

ISET de Radès

Département Technologies de l’Informatique



Mini projet : Contrôle d’accès avec reconnaissance faciale

Réalisé par Fedi Sabri de L1I5

Guidé par Mr Sami Melki

2023

# Matériel requis :

|  |  |  |
| --- | --- | --- |
| RPi3B | Ordinateur single board, système d’exploitation Raspbian et Connecté à l’internet |  |
| Smartphone Android (cam) | Pour camera |  |
| Câbles | Câbles de connexion |  |
| Câble USB vers Micro-USB | Identique aux câbles de du téléphone portable |  |
| Câble USB vers USB type-C | Identique aux câbles de charge du téléphone portable |  |
| Pc pour développement | Système d’exploitation Windows et Connecté à l’internet |  |
| Diodes led | Vert et rouge pour indiquer la réponse |  |
| Bottons |  |  |

# Table des matières

[I)Configurer la carte et l’environnement de développement 4](#_Toc136233338)

[1) Raspberry pi imager 4](#_Toc136233339)

[2) Connecter vs code à travers SSH 8](#_Toc136233340)

[3) Installation des bibliothèques et binaires nécessaires 8](#_Toc136233341)

[4) Installation de flutter et Dart 9](#_Toc136233342)

[5) Installation de Android studio et création de machine virtuelle 10](#_Toc136233343)

[II) Le schéma principal 14](#_Toc136233344)

[III) Le montage 14](#_Toc136233345)

[IV) Le code 15](#_Toc136233348)

[6) Le code de la carte 15](#_Toc136233349)

[7) Le code coté serveur 18](#_Toc136233350)

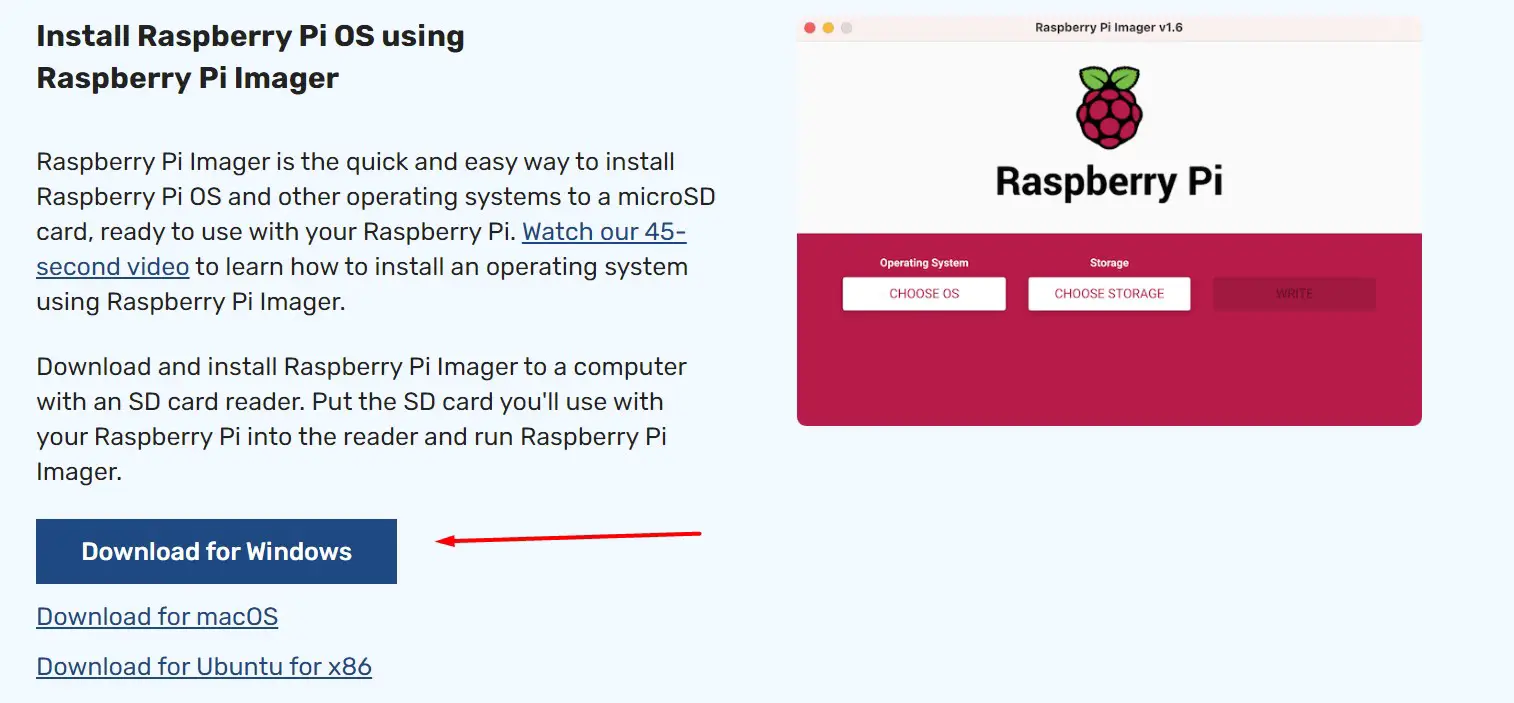
[8) le code de l’application client 22](#_Toc136233351)

[9) Capture d’écran de l’application 35](#_Toc136233352)

# Configurer la carte et l’environnement de développement

## Raspberry pi imager

Téléchargez et lancez Raspberry pi imager setup



Suivez l’installateur



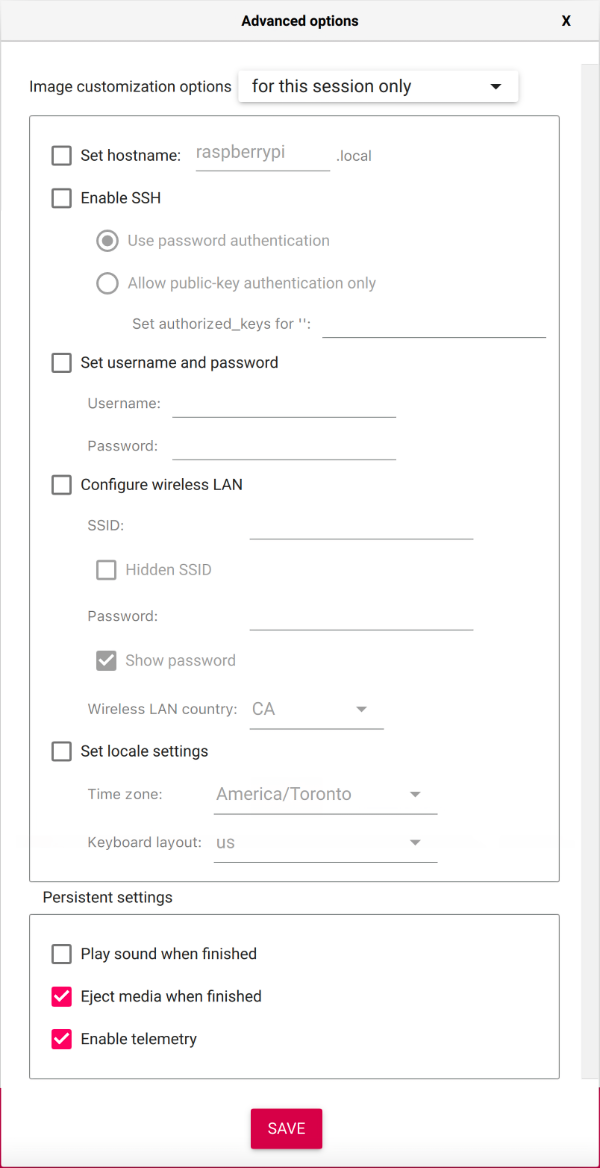
Une fois installé, lancez-le.



