MATCHDAY

Flutter Developer Onboarding

Server URLs | Test Accounts | Firebase | OAuth | WebSocket | Deep Links

This document supplements the Flutter Developer Guide
Hand both documents to the Flutter developer on Day 1

1. Server URLs (P0 - Day 1)

WARNING: Fill in the highlighted TBD fields before sending to Flutter dev

1.1 API Base URLs

Environment	Base URL	Status
Local (Android Emulator)	http://10.0.2.2:8000/api/v1	Ready
Local (iOS Simulator)	http://localhost:8000/api/v1	Ready
Local (Physical Device)	http://{YOUR_IP}:8000/api/v1	Use ipconfig/ifconfig
Staging	https://__________.com/api/v1	TBD - Fill before handoff
Production	https://__________.com/api/v1	TBD - Fill before launch

1.2 WebSocket (Reverb) URLs

Environment	Host	Port	TLS	Auth Endpoint
Local	10.0.2.2	8080	false	{baseUrl}/broadcasting/auth
Staging	TBD	443	true	{baseUrl}/broadcasting/auth
Production	TBD	443	true	{baseUrl}/broadcasting/auth

1.3 Image Storage URLs

Environment	Storage Base URL
Local	http://10.0.2.2:8000/storage/
Staging	https://__________.com/storage/
Production	https://__________.com/storage/

TIP: All image URLs returned by the API are absolute. Flutter just uses the URL as-is.

1.4 Running Backend Locally

```
# Clone and setup
git clone {repo-url}
cd matchday-api
composer install
npm install && npm run build
cp .env.example .env
php artisan key:generate

# Database
php artisan migrate
php artisan db:seed
php artisan storage:link

# Start servers
php artisan serve # API on :8000
php artisan reverb:start # WebSocket on :8080 (optional)
php artisan queue:work # Queue worker (optional)
```

2. Test Accounts (P0 - Day 1)

WARNING: These accounts are created by php artisan db:seed

2.1 Account Credentials

Role	Email	Password	Use For
Fan (User)	fan@matchday.app	password	Testing fan app: home, bookings, chat, loyalty
Cafe Owner	owner@matchday.app	password	Testing cafe admin: branches, matches, QR scan
Staff (Manager)	staff@matchday.app	password	Testing staff: bookings, check-in, QR scan
Platform Admin	admin@matchday.app	password	Web dashboard only (/platform)

2.2 Quick Login Test

```
POST /api/v1/auth/login
Content-Type: application/json

// Fan login
{ "email_or_phone": "fan@matchday.app", "password": "password" }

// Cafe Owner login
{ "email_or_phone": "owner@matchday.app", "password": "password" }
```

2.3 What the Seeder Creates

Data	Count	Notes
Fan users	3-5	With profiles, loyalty cards, favorite teams
Cafe owners	2-3	With cafes, branches, sections, seats
Teams	20+	Premier League + Arab teams (Al Hilal, Al Nassr, etc.)
Matches	10+	Mix of upcoming, live, completed
Bookings	5-10	Various statuses with QR codes
Subscription plans	3	Starter, Professional, Enterprise
Achievements	8+	Various categories
FAQs	8	Common questions
Legal pages	4	Privacy, terms, data usage, cookie policy

2.4 Role-Based Access

```
fan@matchday.app:
Access: /auth/*, /profile/*, /cafes/*, /matches/*,
/bookings/*, /payments/*, /chat/*, /notifications/*,
/home, /explore, /search, /loyalty/*, /achievements/*

owner@matchday.app:
Access: All fan endpoints + /cafe-admin/* (all 85 endpoints)
Has all permissions by default
```

```
staff@matchday.app:
Access: /cafe-admin/* (limited by permissions)
Default: view-bookings, scan-qr, check-in-customers

admin@matchday.app:
Access: Web dashboard /platform/* only
Not for mobile API testing
```

3. Firebase / FCM Setup (P1)

3.1 What the Flutter Dev Needs

File/Config	Platform	Where to Get	Status
google-services.json	Android	Firebase Console > Project Settings > Android app	TBD
GoogleService-Info.plist	iOS	Firebase Console > Project Settings > iOS app	TBD
Firebase Project ID	Both	Firebase Console > Project Settings > General	TBD
FCM Server Key	Backend	Firebase Console > Cloud Messaging > Server Key	TBD

3.2 Backend Config

```
# .env (backend needs this to SEND push notifications)
FCM_SERVER_KEY=AAAA...your_server_key...

# The Flutter app needs:
# 1. google-services.json in android/app/
# 2. GoogleService-Info.plist in ios/Runner/
# 3. firebase_messaging package in pubspec.yaml
```

3.3 Action Items

Backend team must:

* Create Firebase project (or share existing)
* Register Android app (package name: com.matchday.app or TBD)
* Register iOS app (bundle ID: com.matchday.app or TBD)
* Download and share both config files
* Set FCM_SERVER_KEY in backend .env

Flutter dev must:

* Place google-services.json in android/app/
* Place GoogleService-Info.plist in ios/Runner/
* Call PUT /profile/device-token after login
* Handle FCM onMessage, onBackgroundMessage, onMessageOpenedApp

4. Google and Apple OAuth (P2)

4.1 Google OAuth

Config	Value	Status
Web Client ID	TBD (from Google Cloud Console)	TBD
Android Client ID	Auto from google-services.json	Needs Firebase setup
iOS Client ID	Auto from GoogleService-Info.plist	Needs Firebase setup
Backend verifies via	Google API token verification	Code ready

