Intelligent mirror

Wykonawcy Projektu

1. Szymon Marciniak - założyciel projektu oraz programista głównej aplikacji,
2. Sebastian Dolata - programista ds. backendu aplikacji komórkowej oraz strony internetowej,
3. Wiktor Dembny - Programista ds. frontendu

Spis treści

Wygląd


Calendar
10°C
17:52
Spread love everywhere you go. Let no one ever come to you without leaving happier. -Mother Teresa
Calendar
10°C
Spread love everywhere you go. Let no one ever come to you without leaving happier. -Mother Teresa
Calendar
10°C
Spread love everywhere you go. Let no one ever come to you without leaving happier. -Mother Teresa
Calendar
10°C
18:03
Spread love everywhere you go. Let no one ever come to you without leaving happier. -Mother Teresa


Calendar
2022-04-25 16:30
urodziny dziadka
2022-04-26 08:00
sprawdzian z matmy
2022-04-27 19:25
trening
2022-04-28 12:00
spotkanie biznesowe
Sunday. Apr 24
18:04
10°C
HUM 85 % I 1002 hPa
Spread love everywhere you go. Let no one ever come to you without leaving happier. -Mother Teresa
MARCIN


10°C
HUM 85 % I 1002 hPa
Sunday. Apr 24
18:05
Spread love everywhere you go. Let no one ever come to you without leaving happier. -Mother Teresa
MARCIN


Sunday. Apr 24
18:03
10°C
HUM 85 % I 1002 hPa
Calendar
2022-04-25 16:30
urodziny dziadka
2022-04-26 08:00
sprawdzian z matmy
2022-04-27 19:25
trening
2022-04-28 12:00
spotkanie biznesowe

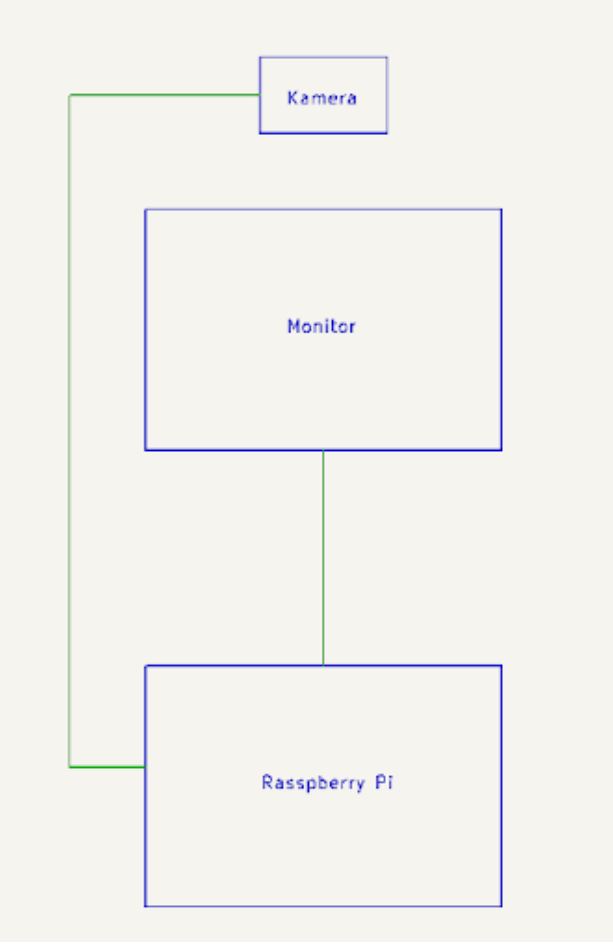
Intelligent Mirror – główna aplikacja

Lustro jest nierozerwalnym elementem życia każdego z nas. Wielokrotnie przyglądamy się swojemu odbiciu, a zwłaszcza o poranku. Ten projekt powstał w trosce o lepsze gospodarowanie czasem, by częste codzienne czynności móc wykonać spojrzeniem w zwierciadło. Dodatkowo dzięki nowoczesnemu designowi może lustro to wkomponować się we wnętrze współczesnych domów wzbudzając zazdrość znajomych. Wyposażone w sztuczną inteligencję oprogramowanie pozwala na wykrywanie twarzy, aby nikt niepożądany nie miał dostępu do naszych danych, a także na sterowanie gestami, by nie brudzić szyby!

Budowa

Projekt zbudowany jest z:

- Raspberry Pi 4b, które jest elementem sterowniczym projektu,
- Monitor o przekątnej 24 cali, który pozwala na podświetlenie funkcji lustra,
- Lustro weneckie 60x80 cm, które jednocześnie spełnia funkcje normalnego lustra oraz pozwala na wyświetlenie na nim ustawionych przez użytkownika funkcji,
- Kamera, dzięki której możliwe jest rozpoznawania twarzy użytkownika oraz jego gesty,
- Stelaż, mający na celu imitować ścianę na której jest zawieszone lustro.



Sterowanie

W momencie, w którym podchodzimy do lustra, sztuczna inteligencja automatycznie odczytuje naszą twarz (oczywiście jeśli wcześniej założyliśmy sobie na nim konto). lustro w tym momencie rozpoznaje kto przed nim stoi. Wnet w prawym dolnym rogu pojawia się nasz nick. Po czym wyświetlają się spersonalizowane funkcje (omówione poniżej) dla danej osoby. Rozmieszczają się one płynnie do ustalonej wcześniej przez użytkownika pozycji. Jeśli logujemy się pierwszy raz - domyślnie wszystkie funkcje są wyłączone i tylko czekają aby pokazać co potrafią. Aby to zrobić należy włączyć na aplikacji komórkowej możliwość sterowania lustrem 'vm' (virtual mouse) i pokazać w stronę kamery gest “zwycięstwa” podnosząc dwa palce ku górze.

Gdy na lustrze ukaże się kursor możemy już podnieść wszystkie palce (tryb przesuwania), oraz zacząć poruszać naszą dłonią. Kursor w tym momencie będzie za nią podążał. Jeśli najedziemy na wybraną przez nas ikonkę bądź funkcję zamykamy co najmniej 3 palce na sekundę, a następnie otwieramy je, tym samym potwierdzając nasz wybór.

