MACHAD

Android Application and Railway Deployment

Step-by-step implementation, security, testing, database and maintenance guide

PROJECT RECORD

Item	Result
Application	Machad institution management system
Production URL	machad-production.up.railway.app/login
Android release	Version 1.1.0 - version code 2
Git commit	ef73e6a - Publish signed Machad Android app
Deployment	GitHub main branch to Railway production
Prepared	16 August 2026

This document records the completed implementation and provides a repeatable operating procedure for future releases.

1. Purpose and completed outcome

The goal was to make the Laravel and TypeScript web system available as a downloadable Android application while keeping the web application as the single source of business logic and data.

Completed outcome

- A public Android download card is displayed on the login page.
- The APK can be downloaded without authentication.
- The Android application opens the Railway production system in a secure WebView.
- The release is signed, versioned and non-debuggable.
- The installed Android app does not display a button to download itself again.
- The published APK was verified against the local signed release.
- GitHub and Railway deployment completed successfully.

Document map

Item	Result
Sections 2-4	Architecture, prerequisites and public APK download implementation
Sections 5-7	Android wrapper, release signing, security and build process
Sections 8-10	GitHub, Railway, MySQL and TablePlus procedures
Sections 11-13	Testing evidence, troubleshooting and future maintenance

Security scope

Passwords, database secrets and release signing passwords are intentionally excluded. Store them only in Railway secret variables or an approved password manager. Never commit them to Git.

2. Solution architecture

Machad uses a thin Android wrapper. Teachers and administrators use the same Laravel application and the same Railway MySQL database whether they open the system in a browser or in the Android app.

Laravel + TypeScript
Railway production web application
Secure Android WebView
Release-signed APK 1.1.0
Teachers, administrators and other authorized users

How data works

1. A user opens the Machad Android application.
2. The WebView loads the HTTPS Railway production URL.
3. The user logs in using the same account used by the web application.
4. Laravel validates the session and permissions.
5. All changes are stored in the same Railway MySQL database.
6. The same data is immediately available to authorized browser and Android users.

The APK does not contain a separate copy of the school database. It is a secure client for the live web system.

3. Prerequisites

- Laravel application with the TypeScript/Inertia login page.
- Git repository connected to GitHub.
- Railway project with an application service and MySQL service.
- Android SDK, build tools, Java runtime and Gradle.
- A secure release keystore stored outside Git.
- A physical Android phone or Android emulator for testing.

4. Public APK download implementation

Step 1 - Configure the Android artifact

The Android configuration stores whether downloading is enabled, the displayed version, the server-side filename and the public download filename.

```
config/android.php

'enabled' => env('ANDROID_APP_ENABLED', true),
'version' => env('ANDROID_APP_VERSION', '1.1.0'),
'file' => env('ANDROID_APP_FILE', 'system-app.apk'),
'download_name' => env('ANDROID_APP_DOWNLOAD_NAME', 'machad-android.apk'),
```

Step 2 - Add the public download route

The route is outside the authentication middleware so anyone on the login page can download the installer.

```
GET /download/android -> AndroidAppDownloadController
```

Step 3 - Validate and serve the file

- Reject unavailable or disabled downloads with HTTP 404.
- Restrict the configured filename to the APK directory.
- Send Content-Type: application/vnd.android.package-archive.
- Send Content-Disposition as an attachment named machad-android.apk.
- Send X-Content-Type-Options: nosniff.

Step 4 - Display the card on the login page

The login page displays the application title, installation description, download button, version and formatted file size. If the file is unavailable, the interface explains that a new Android build is required.

5. Build the Android WebView application

Step 1 - Create the Android project

The Android source is stored in android-app/. The final package is com.itqaanict.machad and the user-facing application name is Machad.

Step 2 - Set the production URL

```
DEFAULT_SERVER_URL = "https://machad-production.up.railway.app"
```

A fresh installation opens the live system immediately. The user does not need to type a server address.