4.2 Google Setup Steps

```
Backend:
1. Go to Google Cloud Console > APIs & Services > Credentials
2. Create OAuth 2.0 Client ID (Web application type)
3. Set GOOGLE_CLIENT_ID and GOOGLE_CLIENT_SECRET in .env

Flutter:
1. Use google_sign_in package
2. Get idToken from Google Sign In
3. POST /auth/login/google { google_token: idToken }
```

4.3 Apple Sign In

Config	Value	Status
Apple Team ID	TBD (Apple Developer Account)	TBD
Service ID	TBD	TBD
Key ID	TBD	TBD
Private Key (.p8)	TBD	TBD

4.4 Apple Setup Steps

```
Backend:
1. Apple Developer > Certificates, Identifiers & Profiles
2. Register App ID with Sign In with Apple capability
3. Create Service ID and Key
4. Set APPLE_CLIENT_ID, APPLE_TEAM_ID, APPLE_KEY_ID in .env
5. Place .p8 key file in storage/

Flutter:
1. Use sign_in_with_apple package
2. Get identityToken from Apple
3. POST /auth/login/apple { apple_token: identityToken, name: '...' }
```

WARNING: Apple Sign In only works on real iOS devices, not simulator

TIP: For testing, skip social auth and use email/password login

5. Deep Links and URL Scheme (P3)

5.1 Custom URL Scheme

```
Scheme: matchday://

Examples:
matchday://home
matchday://bookings/123
matchday://matches/456
matchday://cafes/789
matchday://loyalty
matchday://achievements
matchday://chat/rooms/12
matchday://invite/{token}
```

5.2 Push Notification Deep Links

Notification Type	Deep Link	Flutter Route
booking_confirmed	matchday://bookings/{booking_id}	/booking-detail
booking_cancelled	matchday://bookings/{booking_id}	/booking-detail
match_reminder	matchday://matches/{match_id}	/match-detail
match_score_update	matchday://matches/{match_id}	/match-detail
achievement_unlocked	matchday://achievements	/achievements
points_earned	matchday://loyalty	/loyalty
welcome	matchday://home	/home
staff_invitation	matchday://invite/{token}	/accept-invite

5.3 Universal Links (Optional - Production)

```
If using universal links instead of custom scheme:

Domain: https://matchday.app

Android: assetlinks.json at /.well-known/
iOS: apple-app-site-association at /.well-known/

Examples:
https://matchday.app/booking/123
https://matchday.app/match/456
```

TIP: Start with custom scheme (matchday://). Add universal links later for production.

6. Flutter Environment Config Template

```
// lib/config/environment.dart

enum Environment { local, staging, production }

class AppConfig {
static Environment env = Environment.local;

static String get apiBaseUrl {
switch (env) {
case Environment.local:
return 'http://10.0.2.2:8000/api/v1';
case Environment.staging:
return 'https://__________.com/api/v1'; // TBD
case Environment.production:
return 'https://__________.com/api/v1'; // TBD
}
}

static String get wsHost {
switch (env) {
case Environment.local:
return '10.0.2.2';
case Environment.staging:
return '__________'; // TBD
case Environment.production:
return '__________'; // TBD
}
}

static int get wsPort =>
env == Environment.local ? 8080 : 443;

static bool get wsTLS =>
env != Environment.local;

static String get storageUrl {
switch (env) {
case Environment.local:
return 'http://10.0.2.2:8000/storage';
case Environment.staging:
return 'https://__________.com/storage'; // TBD
case Environment.production:
return 'https://__________.com/storage'; // TBD
}
}
}
```

7. Recommended Flutter Packages

Package	Purpose	Required?
http or dio	API calls	Yes - pick one
flutter_secure_storage	Token storage	Yes
pusher_channels_flutter	WebSocket (Reverb)	Yes - for chat
mobile_scanner	QR code scanning (staff)	Yes - for cafe admin
cached_network_image	Image caching + loading	Yes
firebase_messaging	Push notifications	Yes
firebase_core	Firebase initialization	Yes
google_sign_in	Google OAuth	Yes - for social login
sign_in_with_apple	Apple Sign In	Yes - for iOS
image_picker	Avatar/image upload	Yes
provider or riverpod	State management	Recommended
go_router	Navigation + deep links	Recommended
intl	Date formatting + Arabic	Recommended
shimmer	Loading placeholders	Optional
pull_to_refresh	Pull to refresh lists	Optional

8. Handoff Checklist

Before Handing to Flutter Dev

#	Task	Priority	Done?
1	Backend running + accessible (local or staging URL)	P0	
2	php artisan db:seed completed (test data exists)	P0	
3	Test accounts verified (fan, owner, staff, admin)	P0	
4	Flutter Developer Guide PDF delivered	P0	
5	This Onboarding Sheet delivered (with TBDs filled)	P0	
6	Postman collection + environment delivered	P0	
7	Firebase project created + files shared	P1	
8	FCM_SERVER_KEY set in backend .env	P1	
9	Reverb running (for chat testing)	P1	
10	Google OAuth Client ID shared	P2	
11	Apple Sign In configured	P2	
12	Deep link scheme agreed (matchday://)	P3	
13	App package name / bundle ID decided	P3	

What Flutter Dev Gets

* **MatchDay_Flutter_Developer_Guide.pdf -** Full API reference (15 sections)
* **MatchDay_Flutter_Onboarding.pdf -** This document (URLs, accounts, setup)
* **Postman collection + environment -** In docs/ folder
* **google-services.json** - When Firebase is ready
* **GoogleService-Info.plist** - When Firebase is ready

MATCHDAY

Laravel 12 | ~263 Endpoints | ~43 Tables | 367 Tests

February 2026 | Version 1.0