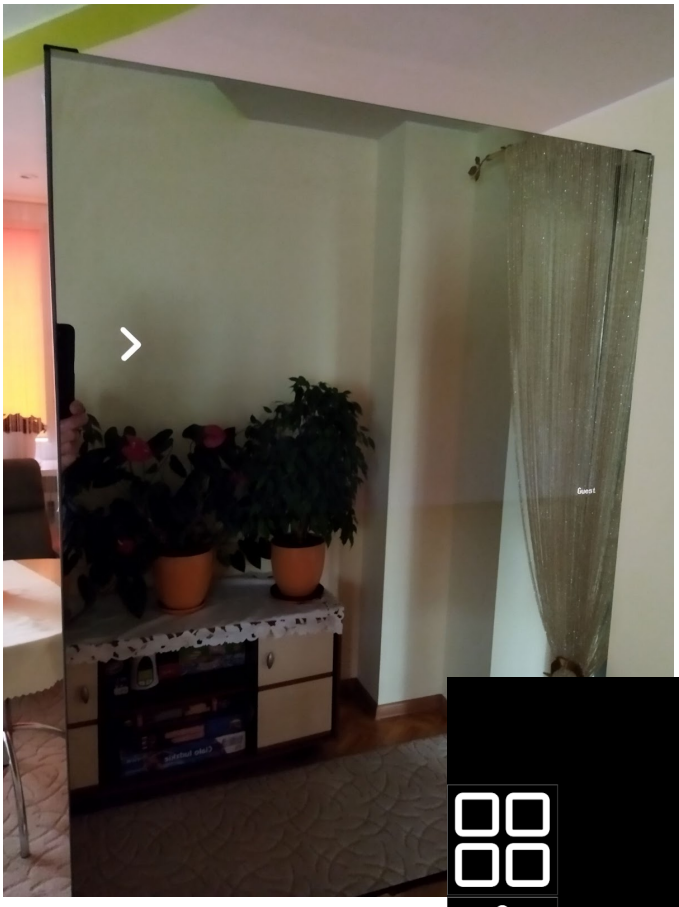
Kursor śledzi położenie palca wskazującego i nie zaleca się więc jego zamykania w momencie wyboru funkcji.

Opis funkcji

Funkcje, które posiada nasze lustro dzielą się na 3 kategorie:

1. **Funkcje codziennego użytku,**
2. **Funkcje z dostępem do internetu,**
3. **Funkcje odpowiedzialne za sterowanie oświetleniem i roletami w pokoju.**

Aby wybrać kategorię należy otworzyć główny pasek zadań, klikając w strzałkę. Spowoduje to otworzenie głównego paska zadań.



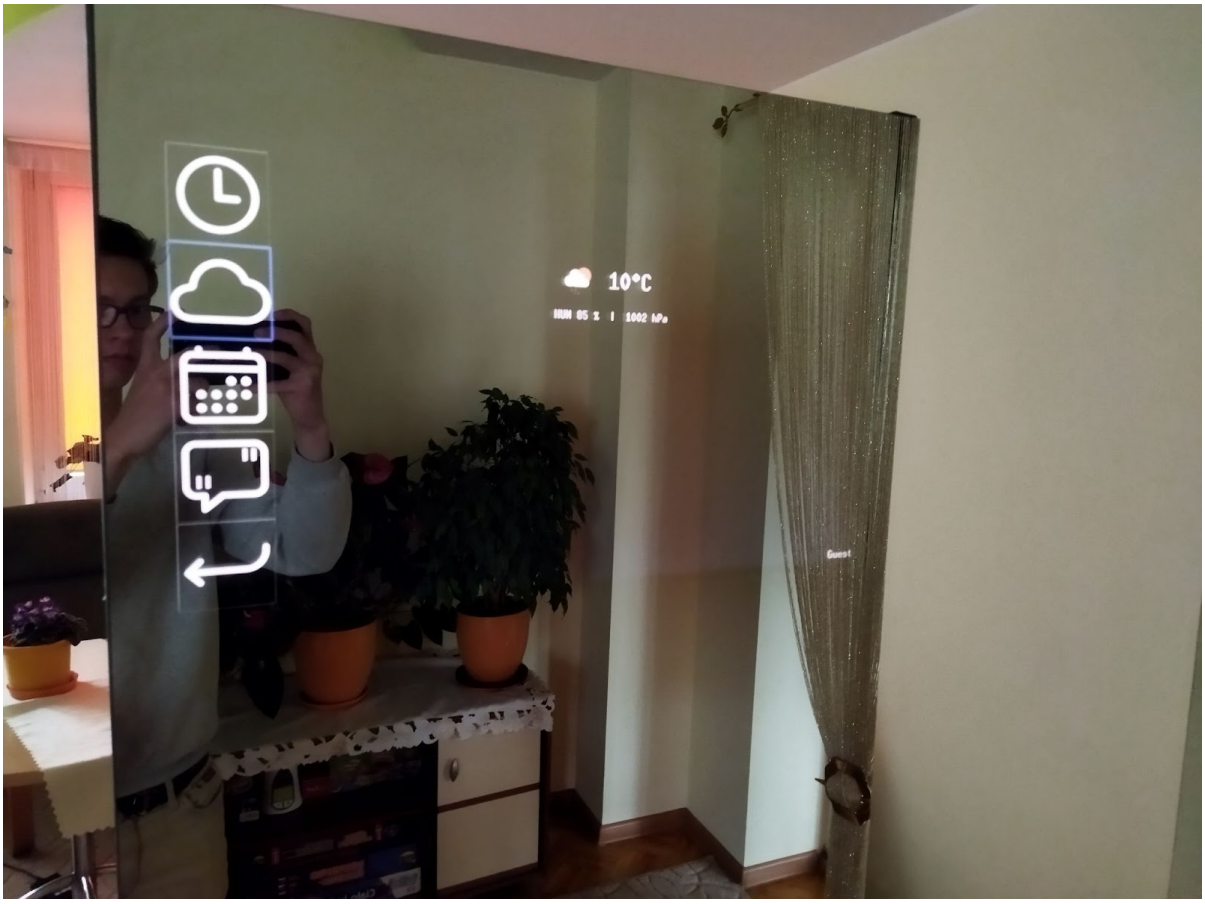
Aby schować główny pasek, należy analogicznie kliknąć strzałkę, tym razem skierowaną w lewą stronę

1. **Funkcje codziennego użytku**
   (ikona na samej górze - 4 kwadraty):

Kliknięcie strzałki ‘return’ (na dole) pozwala na przejście o krok do tyłu.

Data
Jest to prosta i podstawowa funkcja, która polega na wyświetlaniu obecnej daty i godziny.