Step 3 - Configure WebView

- Enable JavaScript and DOM storage for the Laravel interface.
- Accept first-party and required third-party cookies for sessions.
- Support file selection where the web system requires uploads.
- Keep internal Machad links inside the app.
- Open external links using an appropriate Android application.
- Add the user agent marker MachadAndroid/1.1.0.

Step 4 - Hide the redundant download card

Laravel checks the user agent. When MachadAndroid/ is present, the login controller returns androidApp as null. Browser visitors still see the download card; installed-app users do not.

Step 5 - Handle device layout

The Android root view applies system window insets so the WebView content does not overlap the status and navigation bars.

Why the APK is small

Machad is a pure Java WebView wrapper and uses the Android system WebView already installed on the phone. It does not bundle a browser engine, Flutter engine, CPU-native libraries, fonts or media assets. Therefore a small package size is expected.

6. Release signing and versioning

Step 1 - Create the release keystore

A dedicated 4096-bit RSA release key was created for ITQaan ICT. The keystore is the permanent identity of future Machad Android updates.

Step 2 - Protect signing credentials

- Keep the .jks keystore outside version control.
- Keep release-signing.properties outside version control.
- The Android .gitignore excludes *.jks, *.keystore and release-signing.properties.
- Back up the keystore and passwords in at least two secure locations.
- Never generate a replacement key for a routine update. Android will treat it as a different signer.

Step 3 - Set the release version

```
versionCode 2
versionName '1.1.0'
release.debuggable false
release.minifyEnabled false
```

Step 4 - Build the signed release

```
gradle assembleRelease
```

The generated app-release.apk is copied to storage/app/apk/system-app.apk, which is the server-side source for the public download.

7. Android security hardening

- Only the INTERNET permission is retained.
- The unused POST_NOTIFICATIONS permission was removed.
- Cleartext HTTP traffic is disabled.
- The production wrapper uses HTTPS.
- The release is non-debuggable.
- File access from WebView is disabled; controlled content access remains available.

8. GitHub and Railway deployment

Step 1 - Run local checks

1. Build the Android release.
2. Verify APK metadata and signing schemes.
3. Run the authentication and Android download feature tests.
4. Run the TypeScript and Vite production build.
5. Run git diff --check and confirm secrets are ignored.

Step 2 - Commit and push

```
git add <approved files>
git commit -m "Publish signed Machad Android app"
git push origin main
```

Production commit: ef73e6a

Step 3 - Railway automatic build

Railway detected the GitHub push, built the Laravel/TypeScript application and published a new production deployment.

Step 4 - Set Railway Android variables

Item	Result
ANDROID_APP_ENABLED	true
ANDROID_APP_VERSION	1.1.0
ANDROID_APP_FILE	system-app.apk

Step 5 - Apply variable changes

After editing Railway variables, click Deploy so the new environment revision becomes active. Confirm that the service returns to Online and the deployment is marked successful.

Step 6 - Confirm production

Open the production login page and verify that the card shows version 1.1.0 and the download link points to /download/android.

9. Railway MySQL and TablePlus

Step 1 - Enable public TCP networking

In the Railway MySQL service, open Settings and locate Public Networking. Railway provides an external proxy hostname and port for desktop database tools.

Step 2 - Create the TablePlus connection

Item	Result
Name	Machad Railway MySQL
Host/IP	Current Railway public proxy host
Port	Current Railway public proxy port
User	Value from MYSQLUSER
Password	Value from MYSQLPASSWORD - never document or share it
Database	Value from MYSQLDATABASE
SSL mode	Preferred, unless Railway requires a stricter mode
SSH	Disabled for the Railway public proxy connection

Important

The internal hostname mysql.railway.internal works only between Railway services. TablePlus on a local computer must use the public proxy hostname and public port.

10. Prepare and import the existing database

1. Create a Railway database backup before importing.
2. Inspect the SQL file and confirm it targets MySQL.
3. Compare the backup schema with the deployed Laravel migrations.
4. Create a compatibility copy of the SQL file if migration differences exist.
5. Open the Railway connection in TablePlus.
6. Use the restore/import function and select the prepared SQL file.
7. Monitor errors and stop if destructive statements target the wrong database.
8. After import, clear stale session/cache data and run the required Laravel migrations.
9. Test login, users, students, payments and reports.

