

[www.reallygreatsite.com](http://www.reallygreatsite.com)

# Mobile Application Security Analysis





# Phase 1: Environment Setup

01

```
openjdk version "23.0.2" 2025-01-21
ajzankulkibaeva@MacBook-Air-Ajzan ~ % brew install android-platform-tools
```


02

```
Android Debug Bridge version 1.0.41
Version 36.0.0-13206524
Installed as /opt/homebrew/bin/adb
Running on Darwin 23.6.0 (arm64)
ajzankulkibaeva@MacBook-Air-Ajzan ~ %
```

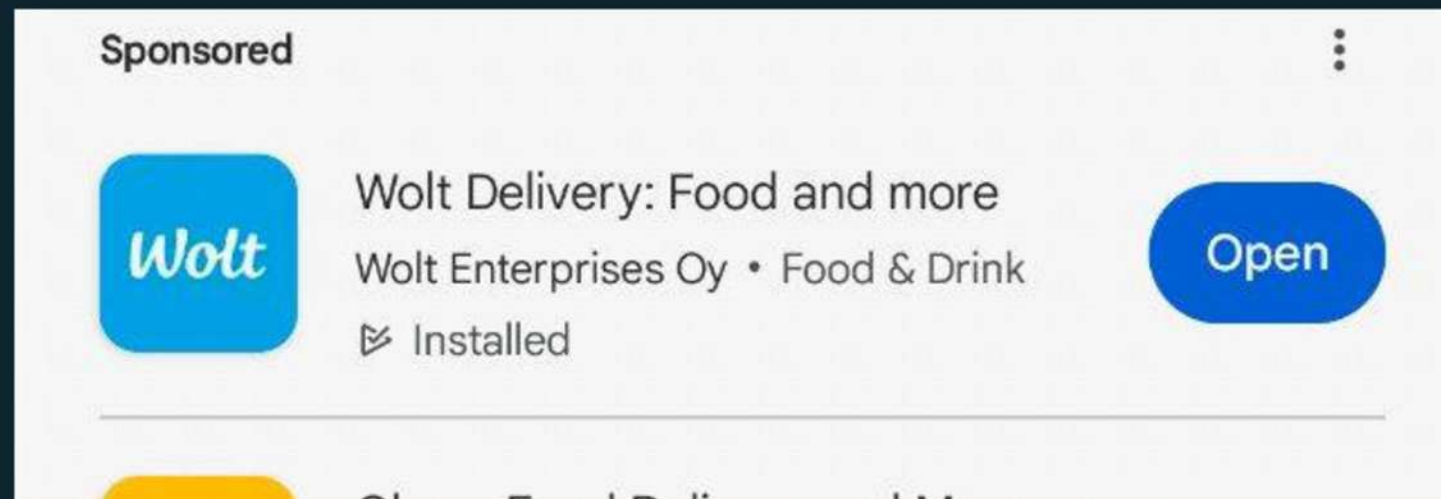
03

 /opt/homebrew/Cellar/apktool/2.11.1: 5 files, 23.5MB  
APKTool Installed

04

```
#####
==> Pouring jadx--1.5.1.all.bottle.tar.gz
 /opt/homebrew/Cellar/jadx/1.5.1: 12 files, 121.2MB
==> Running `brew cleanup jadx`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
rc Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
```

# Phase 2: Application Selection



Wolt

Next, I selected the Wolt application (a food delivery service) because it clearly relies on network API communication. I launched the AVD emulator, opened Google Play Store, signed in, installed Wolt, and confirmed the app was functioning properly



# Phase 3: APK Extraction

```
adb devices
```

List of devices attached  
emulator-5554 device

2. Identify the package name of your installed application:

```
adb shell pm list packages | grep <keyword>
```

```
adb shell pm list packages | grep wolt package:com.wolt.android
```

3. Locate the APK path on the device:

```
.adb shell pm path com.wolt.android
```

```
adb shell pm path <package.name>
```

```
package:/data/app/~~0_Oft10KOzysVmnAQbIUlw==/com.wolt.android-rXJxXkLQWZDece2CZTWxJA==/base.apk  
package:/data/app/~~0_Oft10KOzysVmnAQbIUlw==/com.wolt.android-rXJxXkLQWZDece2CZTWxJA==/split_config.arm64_v8a.apk  
package:/data/app/~~0_Oft10KOzysVmnAQbIUlw==/com.wolt.android-rXJxXkLQWZDece2CZTWxJA==/split_config.en.apk  
package:/data/app/~~0_Oft10KOzysVmnAQbIUlw==/com.wolt.android-rXJxXkLQWZDece2CZTWxJA==/split_config.xxhdpi.apk
```