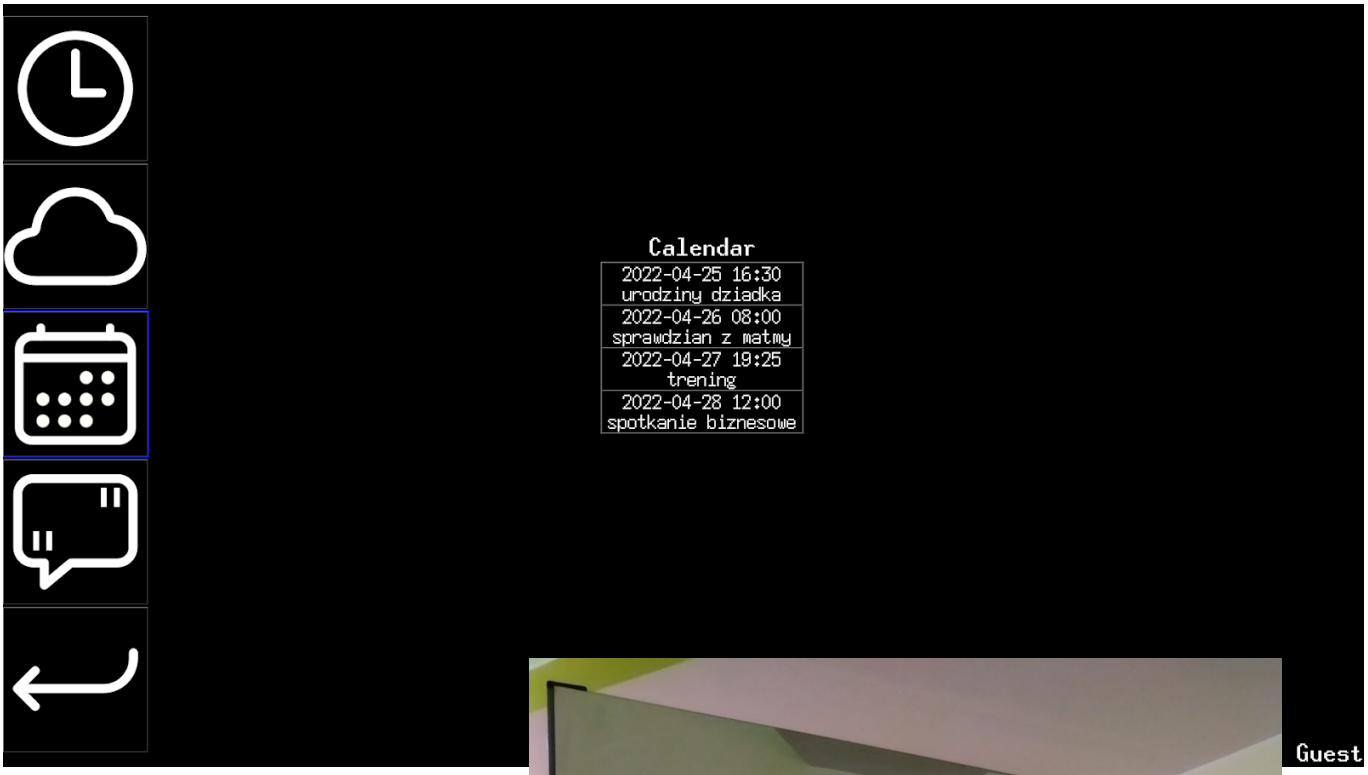

14°C
HUM 53 %
1005 hPa
Guest

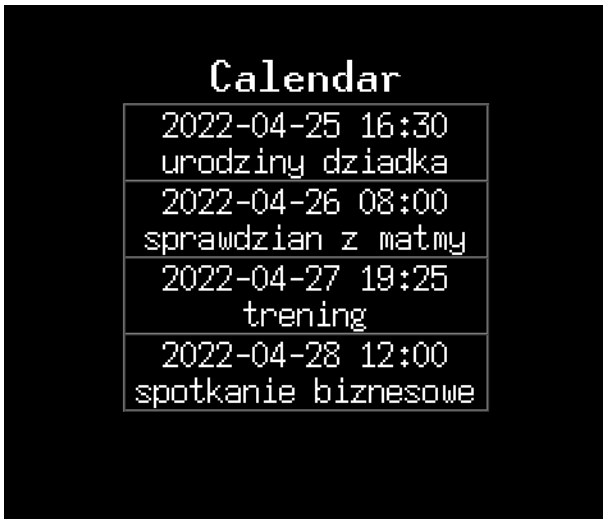

10°C

Cytat
Jest to wyróżniająca się funkcja, mająca na celu codzienne zainspirowanie człowieka do działania podczas porannego przyglądania się w zwierciadle. Cytaty są dostępne w języku angielskim, aby nawet w tak banalny sposób nabywać umiejętności językowe.

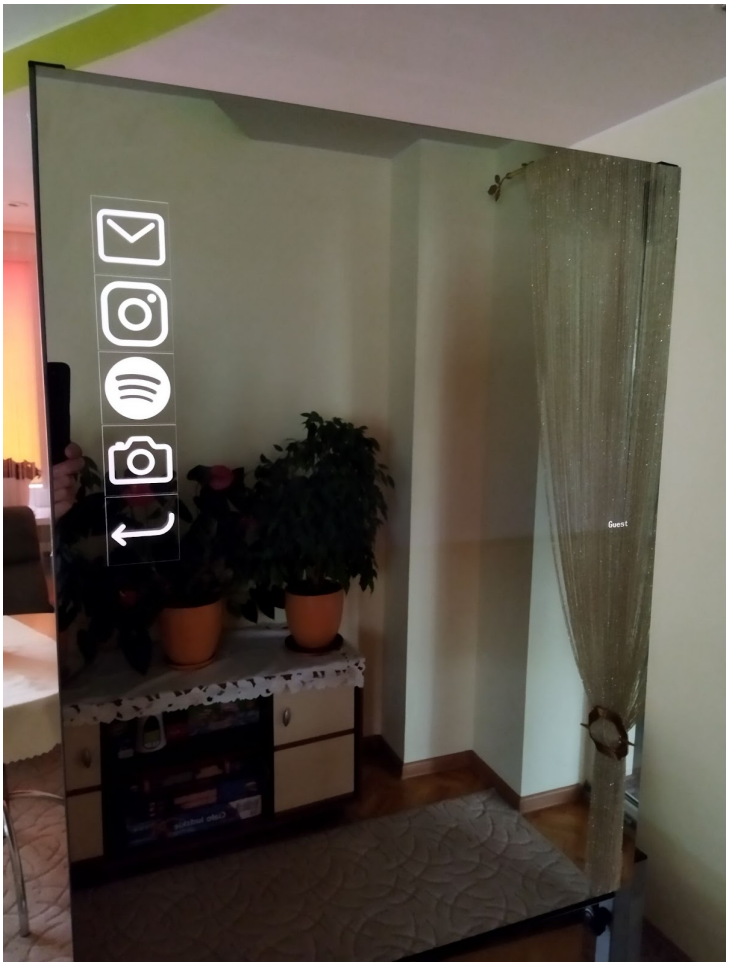
Wydarzenia

Funkcja przygotowana dla biznesmenów i nie tylko. Wyświetla zaplanowane wydarzenia rozpoznanej osoby na najbliższy czas, aby niczego nie zapomnieć podczas porannego roztargnienia. Dodawać zdarzenia można z pozycji aplikacji internetowej lub mobilnej.