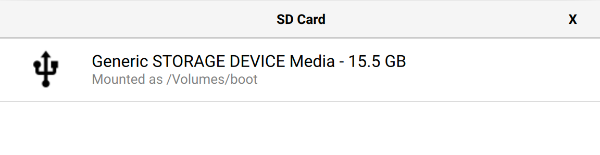
Cliquez sur « Choose OS »



Configurez les options avancées en appuyant sur les touches Ctrl + Maj + X pour faire apparaitre ce menu (SSH pour pouvoir utiliser la carte en mode headless)



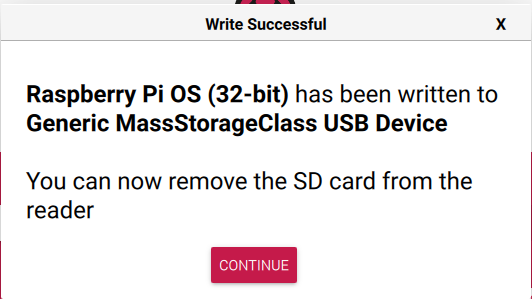
Cliquez sur « Choose SD card » puis sélectionnez votre carte micro SD



Cliquez sur « write »

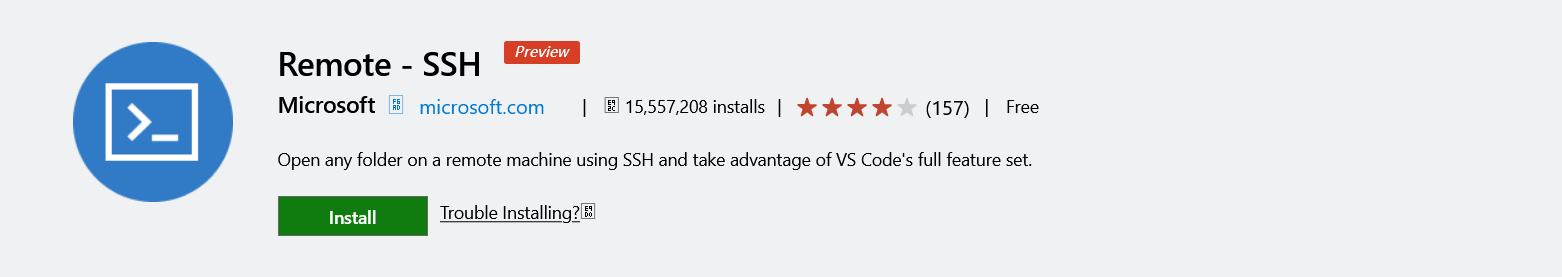


Cliquez sur « ok »

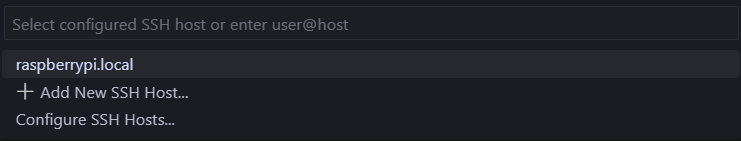


## Connecter vs code à travers SSH

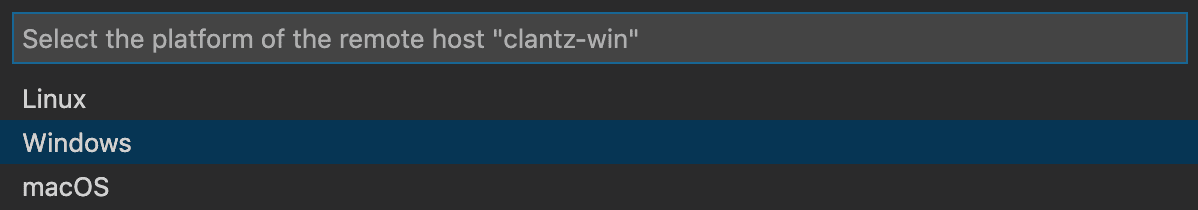
Installez l’extension « remote ssh » par Microsoft



Tapez “Remote-SSH: Connect to Host” dans la palette des commandes et écrivez l’adresse IP locale de votre carte puis entrez votre mot de passe



Sélectionnez le type du système si demandé



## Installation des bibliothèques et binaires nécessaires

* **Sur** serveur

py -m pip install deepface

py -m pip install sqlite3

py -m pip install flask

py -m pip install flask\_cors

winget install git

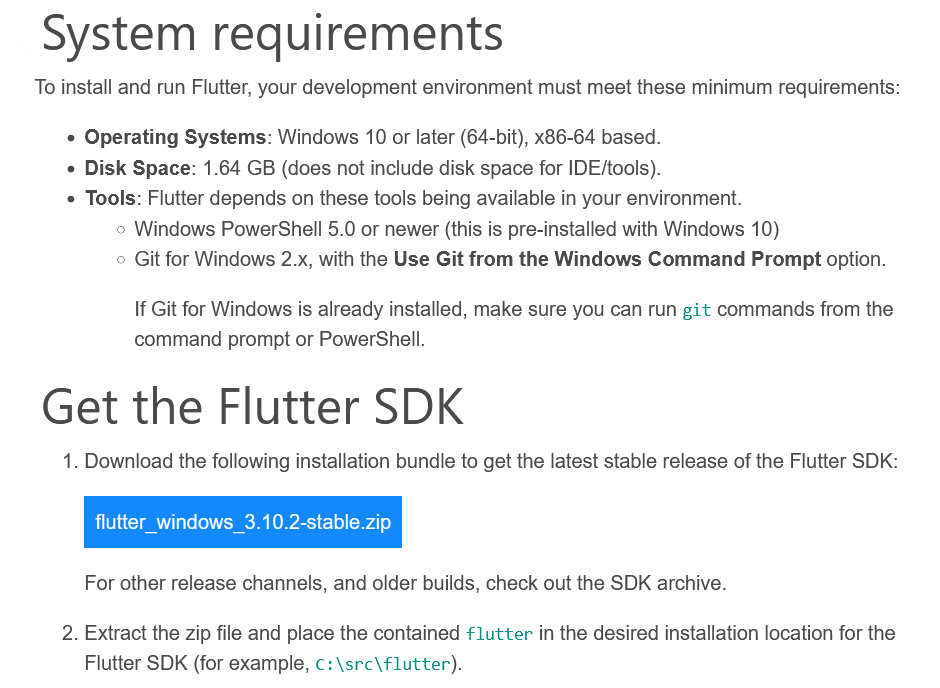
* **sur la carte**

python -m pip install requests

sudo apt install adb

## Installation de flutter et Dart

Téléchargez les fichiers compresses et extraire les dans un dossier comme indiqué dans le site puis ajoutez les binaires au path du système

