

Kotlin in the New Era

From Android Apps to MVPs, AI Features, and Scalable Backend Systems — 3 Days

FA-0280 · 3 Days

Software Development · DevOps, Cloud & Infrastructure

Introduction

Kotlin has grown far beyond its early reputation as a cleaner way to write Android code. Today, Kotlin developers are expected to understand how mobile apps connect with APIs, cloud services, security layers, data streams, AI features, and product goals. A modern app is rarely just an app; it is usually one part of a wider digital product.

This course is designed to help Kotlin and Android developers move from feature-level development to product-aware engineering. Participants will learn how to build stronger Kotlin applications, structure code more clearly, apply MVVM correctly, improve performance, secure applications, integrate AI, and understand how mobile systems connect with backend and cloud services.

A key part of the course is the **MVP, or Minimum Viable Product**, mindset. Participants will learn how to identify the smallest useful version of a product, build it properly, validate it early, and improve it based on real needs rather than assumptions.

The course will be led by an instructor with over **30 years of industry experience**, using real industry-documented content rather than overly academic material. The latest Kotlin ecosystem continues to support mobile, server-side, desktop, web, and multiplatform development, while recent Android 16 changes place stronger emphasis on security, privacy, behavior changes, and app compatibility.

Learning outcomes

By the end of this course, participants will be able to:

- Understand the role of Kotlin in modern Android and backend development
- Design cleaner and more maintainable Kotlin applications
- Apply MVVM and ViewModel best practices in real-world applications

- Plan and structure an MVP for a Kotlin-based product
 - Identify core features, supporting features, and future enhancements
 - Improve application security using authentication, authorization, and secure coding practices
 - Optimize app performance, memory usage, threading, and application size
 - Prepare Android applications for platform changes from Android 12 to Android 16
 - Integrate AI features using on-device and API-based approaches
 - Understand how Kotlin applications connect with backend services and microservices
 - Understand the basics of event-driven systems and real-time data flow
 - Build applications with better scalability, maintainability, and industry relevance
-

Prerequisites

- Basic knowledge of Kotlin programming
 - Basic familiarity with Android application development
 - Understanding of object-oriented programming concepts
 - Basic knowledge of REST APIs and networking
 - Familiarity with Git or version control
 - Optional but helpful: basic Java knowledge
 - Optional but helpful: exposure to backend development concepts
-

Training outline

1. Modern Kotlin Development Overview

1. Kotlin's role in modern software development
2. Kotlin for Android applications
3. Kotlin for backend and server-side systems
4. Kotlin in multiplatform development
5. How mobile applications fit into larger digital systems
6. Moving from simple app development to product-focused engineering
7. Industry expectations for modern Kotlin developers

2. MVP Thinking for Kotlin Applications

1. Meaning of Minimum Viable Product

2. Difference between prototype, MVP, and full product
3. Identifying the main business problem
4. Defining the target user
5. Selecting the core feature set
6. Separating must-have features from nice-to-have features
7. Designing the first usable version of an application
8. Avoiding overengineering in early product stages
9. Planning future improvements after MVP release
10. MVP validation and user feedback
11. Technical decisions that support MVP delivery
12. Balancing speed, quality, security, and maintainability

3. Kotlin Project Structure and Code Organization

1. Standard Kotlin project structure
2. Organizing packages and files clearly
3. Feature-based code organization
4. Layer-based code organization
5. Separating UI, business logic, and data handling
6. Managing shared utilities and common code
7. Understanding Gradle project structure
8. Introduction to modular application design
9. Benefits of modularization in larger applications
10. Keeping MVP code clean enough for future growth

4. Application Architecture Fundamentals

1. Why architecture matters in Kotlin applications
2. Common architecture problems in growing apps
3. Separation of concerns
4. Clean architecture concepts
5. Presentation layer
6. Domain layer
7. Data layer
8. Repository pattern
9. Use case pattern

10. Dependency management
11. Maintainable architecture for MVP and post-MVP products

5. MVVM Pattern

1. Purpose of MVVM
2. Model, View, and ViewModel responsibilities
3. Data flow in MVVM
4. UI state management
5. Business logic placement
6. Repository integration with MVVM
7. Handling loading, success, and error states
8. Managing user actions
9. Common MVVM mistakes
10. MVVM for scalable Android applications
11. MVVM in MVP-focused development

6. ViewModel Best Practices

1. Purpose of ViewModel in Android
2. Lifecycle-aware data handling
3. Surviving configuration changes
4. Avoiding memory leaks
5. Using `SavedStateHandle`
6. Managing coroutine scopes in ViewModels
7. Handling errors and retries
8. Sharing data between screens
9. Testing ViewModels
10. Structuring ViewModels for simple and complex screens