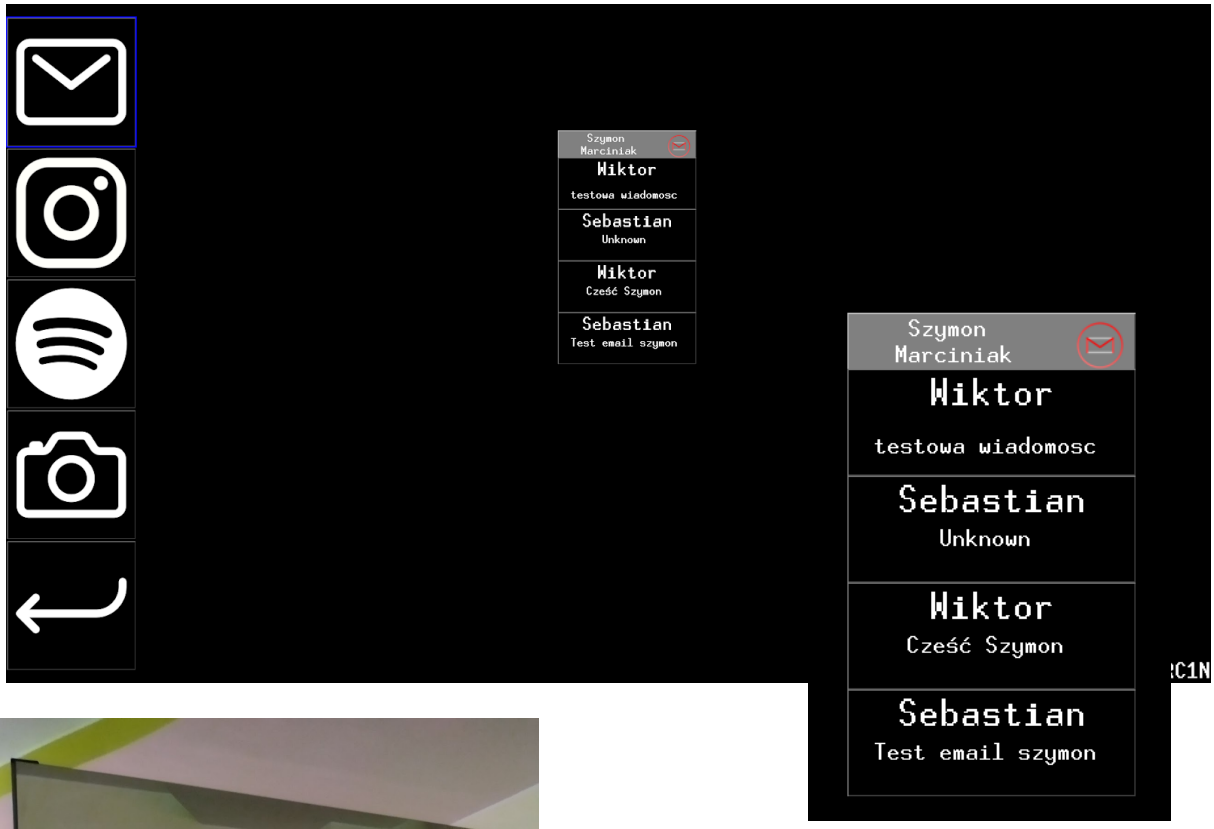


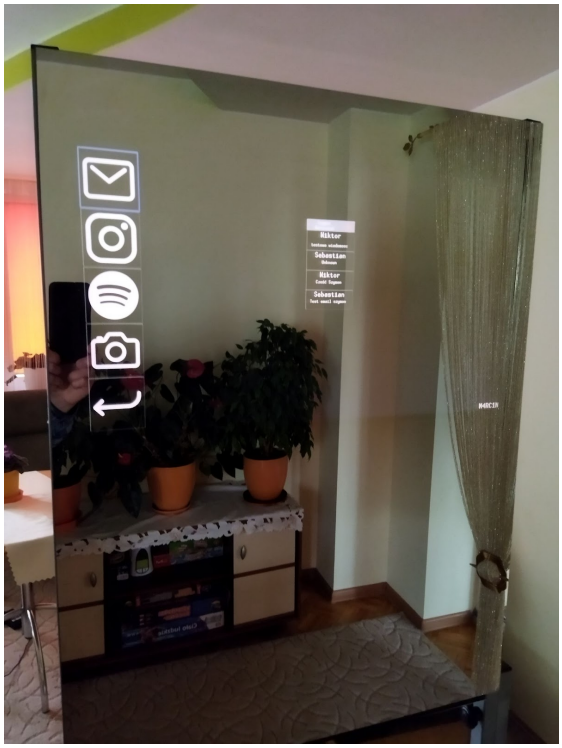
2. **Funkcje z dostępem do internetu:**



Email

Bezpośredni dostęp do poczty elektronicznej to również dobre rozwiązanie dla ludzi sukcesu oraz ludzi co wciąż do niego dążą. Maile otrzymujemy również w czasie, kiedy śpimy. Dzięki tej funkcji przed wyjściem z domu nie ominie nas żadna ważna wiadomość.