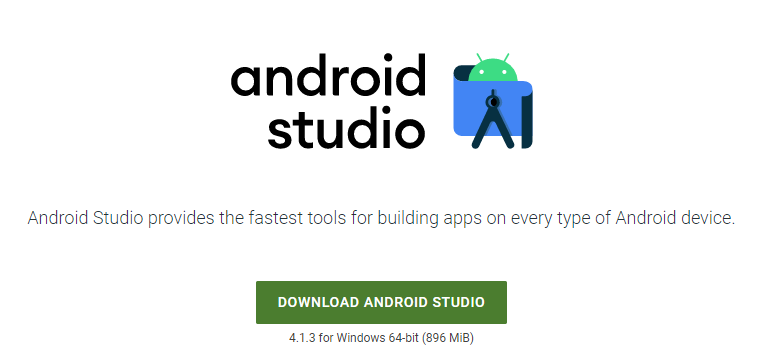


Installez l’extension officielle de flutter



## Installation de Android studio et création de machine virtuelle

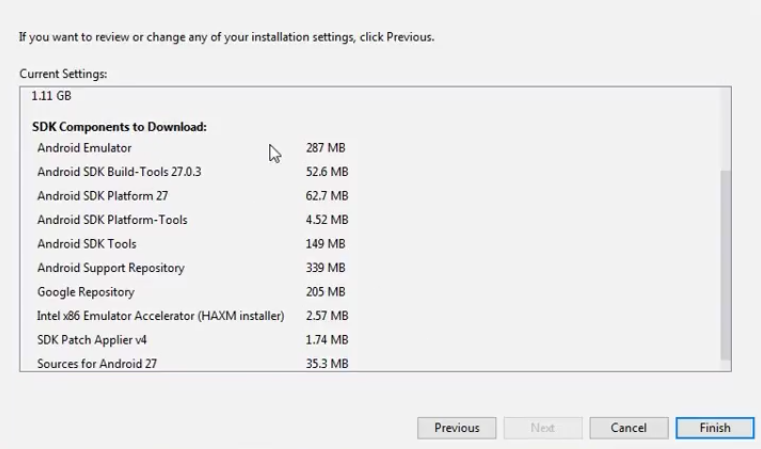
Téléchargez Android studio et lancez-le



Suivez l’installateur et lancez Android studio dès qu’il fini