4. Pull the APK from the device to your computer:

```
adb pull /data/app/~~0_Oft10KOzysVmnAQbIUlw==/com.wolt.android-rXJxXkLQWZDece2CZTWxJA==/base.apk ./extracted_wolt_base.apk
```

```
adb pull <path/to/base.apk> ./extracted_app.apk
```

```
/data/app/~~0_Oft10KOzysVmnAQbIUlw==/com.wol...ipped. 65.0 MB/s (136851690 bytes in 2.007s)
```

## **adb devices**

To check if your Android emulator or physical device is connected and recognized by ADB. This is a basic check to ensure that the device is ready for communication.

## **adb shell pm list packages | grep <keyword>**

To find the package name of the installed application by filtering with a keyword.

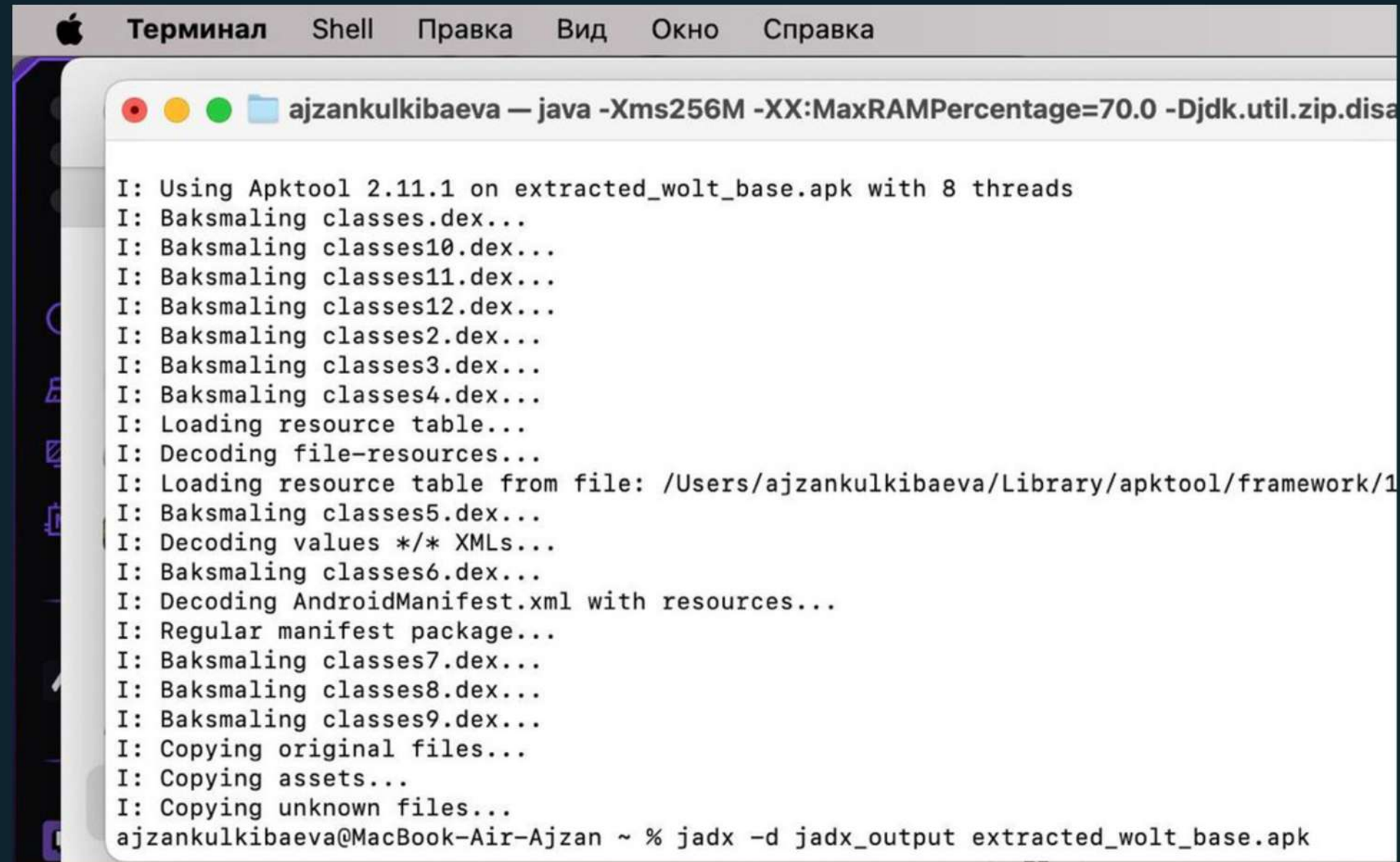
## **adb shell pm path <package.name>**

To locate the exact path to the APK file of the application on the device.