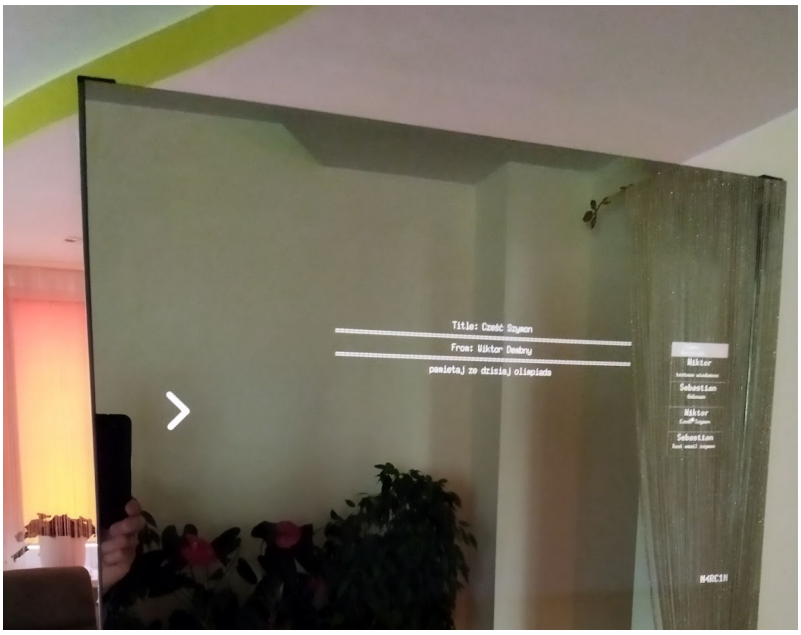




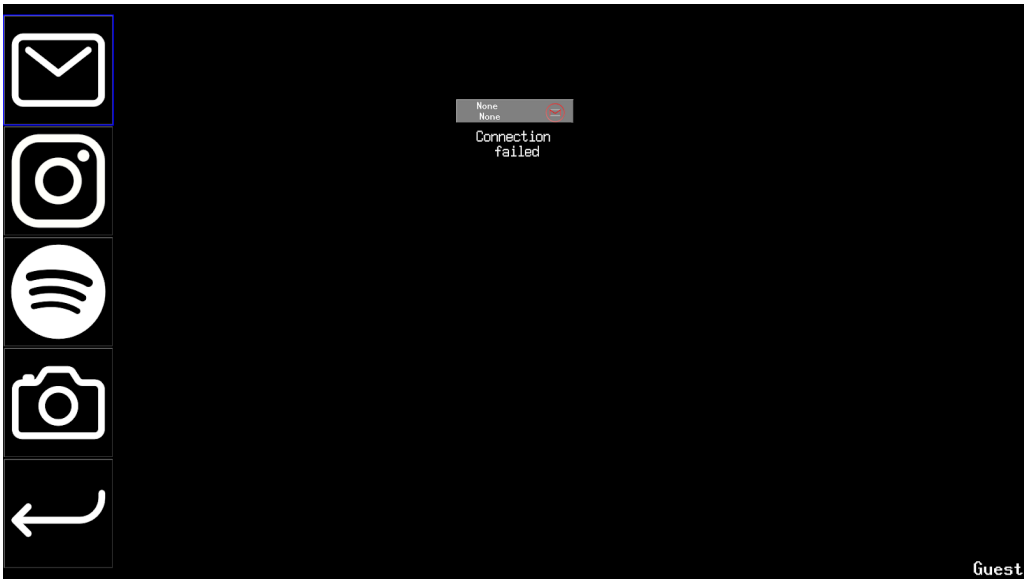
Gdy chcemy odczytać zawartość maila wystarczy kliknąć na interesujący nas nagłówek.

Tak prezentuje się mail w całej swojej okazałości, który samoistnie po przeczytaniu zniknie z lustra.

```
                        Title: Cześć Szymon
========================================================================
                        From: Wiktor Dembny
========================================================================
                    pamietaj ze dzisiaj olimpiada
```

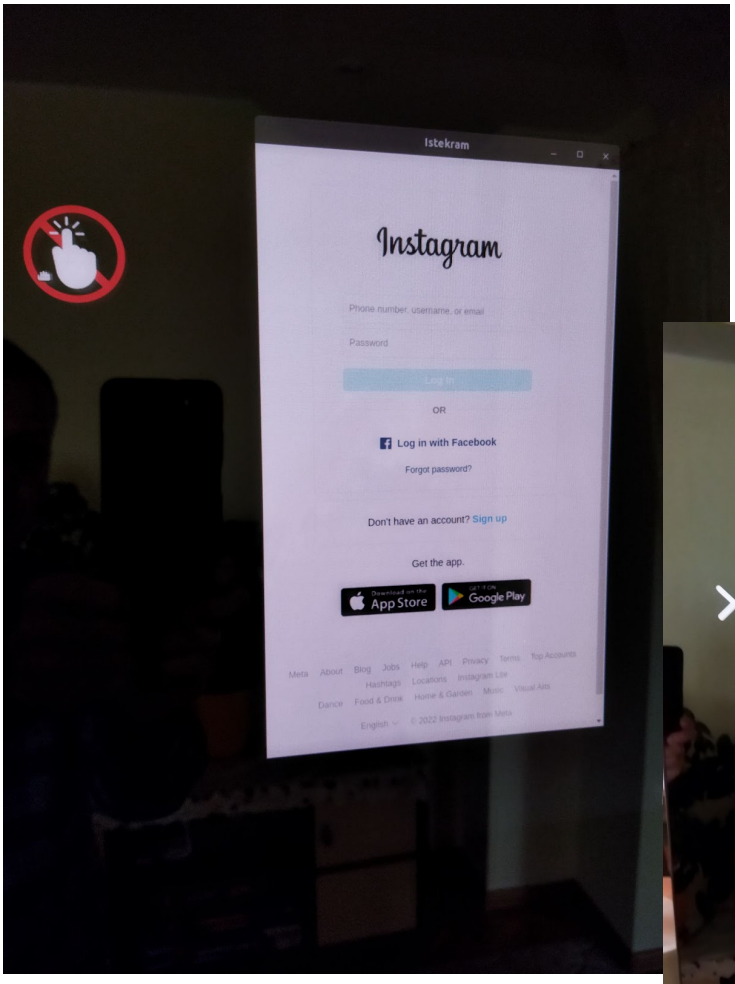


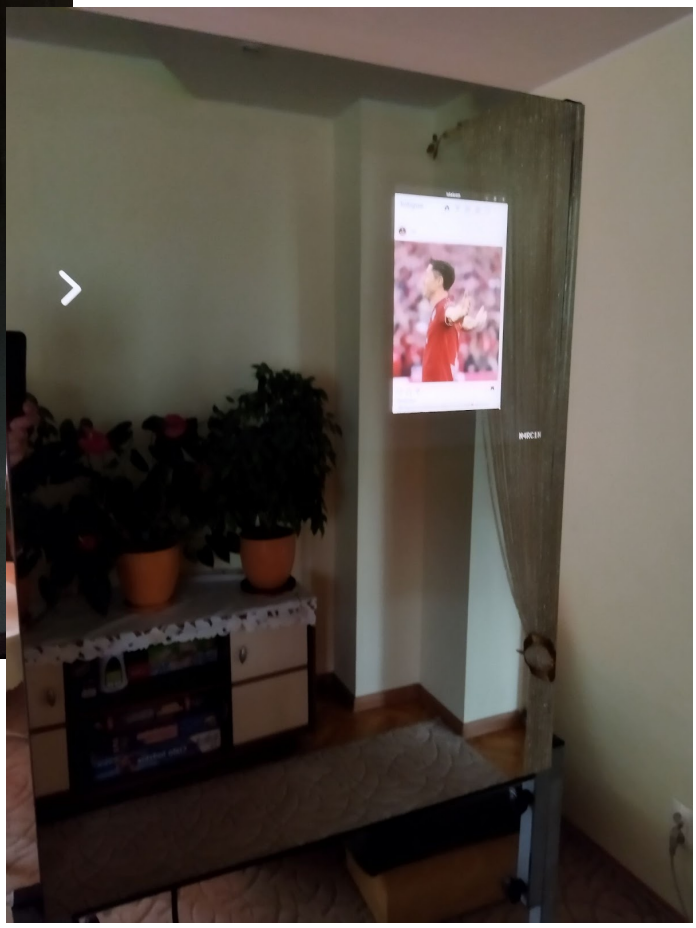
Oczywiście, jeśli nie mamy założonego konta, bądź też nie podaliśmy danych potrzebnych do jego działania, wyświetli się informacja o błędzie połączenia.