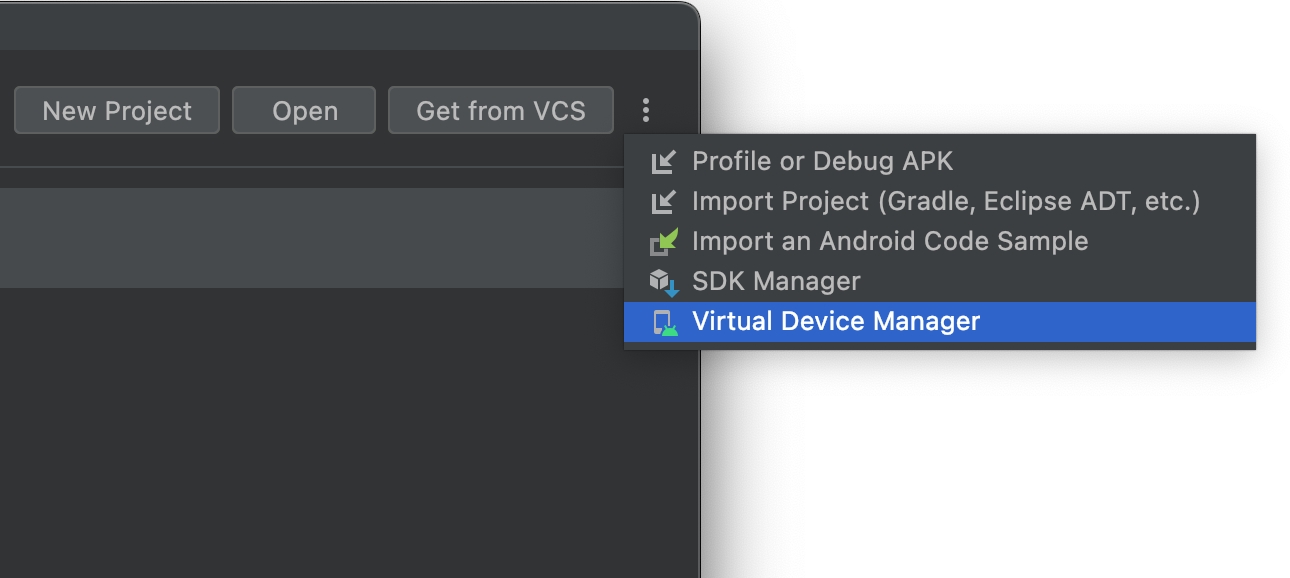


Téléchargez SDK components

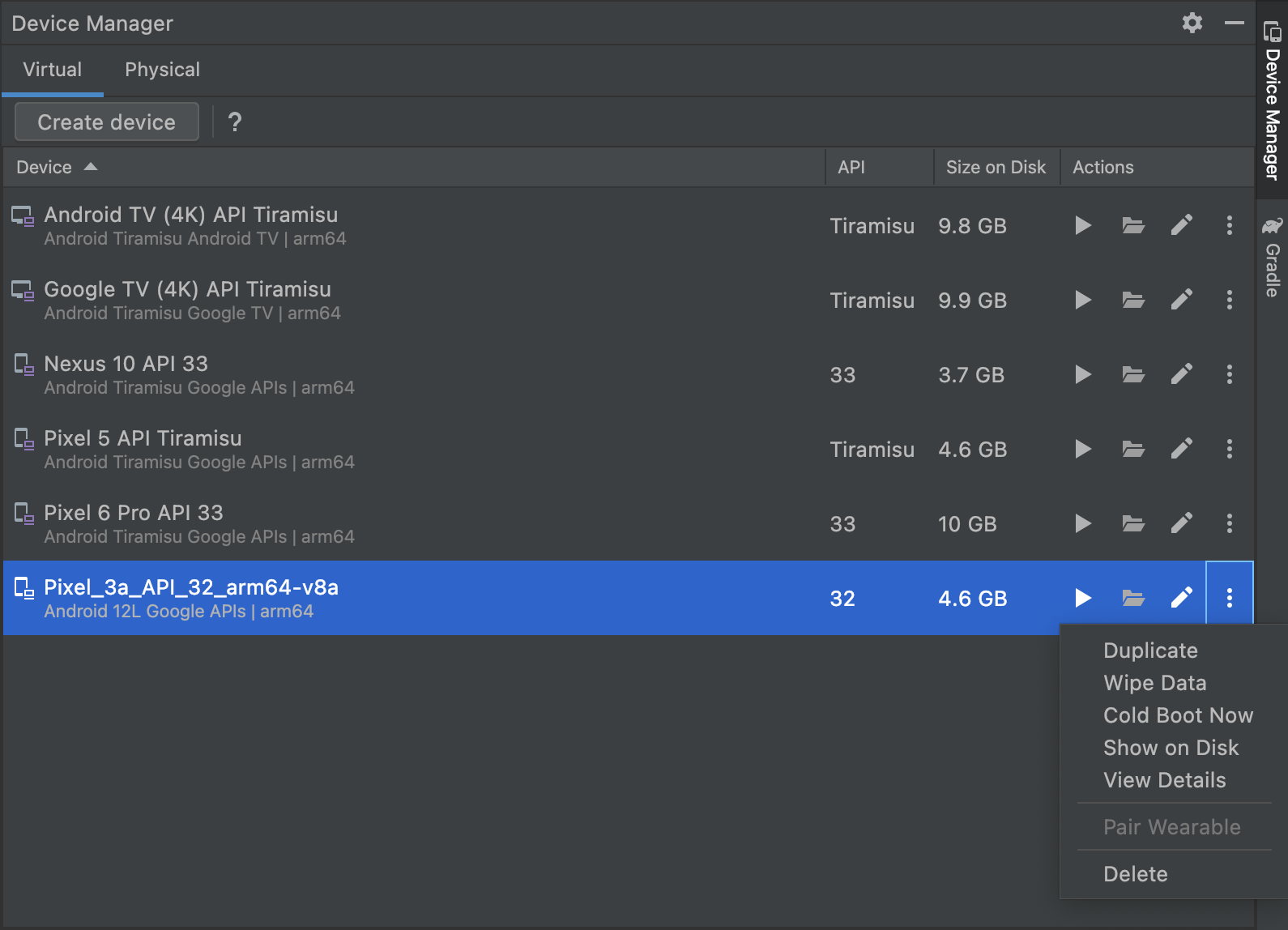


Créer un émulateur Android

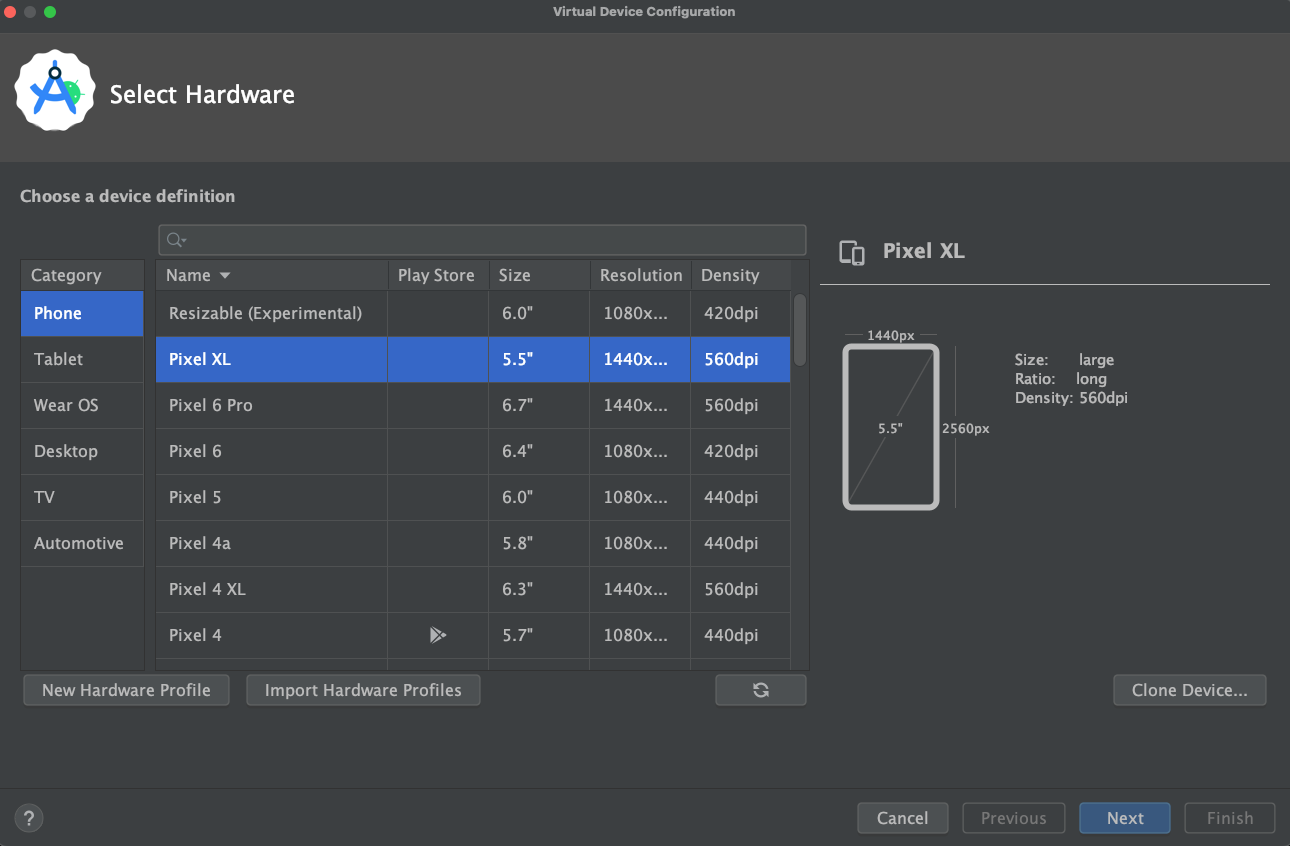
Sur l'écran d'accueil d'Android Studio, sélectionnez **More Actions > Virtual Device Manager**

