Mobile Computing - Katze (Werwolf-Spiel)

Technische Grundlagen

Das Projekt ist eine Flutter-basierte mobile Anwendung des klassischen Werwolf-Spiels. Wir nutzen Flutter SDK 3.4.0, um eine einzige Codebasis für iOS und Android zu verwenden. Die Echtzeit-Kommunikation erfolgt über WebSockets, wodurch wir Spielzustände sofort synchronisieren können. Für Benachrichtigungen setzen wir auf Flutter Local Notifications.

Architektur

Die Anwendung verwendet Clean Architecture mit drei Schichten. Die Präsentationsschicht enthält die UI-Komponenten, die Kernlogik verwaltet die Spielregeln, und die Service-Schicht kümmert sich um Kommunikation und Authentifizierung. Das State Management erfolgt über das Provider Pattern, das die Spielzustände zentral verwaltet.

Spielmechanik

In unserer Version sind die Werwölfe durch Katzen ersetzt. Das Spiel läuft in Tag- und Nachtphasen ab. Tagsüber diskutieren alle Spieler im öffentlichen Chat, nachts agieren die Katzen heimlich. Die Kommunikation erfolgt über ein Chat-System mit separaten Räumen für Tag- und Nachtphase.

Mobile Features

Die App nutzt Deep Linking für Spieleinladungen und Push-Benachrichtigungen für wichtige Ereignisse. Bei Verbindungsabbrüchen werden Daten lokal zwischengespeichert und später synchronisiert. Die WebSocket-Verbindung ist für minimalen Batterieverbrauch optimiert.

Sicherheit

Die Sicherheit basiert auf JWT-Authentifizierung und verschlüsselter Kommunikation. Alle Spielaktionen werden serverseitig validiert. Die Datenspeicherung erfolgt DSGVO-konform, wobei nur notwendige Spielinformationen gespeichert werden.

Herausforderungen

Die größten Herausforderungen waren die Netzwerkstabilität und der Batterieverbrauch. Wir lösten dies durch automatische Wiederverbindung und optimierte Update-Intervalle. Das UI wurde speziell für mobile Geräte angepasst, mit besonderem Fokus auf verschiedene Bildschirmgrößen.

Technische Implementierung

Der WebSocket-Service bildet das Herzstück der Echtzeit-Kommunikation. Er verarbeitet typisierte Nachrichten für verschiedene Spielevents und implementiert einen Heartbeat-Mechanismus zur Verbindungsüberwachung. Die Spiellogik nutzt eine Zustandsmaschine für Phasenübergänge und validiert alle Aktionen. Das Provider Pattern ermöglicht effizientes State Management mit automatischer UI-Aktualisierung.

Architekturvorteile für Mobile

Die gewählte Architektur bietet spezifische Vorteile für mobile Anwendungen. Die Trennung der Schichten ermöglicht effizientes Caching und Offline-Funktionalität. Das Provider Pattern minimiert UI-Updates und schont damit den Akku. Die Service-Schicht abstrahiert plattformspezifische Implementierungen, was die Wartbarkeit erhöht.

Erkenntnisse Mobile Computing

Die Entwicklung zeigt die Besonderheiten mobiler Anwendungen: Energieeffizienz und Netzwerkstabilität sind kritisch. Cross-Platform-Entwicklung mit Flutter reduziert den Entwicklungsaufwand erheblich. Mobile-spezifische Features wie Push-Benachrichtigungen und Deep Linking sind entscheidend für die Benutzerinteraktion.