## **adb pull <path/to/base.apk> ./extracted\_app.apk**

To pull (download) the APK file from the device to your local computer for analysis.

# Phase 4: Decompiling the Application



```
Терминал  Shell  Правка  Вид  Окно  Справка
ajzankulkibaeva — java -Xms256M -XX:MaxRAMPercentage=70.0 -Djdk.util.zip.disa

I: Using Apktool 2.11.1 on extracted_wolt_base.apk with 8 threads
I: Baksmaling classes.dex...
I: Baksmaling classes10.dex...
I: Baksmaling classes11.dex...
I: Baksmaling classes12.dex...
I: Baksmaling classes2.dex...
I: Baksmaling classes3.dex...
I: Baksmaling classes4.dex...
I: Loading resource table...
I: Decoding file-resources...
I: Loading resource table from file: /Users/ajzankulkibaeva/Library/apktool/framework/1
I: Baksmaling classes5.dex...
I: Decoding values */* XMLs...
I: Baksmaling classes6.dex...
I: Decoding AndroidManifest.xml with resources...
I: Regular manifest package...
I: Baksmaling classes7.dex...
I: Baksmaling classes8.dex...
I: Baksmaling classes9.dex...
I: Copying original files...
I: Copying assets...
I: Copying unknown files...
ajzankulkibaeva@MacBook-Air-Ajzan ~ % jadx -d jadx_output extracted_wolt_base.apk
```



# Phase 4.5: SSL Certificate Pinning Bypass

```
zsh: no such file or directory: /Users/ajzankulkibaeva/Desktop/ssl-bypass.js
ajzankulkibaeva@MacBook-Air-Ajzan ~ % adb shell chmod 777 /data/local/tmp/frida-
server-16.7.14-android-arm64
ajzankulkibaeva@MacBook-Air-Ajzan ~ % adb root
```

