_ru.arb:

```
{
  "hello": "Привет",
  "appTitle": "Мое приложение"
}
```

FLUTTER: ПЛЮСЫ

- Высокая производительность
- Единый UI на всех платформах
- Быстрая разработка (hot reload)
- Отличная документация
- Растущее сообщество

FLUTTER: **МИНУСЫ**

- Размер приложения
- Молодость экосистемы
- Необходимость изучения Dart
- Ограниченные нативные API
- Web-версия уступает нативным PWA

РАЗРАБОТКА ДЛЯ ОС АВРОРА

РОССИЙСКАЯ МОБИЛЬНАЯ ПЛАТФОРМА

ЧТО ТАКОЕ ОС АВРОРА?

- Российская мобильная ОС (ранее Sailfish Mobile OS RUS)
- Разработчик: ООО "Открытая мобильная платформа"
- Входит в реестр отечественного ПО
- Основана на ядре Linux и Mer
- Интерфейс на Qt5
- Ориентация на корпоративный и госсектор

АРХИТЕКТУРА ОС АВРОРА

- Ядро Linux
- Промежуточный слой: Middleware (Qt, Lipstick)
- Пользовательское окружение: Silica
- Приложения: нативные (C++/Qt/QML) или гибридные
- Поддержка Android-приложений (через слой совместимости)

ТЕХНОЛОГИИ РАЗРАБОТКИ ПОД АВРОРУ

- Основной язык: C++, QML, JavaScript
- Фреймворк: Qt 5.15+
- UI-компоненты: Silica (аналог Qt Quick Controls)
- Среда разработки: Qt Creator + Aurora SDK
- Система сборки: CMake, qmake
- Пакетный менеджер: RPM

AURORA SDK

- Набор инструментов для разработки
- Включает эмулятор устройств Авроры
- Эмулятор на базе VirtualBox
- Набор утилит для сборки, отладки, профилирования
- Интеграция с Qt Creator
- Документация и примеры кода

ЖИЗНЕННЫЙ ЦИКЛ ПРИЛОЖЕНИЯ НА АВРОРЕ

- Состояния: остановлено, загружено, активно, покрыто, фон
- Управление ресурсами и памятью
- Обработка событий сворачивания/восстановления
- Сохранение состояния
- Жесткая политика энергосбережения

UI: SILICA КОМПОНЕНТЫ

- Адаптированы под жестовое управление
- SilicaFlickable — пролистывание
- SilicaListView — списки
- SilicaCover — обложки приложений
- SilicaWebView — встроенный браузер
- Поддержка тем оформления

ПРИМЕР QML НА АВРОРЕ

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Application {
    initialPage: Component {
        Page {
            Column {
                anchors.centerIn: parent
                Label {
                    text: "Hello, Aurora!"
                    color: Theme.primaryColor
                }
                Button {
                    text: "Click me"
```

ОСОБЕННОСТИ ПУБЛИКАЦИИ В АВРОРЕ

- Магазин приложений: RuStore (скоро), собственный магазин
- Требуется сертификация для корпоративных решений
- Проверка безопасности ФСТЭК
- Упаковка в RPM пакеты
- Подписание приложений

ПРЕИМУЩЕСТВА РАЗРАБОТКИ ПОД АВРОРУ

- Господдержка и импортозамещение
- Растущий рынок в РФ
- Ниша B2B и госсектора
- Меньшая конкуренция среди разработчиков
- Привычные технологии (Qt/C++)

ВЫЗОВЫ И СЛОЖНОСТИ

- Небольшое сообщество
- Ограниченная документация
- Меньше готовых библиотек
- Фрагментация устройств
- Требования к безопасности

ПРОГРЕССИВНЫЕ ВЕБ-ПРИЛОЖЕНИЯ (PWA)

- Работают через браузер
- Push-уведомления, офлайн-режим
- Доступ к камере, геолокации (ограниченно)
- Не требуют установки из магазина
- Примеры: Twitter Lite, Telegram Web

СРАВНЕНИЕ ПОДХОДОВ

Подход	Производительность	Стоимость	UI/UX
Нативный	Высокая	Высокая	Платформенный
Flutter	Высокая	Средняя	Кастомный
React Native	Средняя	Средняя	Нативный
PWA	Низкая	Низкая	Веб

НЕОБХОДИМЫЕ НАВЫКИ

- ООП, алгоритмы, структуры данных
- Знание платформы (Android/iOS/Aurora)
- Работа с API, базами данных
- Git, системы контроля версий
- Английский язык (документация)
- Soft skills: коммуникация, работа в команде

ИНСТРУМЕНТЫ ОБЩЕГО НАЗНАЧЕНИЯ

- Git (GitHub, GitLab, Bitbucket)
- CI/CD: Jenkins, GitHub Actions, GitLab CI
- Дизайн: Figma, Sketch
- Управление проектами: Jira, Trello
- Тестирование: Postman, Charles Proxy

ЧТО ДАЛЬШЕ?

- Изучайте Kotlin и Swift
- Пробуйте Flutter для стартапов
- Следите за Авророй (российский рынок)
- Участвуйте в open source
- Пишите pet-проекты

РЕСУРСЫ ДЛЯ ОБУЧЕНИЯ

- Android: developer.android.com
- iOS: developer.apple.com
- Flutter: flutter.dev
- ОС Аврора: developer.auroraos.ru
- Stepik, Coursera, Udemy

АНОНС

- Интенсив: "Разработка кроссплатформенных приложений с использованием Flutter на примере российской мобильной ОС Аврора"
- Регистрация: https://leader-id.ru/events/594991
- Разработчики от ОМП
- Даты: 21 апреля 2026, 10:00 — 23 апреля 2026, 17:30
- Освобождение от занятий