Never import directly over production without a recent backup and a confirmed rollback plan.

11. Developer account procedure

A developer account should be created only when operationally required and should receive the minimum permissions necessary.

1. Generate the password hash using Laravel's Hash facade or an approved application command.
2. Use a parameterized application workflow when available. If SQL is required, use placeholders and never store the plaintext password in documentation.
3. Insert or update the user record using the schema's actual role and status columns.
4. Confirm that the account can log in and that authorization rules restrict it appropriately.
5. Rotate the password if it was exposed in screenshots, chat or shell history.

Illustrative pattern only

```
INSERT INTO users (name, email, password, created_at, updated_at)
VALUES ('Platform Developer', :email, :bcrypt_hash, NOW(), NOW());
```

12. Verification and evidence

Item	Result
APK version	1.1.0, version code 2
Minimum Android	API 23
Target Android	API 36
Signing	RSA 4096-bit; APK Signature Scheme v1 and v2 verified
Debug state	Release; non-debuggable
HTTP status	200 OK
Content type	application/vnd.android.package-archive
Download name	machad-android.apk
Published size	16,322 bytes (15.9 KB)
SHA-256	5CE31C440FA0C309896AF0750F7403E0B95A3CA820CBD77772DE46511BFB7B76
Byte comparison	Downloaded production APK exactly matched the verified local release
Installation	Exact downloaded APK installed successfully on Android emulator

13. Horjeed comparison and troubleshooting

Why Horjeed is 45.7 MB

The inspected Horjeed APK is a Flutter application. It bundles Flutter and Dart native libraries for arm64-v8a, armeabi-v7a and x86_64, plus application assets, fonts, images, camera/media support and other framework resources.

Item	Result
Horjeed	47,924,001 bytes; Flutter; three CPU architectures; Android debug certificate
Machad	16,322 bytes; native Java WebView; no bundled native engine; dedicated release certificate

A larger APK is not automatically safer or more installable. The Machad package passed signature verification and Android's package manager accepted the exact downloaded file.

Common direct-download warnings

- Chrome may say that an APK could be harmful because APKs can install software.
- Google Play Protect may say it has not seen the developer before.
- These warnings are common for apps distributed outside Google Play.
- Do not instruct users to disable Play Protect globally.
- Users should install only the APK downloaded from the official Machad production domain.

If Android says App not installed

1. Uninstall any older Machad copy, including copies in work or secondary profiles.
2. Restart the phone and download the current APK again.
3. Confirm Chrome or the file manager is allowed to install unknown applications.
4. If it still fails, enable USB debugging and run adb install against the physical device.
5. Use the returned INSTALL_FAILED_* code to identify signature, storage, policy or compatibility errors.

14. Future release checklist

1. Make and test the required Laravel or Android changes.
2. Increase versionCode for every Android release.
3. Update versionName and the MachadAndroid user-agent marker.
4. Build with the original release keystore.
5. Verify the signature and confirm the APK is non-debuggable.
6. Install on an emulator and at least one physical Android phone.
7. Copy the verified APK to storage/app/apk/system-app.apk.
8. Update the default version in config/android.php.
9. Commit and push only approved source files and the APK.
10. Update ANDROID_APP_VERSION in Railway and deploy the variable change.
11. Download the production APK and compare its SHA-256 hash with the local release.
12. Record the release date, commit, version, hash and test results.

Critical operational rules

- Back up the release keystore. Losing it can prevent seamless updates.
- Never commit secrets, database passwords or signing passwords.
- Back up MySQL before imports, migrations or destructive maintenance.
- Rotate any credential that appears in screenshots or shared messages.
- Keep Railway and GitHub access limited to authorized maintainers.

Final status

The Machad web system, public Android download, release-signed Android wrapper, GitHub deployment and Railway production environment were successfully configured and verified. The application is ready for controlled direct distribution and future signed updates.