Рабочий стол — zsh — 80x24

Mon May 12 18:28:43 on ttys000

```
ajzankulkibaeva@MacBook-Air-Ajzan ~ % adb root
ajzankulkibaeva@MacBook-Air-Ajzan ~ % adb shell chmod 777 /data/local/tmp/frida-
server-arm64 &
```

```
/Users/ajzankulkibaeva/Downloads/caace...ipped. 1.6 MB/s (1326 bytes in 0.001s)
ajzankulkibaeva@MacBook-Air-Ajzan Desktop % frida -U -f com.wolt.android -l /Us
rs/ajzankulkibaeva/Desktop/fridascript.js
```

```

  ____
 /  _ \
| (  ) |
 > _ <
/_/  \_\

Frida 16.7.14 - A world-class dynamic instrumentation toolkit

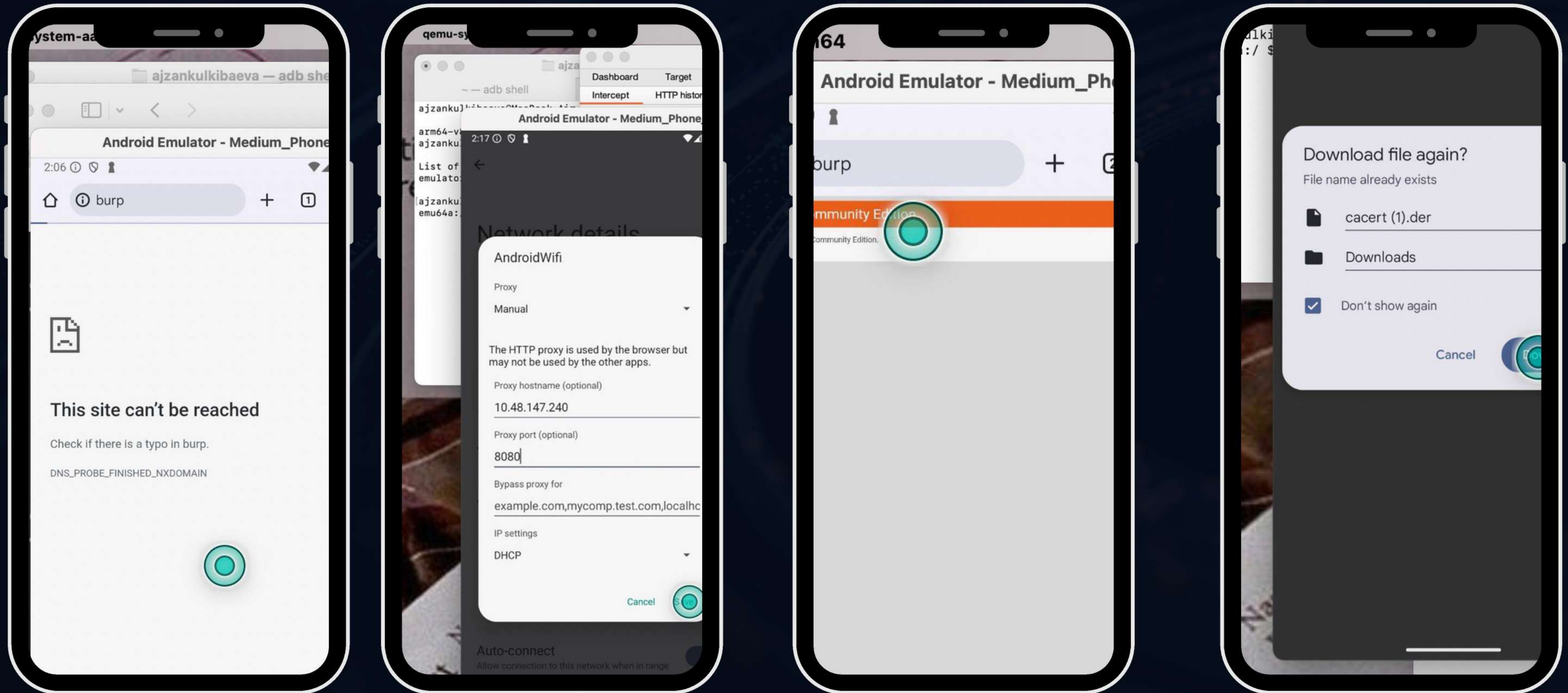
Commands:
  help      -> Displays the help system
  object?   -> Display information about 'object'
  exit/quit -> Exit

More info at https://frida.re/docs/home/

Connected to Android Emulator 5554 (id=emulator-5554)
Spawned `com.wolt.android`. Resuming main thread!
[Android Emulator 5554::com.wolt.android ]->
[.] Cert Pinning Bypass/Re-Pinning
[+] Loading our CA...
[o] Our CA Info: CN=PortSwigger CA, OU=PortSwigger CA, O=PortSwigger, L=PortSwi
ger, ST=PortSwigger, C=PortSwigger
[+] Creating a KeyStore for our CA
```



# Phase 5: Burp Suite Configuration





# Phase 6: Intercepting API Requests

**REQUESTMETHOD: GET**  
**ENDPOINT: /V1/PAGES/RESTAURANTS**  
**QUERY PARAMETERS:**

**LAT=43.245132**  
**LON=76.954158**

## **PURPOSE**

- **FETCH A LIST OF RESTAURANTS NEAR THE PROVIDED COORDINATES.**

# Phase 7: API Documentation

The **`https://authentication.wolt.com/v1/wauth2/access_token`** endpoint is used in the OAuth 2.0 authentication flow to obtain access and refresh tokens.