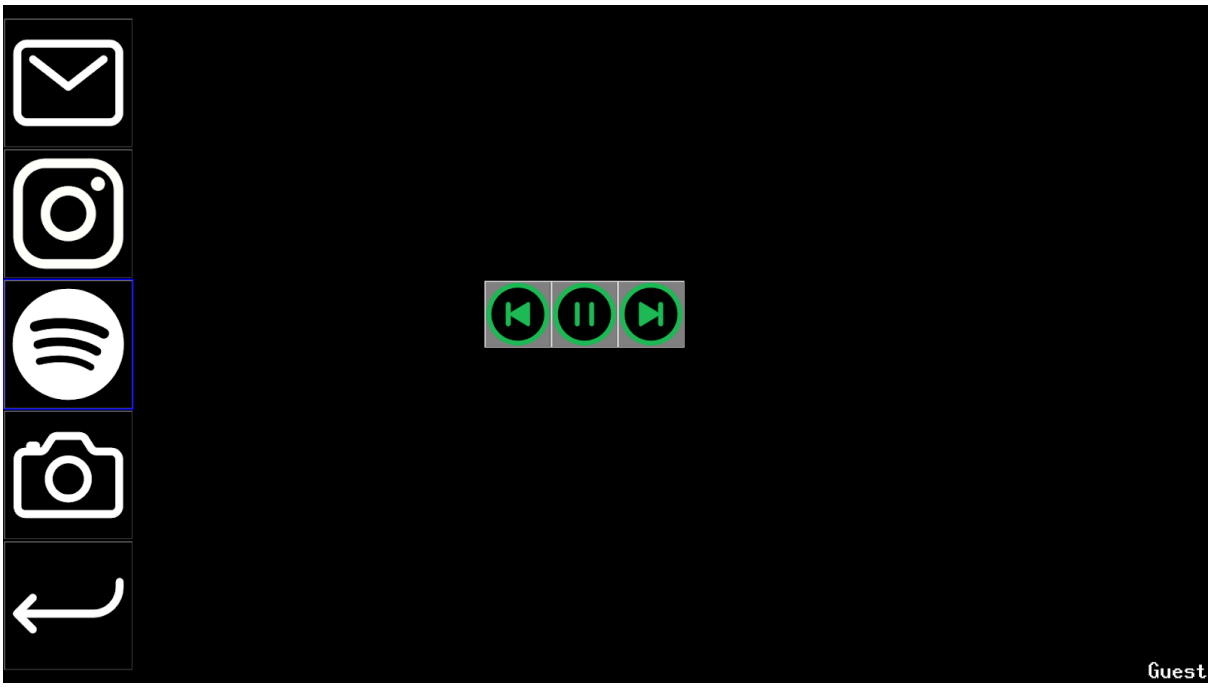
Instagram

Bycie na bieżąco w obecnym świecie jest niezwykle istotne, a może to zapewnić funkcja wyświetlania instagrama. Dzięki sztucznej inteligencji, posty są automatycznie przesuwane w odpowiednim tempe, aby bez problemu dało się przeczytać i obejrzeć zawartość.



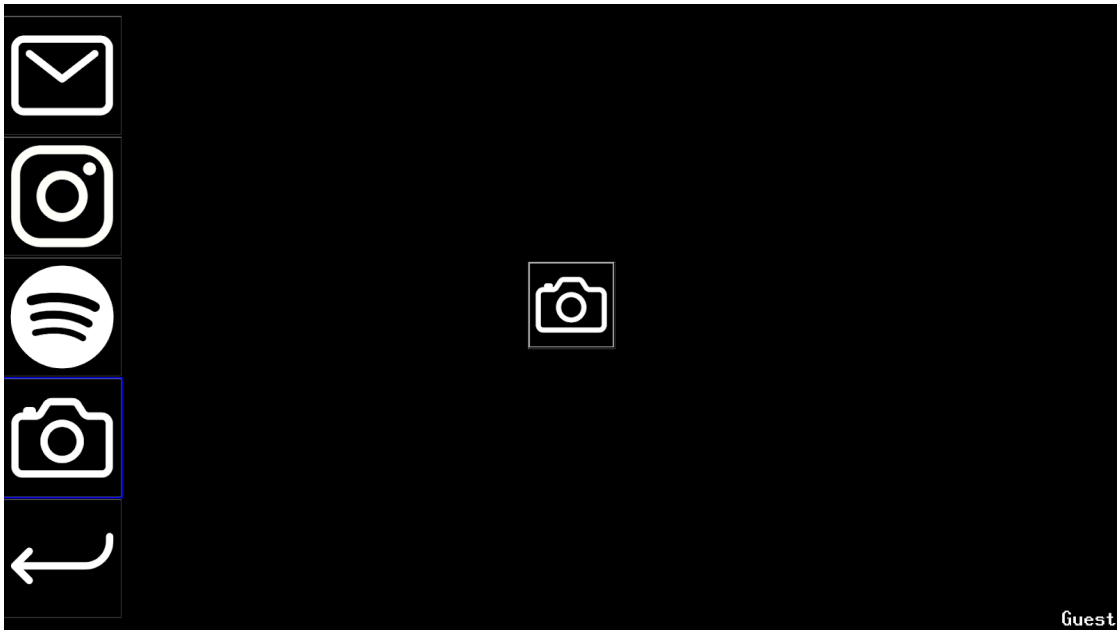


Spotify
Muzyka to nieodłączny element życia wielu ludzi, a włączenie jej podczas porannej rutyny może podnieść poziom własnego samopoczucia. Dzięki temu rozwiązaniu milej będzie rozpocząć dzień.



Ta funkcja, jak każda inna, jest bardzo intuicyjna. Przycisk po lewej stronie cofa utwór do poprzedniej, środkowy ją zatrzymuje, a ten po prawej przewija do przodu.