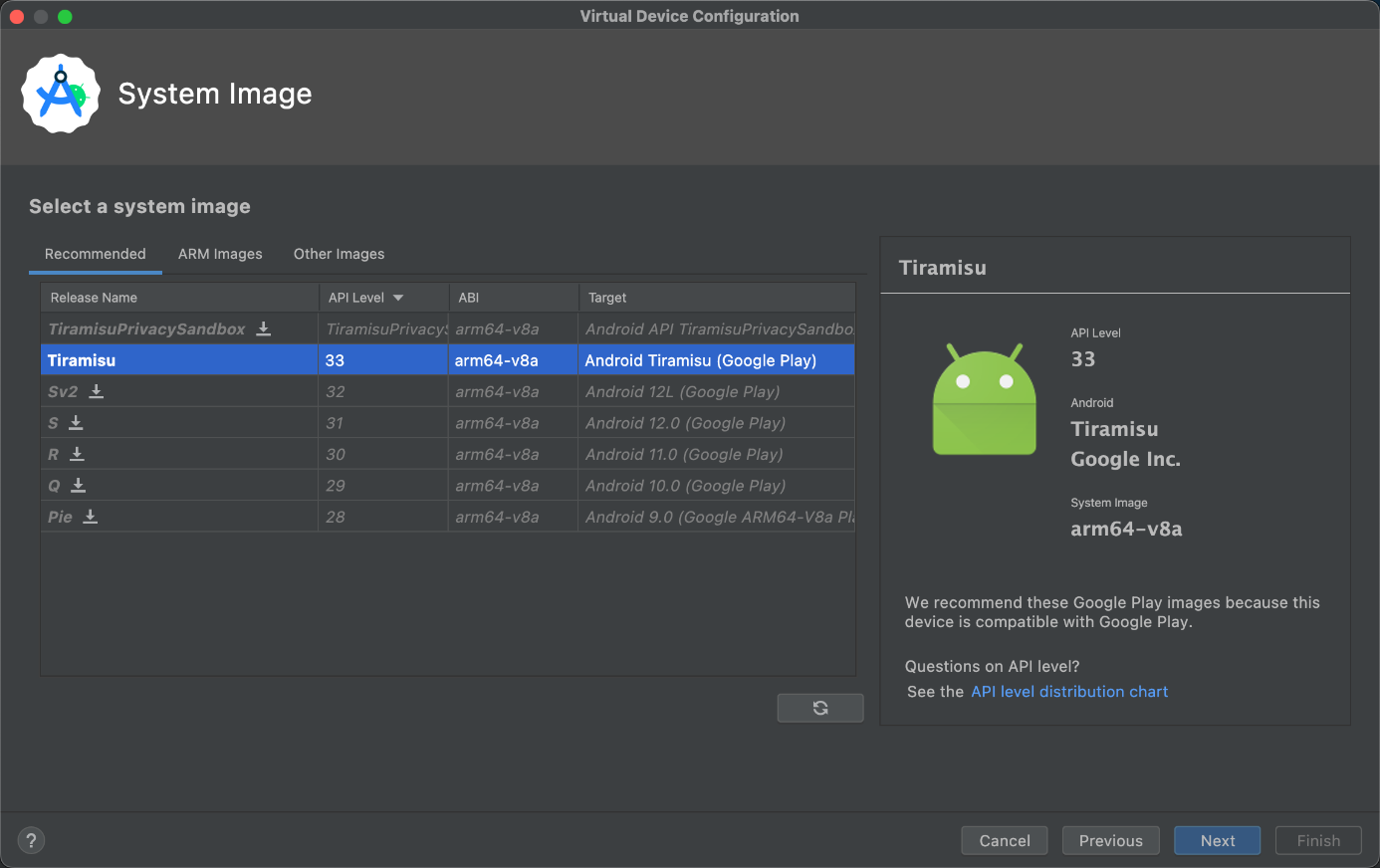


Cliquez sur **Create device**

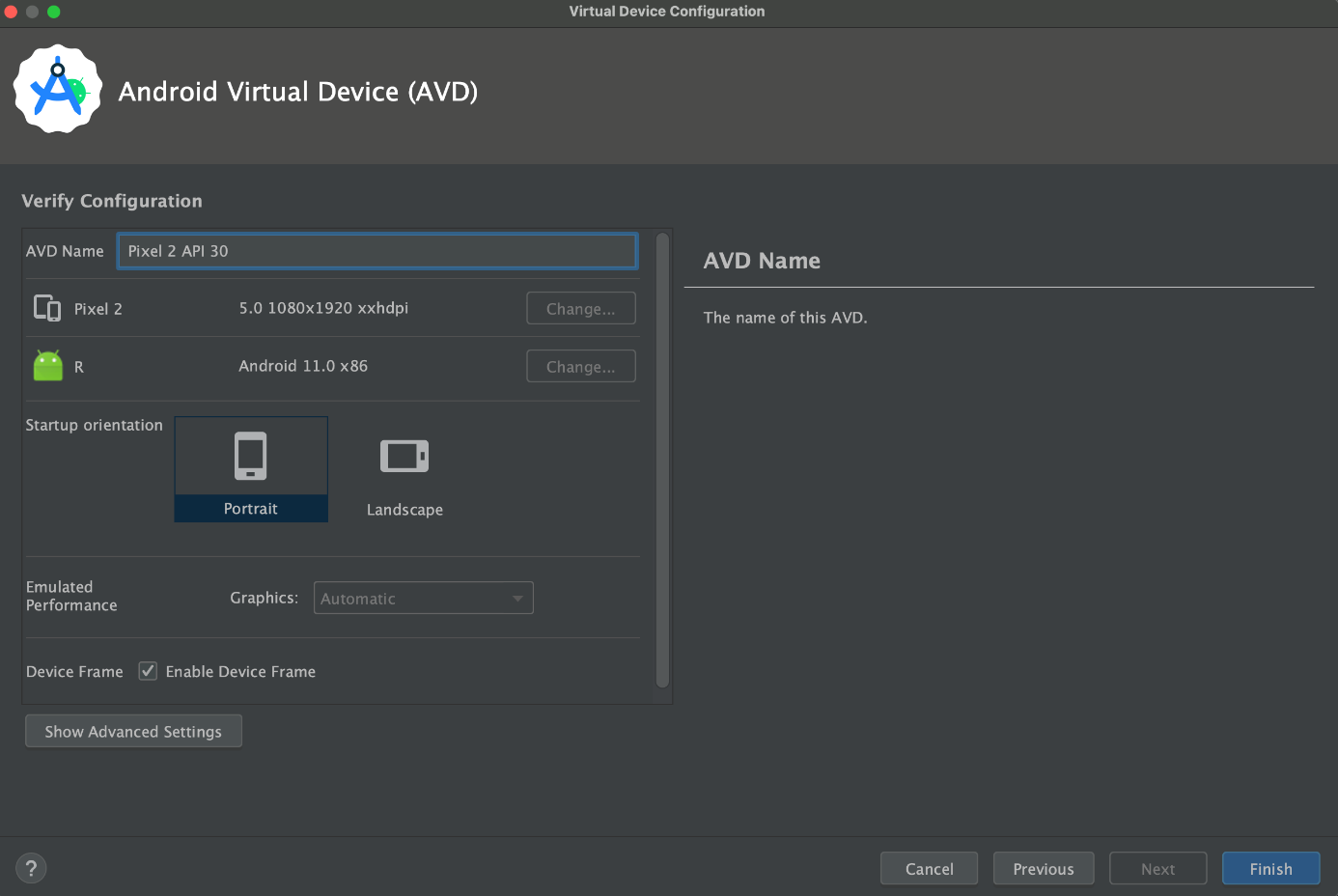


Choisissez un appareil

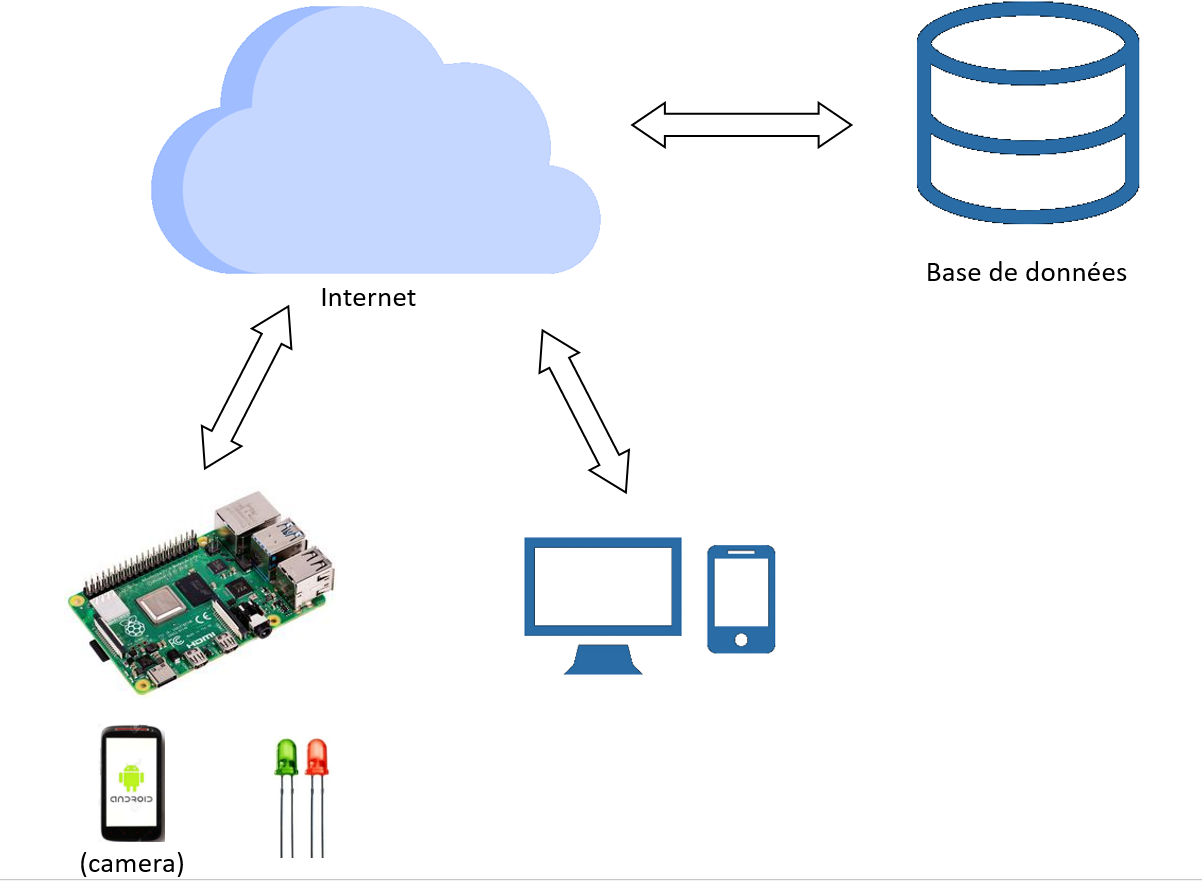
Sélectionnez l’image système



Ajustez votre machine virtuelle puis cliquez « finish »

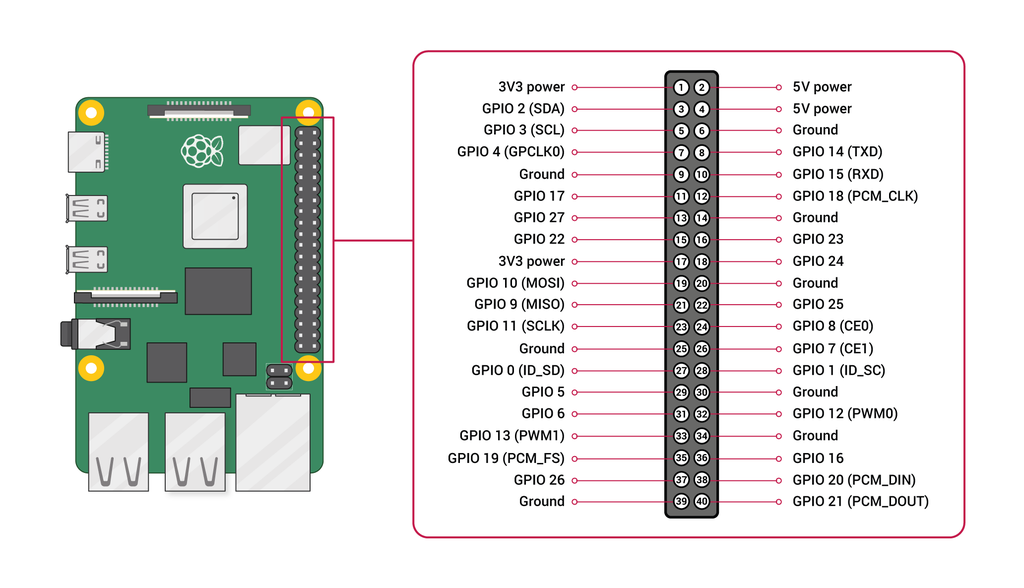


# Le schéma principal

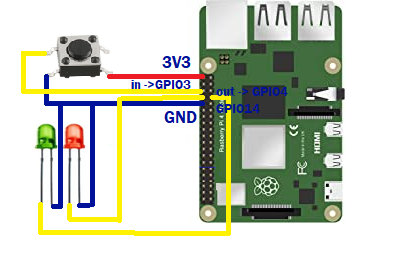


# Le montage

### Les ports de Raspberry pi 3B



### Réaliser le montage suivant :



# Le code

## Le code de la carte

Le code suivant prend une image, l’envoie au serveur et attends une réponse.

Ecrire le code suivant puis exécuter le

*import* *import* requests

*import* RPi.GPIO *as* GPIO

*from* datetime *import* datetime

*from* time *import* sleep

*from* traceback *import* print\_exc

*import* os

"""layout pins

               3V3  (1) (2)  5V

            GPIO GPIO2  (3) (4)  5V

    B1 -->   GPIO3  (5) (6)  GND

    R1 -->   GPIO4  (7) (8)  GPIO14 <--  V1

               GND  (9) (10) GPIO15 <--  B2

    R2 -->  GPIO17 (11) (12) GPIO18 <--  V2

    B3 -->  GPIO27 (13) (14) GND

    R3 -->  GPIO22 (15) (16) GPIO23 <--  V3

               3V3 (17) (18) GPIO24

            GPIO10 (19) (20) GND

             GPIO9 (21) (22) GPIO25

            GPIO11 (23) (24) GPIO8

               GND (25) (26) GPIO7

             GPIO0 (27) (28) GPIO1

             GPIO5 (29) (30) GND

             GPIO6 (31) (32) GPIO12

            GPIO13 (33) (34) GND

            GPIO19 (35) (36) GPIO16

            GPIO26 (37) (38) GPIO20

               GND (39) (40) GPIO21

"""

GPIO.setmode(GPIO.BCM)

Pins = {*# B :[R, V]*

    5  :[7 ,8],

    10 :[11 ,12],

    13 :[15 ,16]

}