**Method: POST**

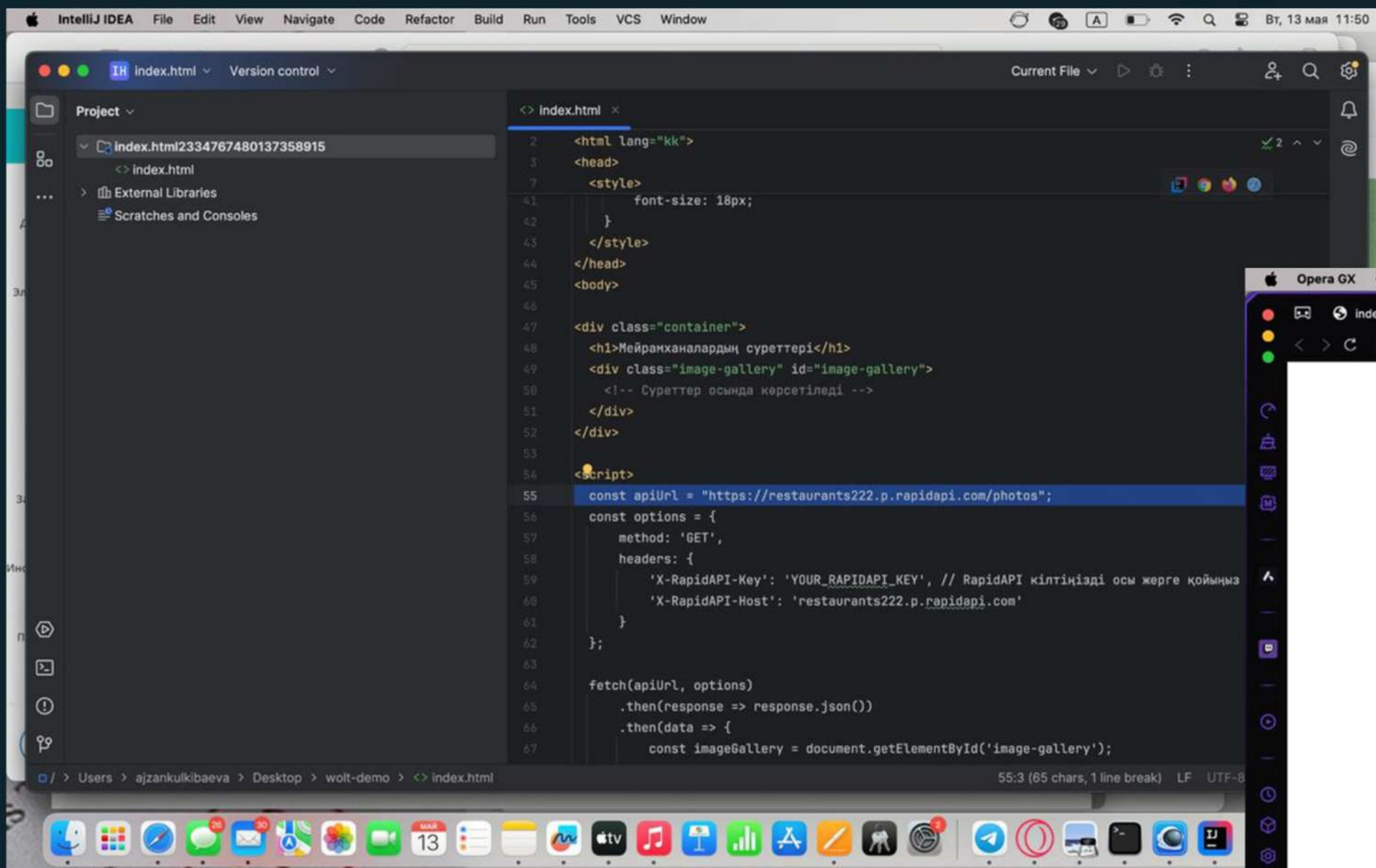
**Purpose:** Exchange an authorization code for an access token (for API requests) and a refresh token (to renew the access token).

Parameters: `grant_type` (authorization\_code), `code` (authorization code), `redirect_uri` (same as used during authorization).

**Response:** Access token, refresh token, and expiration time.



# Phase 8: Demo Application Development



```
<?xml version="1.0" encoding="UTF-8" ?>
<html lang="kk">
<head>
<style>
font-size: 18px;
</style>
</head>
<body>
<div class="container">
<h1>Мейрамханалардың суреттері</h1>
<div class="image-gallery" id="image-gallery">
<!-- Суреттер осында көрсетіледі -->
</div>
</div>
</body>
</html>
<script>
const apiUrl = "https://restaurants222.p.rapidapi.com/photos";
const options = {
method: 'GET',
headers: {
'X-RapidAPI-Key': 'YOUR_RAPIDAPI_KEY', // RapidAPI кілтіңізді осы жерге қойыңыз
'X-RapidAPI-Host': 'restaurants222.p.rapidapi.com'
}
};
fetch(apiUrl, options)
.then(response => response.json())
.then(data => {
const imageGallery = document.getElementById('image-gallery');
```