7. Reactive and Asynchronous Programming

1. Need for asynchronous programming in modern apps
2. Kotlin coroutines overview
3. Suspend functions
4. Coroutine scopes
5. Dispatchers

6. Structured concurrency
7. Flow basics
8. StateFlow basics
9. LiveData compared with Flow and StateFlow
10. Handling background tasks safely
11. Avoiding UI thread blocking
12. Asynchronous patterns for MVP applications

8. Security Fundamentals for Kotlin Applications

1. Basic principles of application security
2. Secure coding practices
3. Common mobile security risks
4. Authentication concepts
5. Token-based authentication
6. OAuth overview
7. Authorization concepts
8. Role-based access control
9. Secure data storage
10. Encryption basics
11. Key management basics
12. Network security
13. HTTPS enforcement
14. Certificate pinning overview
15. Code shrinking and obfuscation
16. Security considerations for MVP release

9. Performance Tuning and Optimization

1. Importance of performance in user experience
2. Memory management basics
3. Garbage collection overview
4. Detecting memory leaks
5. CPU performance concerns
6. Network performance concerns
7. Reducing application size

8. Resource shrinking
9. Code minification
10. Startup time improvement
11. Smooth UI rendering
12. Profiling tools
13. Production performance monitoring
14. Performance trade-offs during MVP development

10. Android 12 to Android 16 Migration Awareness

1. Overview of Android platform evolution
2. Target SDK considerations
3. Privacy and permission changes
4. Background execution changes
5. Security-related platform changes
6. Behavior changes affecting existing applications
7. Large-screen and resizability changes
8. Compatibility testing
9. Backward compatibility planning
10. Refactoring older Android code
11. Testing across multiple Android versions
12. Release readiness for modern Android devices

11. AI Integration in Kotlin and Android Applications

- Role of AI in modern applications
- AI as a product feature
- Choosing AI features for an MVP
- On-device AI concepts
- Cloud-based AI concepts
- API-based AI integration
- Handling AI request and response flows
- Latency and reliability concerns
- Data privacy considerations
- User consent and responsible AI usage
- AI feature testing

- AI feature limitations
- Planning AI features for future product versions

12. Backend Services for Kotlin Developers

- Why mobile developers need backend awareness
- Client-server application structure
- REST API concepts
- Request and response handling
- API error handling
- Authentication with backend services
- Database integration overview
- Logging and monitoring basics
- Backend validation
- Backend responsibilities in an MVP
- Kotlin for backend development
- Java interoperability
- Common backend frameworks and ecosystem awareness

13. Microservices and Distributed Systems Basics

- Meaning of monolithic architecture
- Meaning of microservices architecture
- When microservices are useful
- When microservices are unnecessary
- Service decomposition
- API gateway concepts
- Service-to-service communication
- REST communication
- gRPC overview
- Fault tolerance basics
- Retry policies
- Circuit breaker concepts
- Scaling basics
- Cloud deployment overview
- MVP-first thinking before moving to complex architecture

14. Event Streaming and Real-Time Systems

- Introduction to event-driven architecture
- Difference between request-response and event-driven communication
- Real-time data flow concepts
- Apache Kafka overview
- Topics and partitions
- Producers and consumers
- Event processing basics
- Data consistency considerations
- Ordering considerations
- Connecting mobile applications to real-time backends
- Real-time notifications and updates
- When event streaming is useful
- When event streaming is too complex for an MVP

15. Connecting Mobile Apps with Cloud Systems

- Mobile applications as cloud-connected clients
- API consumption patterns
- Offline-first concepts
- Data synchronization basics
- Conflict handling overview
- Logging across mobile and backend systems
- Metrics and monitoring
- Tracing basics
- Crash reporting
- Release monitoring
- Product feedback loops
- Improving the product after MVP launch

16. Product Readiness and Industry Delivery Practices

- Preparing an MVP for release
- Basic quality checks
- Functional testing
- Security checks

- Performance checks
 - Compatibility checks
 - User feedback collection
 - Issue prioritization
 - Technical debt awareness
 - Planning post-MVP improvements
 - Moving from MVP to scalable product
 - Industry expectations for production-ready Kotlin applications
-

Special Note

Some topics, especially those related to AI services, backend services, cloud platforms, package repositories, and event streaming platforms, may require internet access and firewall bypass during training, depending on the training environment and tools selected by the instructor.

Disclaimer

This course outline is intended to serve as a general training guideline. The trainer reserves the right to amend, reorganize, expand, reduce, or substitute any topic, sub-topic, tool, sequence, or delivery approach as deemed appropriate, based on participant background, class progress, technical constraints, industry updates, or training objectives, without prior notice.