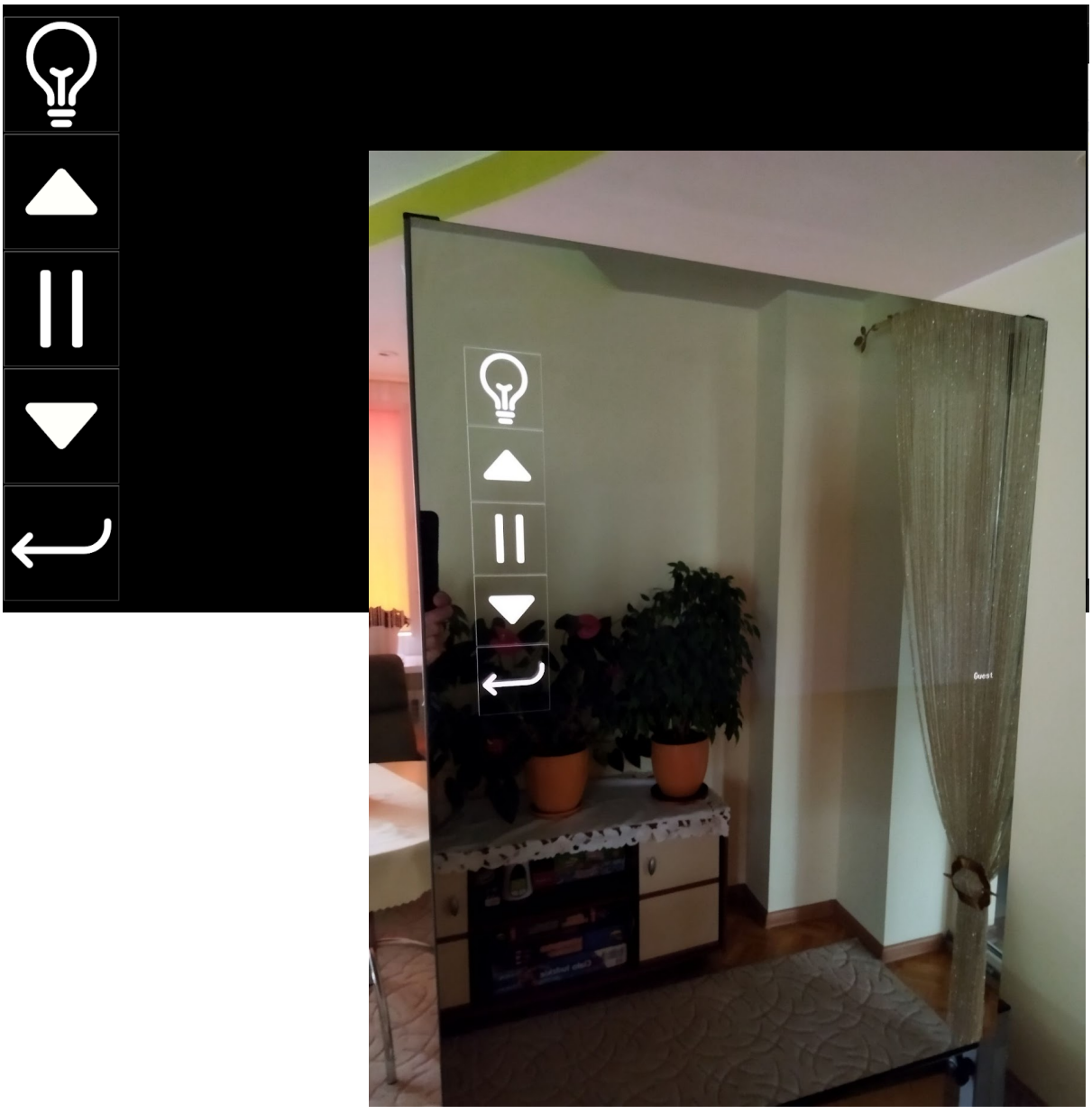
Zdjęcia
Robienie zdjęć to przydatna funkcja. Jest skierowana do tych, którzy swoim pięknym wyglądem chcą się podzielić z innymi w mediach społecznościowych.
Zdjęcia te można pobrać lub usunąć z poziomu aplikacji internetowej bądź mobilnej.



Po kliknięciu na ikonkę zdjęcia, pojawi nam się na lustrze kolejny znak ikoniczny przedstawiający aparat, który zostanie z nami nawet po wyłączeniu wszelkich pasków zadań. Zabieg ten ma na celu ułatwić jak i przyspieszyć czas wykonywania zdjęć osobom które często korzystają z tejże funkcji. Teraz aby wykonać zdjęcie należy kliknąć nową ikonkę, uśmiechnąć się oraz poczekać 5 sekund aż zdjęcie zostanie wykonane. Oczywiście w trakcie robienia fotografii wszystkie aktywne funkcje ‘znikają’, następnie pojawia się zegar który wyświetla odliczający czas potrzebny na ustawienie się (5 sekund), gdy czasomierz wyświetli “0”, zdjęcie automatycznie zostanie wykonane, natomiast fotografia wyświetli się na parę sekund na lustrze. Pod sam koniec wszystkie wcześniej aktywne funkcje wtórnie pojawiają się, a zdjęcie, które już teraz można pobrać za pomocą aplikacji na telefon czy też strony www, trafia do naszej bazy danych.