API\_URL = "http:// :5000"

*def* capture\_and\_send\_image(*n*):

    image\_path = "img.jpg"

    makePic(image\_path)

*with* open(image\_path, "rb") *as* image\_file:

        file = {"image": image\_file}

        data = {"button": n}

        response = requests.post(API\_URL+"/recognize-face", *files*=file, *data*=data, *timeout*=500)

*if* response.status\_code == 200:

        led = Pins[n][response.json()["response"]]

        GPIO.output(led, GPIO.HIGH)  *# LED corr*

        sleep(.5)

        GPIO.output(led, GPIO.LOW)

*#    os.remove(image\_path)*

*def* makePic(*path*):

    os.system("adb shell input keyevent KEYCODE\_CAMERA")

    sleep(.6)

    os.system("adb shell input keyevent KEYCODE\_CAMERA")

    os.system(*f*"aadb pull sdcard/DCIM/camera/$(adb shell ls sdcard/DCIM/camera -t |head -1) {path}")

*def* button\_callback(*channel*):

    button\_number = Pins.keys().index(channel)

    print(*f*"Button {button\_number} pressed")

    capture\_and\_send\_image(button\_number)

*for* button,leds *in* Pins.items():

    GPIO.setup(button, GPIO.IN)

    GPIO.setup(leds, GPIO.OUT)

    GPIO.add\_event\_detect(button, GPIO.RISING, *callback*=button\_callback)

ERROR\_LOG = "card.log"

*with* open(ERROR\_LOG, *mode*='a') *as* f:

*try*:

*while* True:

            sleep(.5)

*except* KeyboardInterrupt:

        print("KeyboardInterrupt")

*except* Exception *as* e:

        f.write(datetime.now().strftime("%Y%m%d%H%M%S")+"  ")

        print\_exc(*file*=f)

*finally*:

        GPIO.cleanup()

## Le code coté serveur

Le code suivant resoit une image et envoie une reponse, envoi une liste de l’hystorique des essais et les images correspondantes.

Ecrire le code suivant puis executer le

*import* json, sqlite3, os

*from* flask *import* Flask, request, jsonify, send\_file

*from* datetime *import* datetime

*from* deepface *import* DeepFace

*from* traceback *import* print\_exc

*from* flask\_cors *import* CORS

app = Flask(\_\_name\_\_)

CORS(app)

*# fichiers et dossiers*

KNOWN\_FACES\_DIR = "known\_faces"

PERMISSIONS\_FILE = "permissions.json"

HISTORY\_DIR = "history"

LOG\_FILE = "attempts.db"

ERROR\_LOG = "app.log"

*def* get\_connection():

    conn = sqlite3.connect(LOG\_FILE)

    conn.execute(

        '''

        CREATE TABLE IF NOT EXISTS logs (

            timestamp TEXT,

            button INTEGER,

            response INTEGER,

            person TEXT

        )

        '''

    )

    conn.commit()

    conn.row\_factory = sqlite3.Row

*return* conn

*with* open(PERMISSIONS\_FILE) *as* f:

    KNOWN\_PERSONS = json.load(f)

*# reconnaissance et log*

*@app*.*route*('/recognize-face', *methods*=['POST'])

*def* recognize\_face():

    image = request.files['image']

    button = request.form.get('button')

    timestamp = datetime.now().strftime("%Y%m%d%H%M%S")

    image\_filename = *f*"{HISTORY\_DIR}/{timestamp}\_{button}.jpg"

    image.save(image\_filename)

    response = False

    person\_name = "unknown"

*try*:

*for* person, access *in* KNOWN\_PERSONS.items():

            person\_folder = *f*"{KNOWN\_FACES\_DIR}/{person}.jpg"

            face\_recognizer = DeepFace.verify(image\_filename, person\_folder)

*if* face\_recognizer["verified"] and access[*int*(button)] == "1":

                person\_name = person

                response = True

*break*

*except* ValueError:print("ValueError")

*with* get\_connection() *as* conn:

        log\_data = (timestamp, *int*(button), *int*(response), person\_name)

        querry = '''

            INSERT INTO logs (timestamp, button, response, person)

            VALUES (?, ?, ?, ?)

        '''

        conn.execute(querry, log\_data)

        conn.commit()

*return* jsonify({'response': response}), 200

*# envoi d'image*

*@app*.*route*('/image-data', *methods*=['GET'])

*def* get\_image\_data():

    image\_filename = request.args.get('name')+".jpg"

    image\_path = os.path.join(HISTORY\_DIR, image\_filename)

*if* not os.path.isfile(image\_path):

*return* jsonify({'error': 'Image not found.'}), 404

*return* send\_file(image\_path, *mimetype*='image/jpeg')

*#liste des essais*

*@app*.*route*('/attempts', *methods*=['GET'])

*def* get\_attempts():

    page = *int*(request.args.get('page', 1))

    per\_page = *int*(request.args.get('per\_page', 10))

*with* get\_connection() *as* conn:

        cursor = conn.cursor()

        query = "SELECT COUNT(\*) FROM logs"

        cursor.execute(query)

        total\_attempts = cursor.fetchone()[0]

        total\_pages = (total\_attempts - 1) // per\_page + 1

        offset = (page - 1) \* per\_page

        query = "SELECT \* FROM logs ORDER BY timestamp DESC LIMIT ? OFFSET ?"

        cursor.execute(query, (per\_page, offset))

        rows = cursor.fetchall()

*return* jsonify({

        'attempts': [{

                'timestamp': row['timestamp'],

                'button': row['button'],

                'response': *bool*(row['response']),

                'person': row['person']

            }

*for* row *in* rows

        ],

        'page': page,

        'has\_previous\_page' : page >1,

        'has\_next\_page' : page <total\_pages,

        'total\_pages': total\_pages

    }), 200

*if* \_\_name\_\_ == '\_\_main\_\_':

*with* open(ERROR\_LOG, *mode*='a') *as* f:

*try*:

            app.run(*host*='0.0.0.0', *port*=5000)

*except* KeyboardInterrupt:

            print("Server stopped with CTRL+C")

*except* Exception *as* e:

            f.write(datetime.now().strftime("%Y%m%d%H%M%S") + "  ")

            print\_exc(*file*=f)

## le code de l’application client

Le code suivant se connecte au serveur et resoit une liste de l’historique des essais et les images correspondantes.

Ecrire le code suivant puis compiler le a la platteforme desirée

import 'dart:async';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

*void* main() {

  runApp(MyApp());

}

class GlobalVariables {

  static String websiteAddress = 'localhost';

}

class MyApp extends StatefulWidget {

*@override*

  \_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

  bool \_isDarkTheme = false;

*void* updateWebsiteAddress(String newAddress) {

    setState(() {

      GlobalVariables.websiteAddress = newAddress;

    });

  }

*@override*

  Widget build(BuildContext context) {

*return* MaterialApp(

      title: 'Image Viewer',

      theme: \_isDarkTheme ? ThemeData.dark() : ThemeData.light(),

      home: Builder(

        builder: (context) => Scaffold(

          appBar: AppBar(

            title: Text('log history'),

          ),

          drawer: Drawer(

            child: ListView(

              padding: EdgeInsets.zero,

              children: [

                DrawerHeader(

                  decoration: BoxDecoration(

                    color: Colors.blue,

                  ),

                  child: Text(

                    'Menu',

                    style: TextStyle(

                      color: Colors.white,

                      fontSize: 24,

                    ),

                  ),

                ),

                ListTile(

                  leading: Icon(Icons.settings),

                  title: Text('Settings'),

                  onTap: () *async* {

                    final websiteAddress = *await* Navigator.push(

                      context,

                      MaterialPageRoute(

                        builder: (context) => SettingsPage(

                          currentAddress: GlobalVariables.websiteAddress,

                        ),

                      ),

                    );

*if* (websiteAddress != null) {

                      updateWebsiteAddress(websiteAddress);

                    }

                  },

                ),

                ListTile(

                  leading: Icon(Icons.brightness\_medium),

                  title: Text('Toggle Theme'),

                  onTap: () {

                    setState(() {

                      \_isDarkTheme = !\_isDarkTheme;

                    });

                    Navigator.pop(context); *// Close the drawer*

                  },

                ),

              ],

            ),

          ),

          body: ImageListScreen(),

        ),

      ),

    );

  }

}

class ImageListScreen extends StatefulWidget {

*@override*

  \_ImageListScreenState createState() => \_ImageListScreenState();

}

class \_ImageListScreenState extends State<ImageListScreen> {

  List<Attempt> \_attempts = [];

  int \_currentPage = 1;

  int \_maxPage = 10;

  bool \_isLoading = false;

  final GlobalKey<RefreshIndicatorState> \_refreshIndicatorKey =

      GlobalKey<RefreshIndicatorState>();

*@override*

*void* initState() {

*super*.initState();

    fetchAttempts();

  }

  Future<*void*> fetchAttempts() *async* {

    setState(() {

      \_isLoading = true;

    });

    final url = Uri.parse('http://${GlobalVariables.*websiteAddress*}:5000/attempts?page=$*\_currentPage*');

    final response = *await* http.get(url);

*if* (response.statusCode == 200) {

      final data = jsonDecode(response.body);

      final attempts = List<Attempt>.from(data['attempts'].map((x) => Attempt.fromJson(x)));

      setState(() {

        \_attempts = attempts;

        \_isLoading = false;

      });

    } *else* {

      setState(() {

        \_isLoading = false;

      });

      showDialog(

        context: context,

        builder: (ctx) => AlertDialog(

          title: Text('Error'),

          content: Text('Failed to fetch attempts.'),

          actions: [

            TextButton(

              child: Text('OK'),

              onPressed: () {

                Navigator.of(ctx).pop();

              },

            ),

          ],

        ),

      );

    }

  }

  Future<*void*> goToPage(int page) *async* {

    setState(() {

      \_currentPage = page;

    });

*await* fetchAttempts();

  }

*void* viewImageDetails(Attempt attempt) {

    Navigator.push(

      context,

      MaterialPageRoute(

        builder: (context) => ImageDetailsScreen(attempt: attempt),

      ),

    );

  }

  Widget buildPageNumber(int page) {

    final bool isCurrentPage = page == \_currentPage;

    final bool isClickable = !isCurrentPage;

*return* MouseRegion(

      cursor: isClickable ? SystemMouseCursors.click : SystemMouseCursors.basic,

      child: GestureDetector(

        onTap: isClickable ? () => goToPage(page) : null,

        child: Container(

          decoration: BoxDecoration(

            color: isCurrentPage ? Colors.blue : Colors.transparent,

            borderRadius: BorderRadius.circular(4),

          ),

          padding: EdgeInsets.all(8),

          child: Text(

            '$*page*',

            style: TextStyle(

              color: isCurrentPage ? Colors.white : Colors.black,

              fontWeight: isCurrentPage ? FontWeight.bold : FontWeight.normal,

            ),

          ),

        ),

      ),

    );

  }

  List<Widget> buildPageNumbers() {

    final List<Widget> pageNumbers = [];

*// Add the first page number*

    pageNumbers.add(buildPageNumber(1));

*// Add the page numbers with ellipsis in between*

*if* (\_currentPage > 3) {

      pageNumbers.add(Padding(

        padding: EdgeInsets.symmetric(horizontal: 4),

        child: Text('...'),

      ));

    }

*for* (int i = \_currentPage - 1; i <= \_currentPage + 1; i++) {

*if* (i > 1 && i < \_maxPage) {

        pageNumbers.add(buildPageNumber(i));

      }

    }

*if* (\_currentPage < \_maxPage - 2) {

      pageNumbers.add(Padding(

        padding: EdgeInsets.symmetric(horizontal: 4),

        child: Text('...'),

      ));

    }

*// Add the last page number*

    pageNumbers.add(buildPageNumber(\_maxPage));

*return* pageNumbers;

  }

  Future<*void*> \_handleRefresh() *async* {

*await* fetchAttempts();

  }

*@override*

  Widget build(BuildContext context) {

*return* Scaffold(

      body: RefreshIndicator(

        key: \_refreshIndicatorKey,

        onRefresh: \_handleRefresh,

        child: \_isLoading

            ? Center(child: CircularProgressIndicator())

            : \_attempts.isEmpty

                ? Center(child: Text('No attempts found.'))

                : ListView.builder(

                    itemCount: \_attempts.length,

                    itemBuilder: (ctx, index) {

                      final attempt = \_attempts[index];

*return* ListTile(

                        title: Text(attempt.timestamp),

                        subtitle: Text(

                            'Button ${*attempt*.*button*}, ${*attempt*.*person*}, ${*attempt*.*response* ? "Allowed" : "Denied"}'),

                        onTap: () {

                          viewImageDetails(attempt);

                        },

                      );

                    },

                  ),

      ),

      bottomNavigationBar: BottomAppBar(

        child: Row(

          mainAxisAlignment: MainAxisAlignment.center,

          children: buildPageNumbers(),

        ),

      ),

    );

  }

}

class ImageDetailsScreen extends StatelessWidget {

  final Attempt attempt;

  ImageDetailsScreen({required *this*.attempt});

*@override*

  Widget build(BuildContext context) {

*return* Scaffold(

      appBar: AppBar(

        title: Text('Image Details'),

      ),

      body: Column(

        children: [

          Image.network(

            'http://${GlobalVariables.*websiteAddress*}:5000/image-data?name=${*attempt*.*timestamp*}\_${*attempt*.*button*}',

            loadingBuilder: (context, child, progress) {

*if* (progress == null) *return* child;

*return* Center(child: CircularProgressIndicator());

            },

          ),

          SizedBox(height: 16),

          Text('Timestamp: ${*attempt*.*timestamp*}'),

          Text('Button: ${*attempt*.*button*}'),

          Text('Response: ${*attempt*.*response* ? "Allowed" : "Denied"}'),

          Text('Person: ${*attempt*.*person*}'),

        ],

      ),

    );

  }

}

class SettingsPage extends StatefulWidget {

  final String currentAddress;

  SettingsPage({required *this*.currentAddress});

*@override*

  \_SettingsPageState createState() => \_SettingsPageState();

}

class \_SettingsPageState extends State<SettingsPage> {

  late TextEditingController \_controller;

*@override*

*void* initState() {

*super*.initState();

    \_controller = TextEditingController(text: widget.currentAddress);

  }

*@override*

  Widget build(BuildContext context) {

*return* Scaffold(

      appBar: AppBar(

        title: Text('Settings'),

      ),

      body: Padding(

        padding: const EdgeInsets.all(16.0),

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.start,

          children: [

            Text(

              'Website Address',

              style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),

            ),

            SizedBox(height: 8),

            TextField(

              controller: \_controller,

              decoration: InputDecoration(

                border: OutlineInputBorder(),

                hintText: 'Enter website address',

              ),

            ),

            SizedBox(height: 16),

            ElevatedButton(

              onPressed: () {

                Navigator.pop(context, \_controller.text);

              },

              child: Text('Save'),

            ),

          ],

        ),

      ),

    );

  }

*@override*

*void* dispose() {

    \_controller.dispose();

*super*.dispose();

  }

}

class Attempt {

  final String timestamp;

  final int button;

  final bool response;

  final String person;

  Attempt({

    required *this*.timestamp,

    required *this*.button,

    required *this*.response,

    required *this*.person,

  });

  factory Attempt.fromJson(Map<String, dynamic> json) {

*return* Attempt(

      timestamp: json['timestamp'],

      button: json['button'],

      response: json['response'],

      person: json['person'],

    );

  }

}

## Capture d’écran de l’application