3. **Sterowanie żaluzjami i światłem w pokoju**

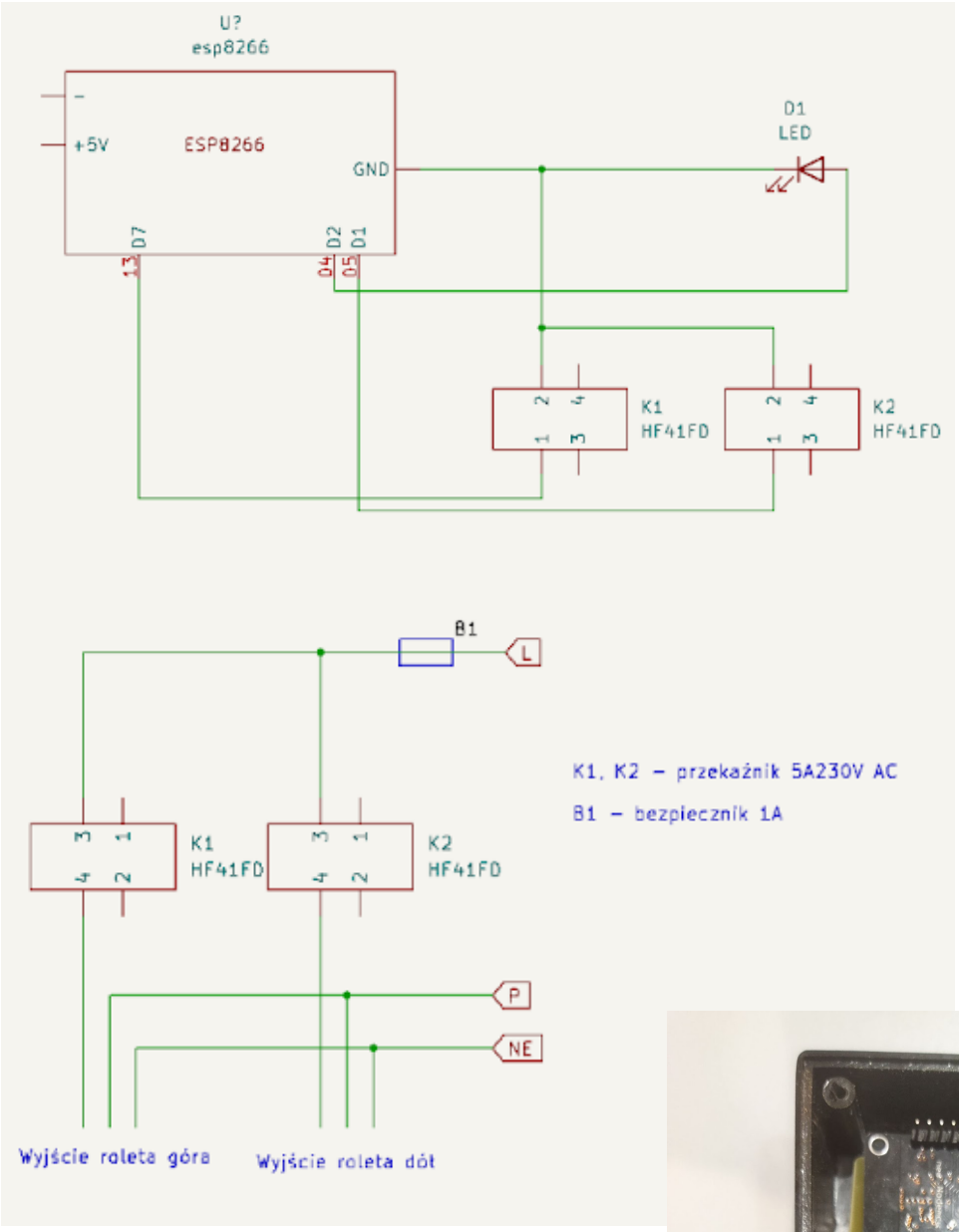
Do lustra dołączony jest bezprzewodowy moduł wykonawczy. Pozwala on na załączanie i wyłączanie oświetlenia w pokoju oraz płynne sterowanie roletami zewnętrznymi.

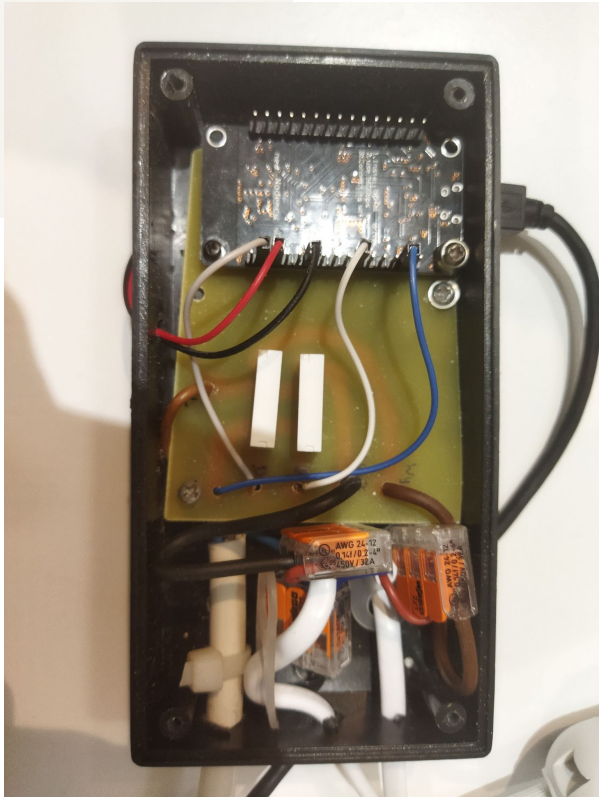


- Ikona żarówki włącza bądź wyłącza światło w pokoju,
- Strzałka w górę inicjuje podnoszenie rolety,
- Ikona pauzy powoduje zatrzymanie rolety,
- Analogicznie strzałka w dół inicjuje jej zamykanie.

Film przedstawiający sterowanie roletami i oświetleniem:
https://youtu.be/9aH6aj4QlVw

Schemat sterownika do rolet oraz włącznika światła.



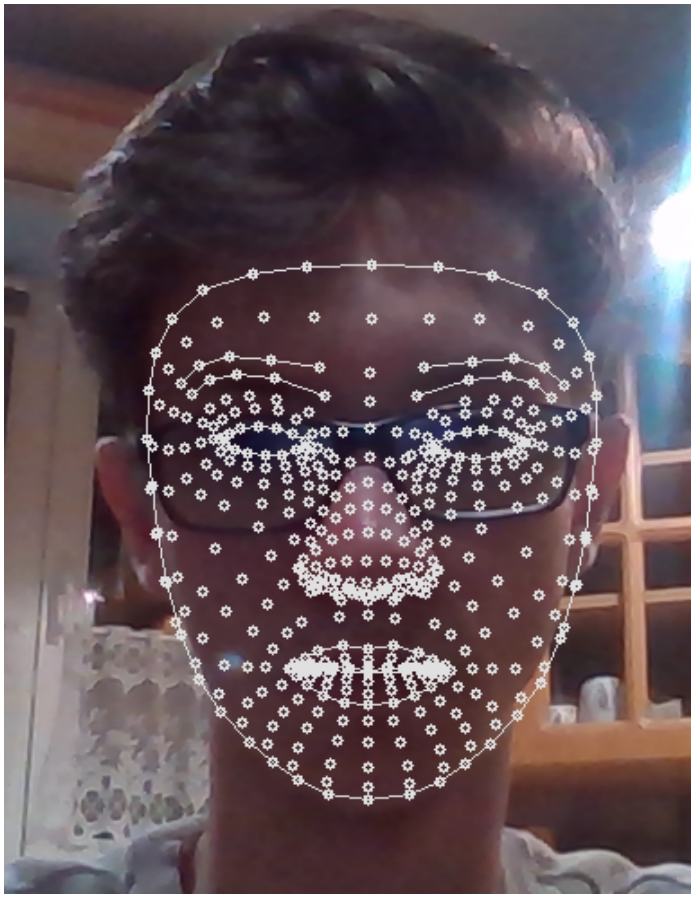


Sztuczna Inteligencja

Nazwa “Inteligentne lustro” nie wzięła się z niczego. Nasz projekt wykorzystuje sztuczną inteligencję, a dokładniej sieci neuronowe do trzech rzeczy:

1. Wykrywanie oraz rozpoznawanie twarzy,
2. Rozpoznawanie gestów oraz śledzenie położenia dłoni,
3. Wykrywania następnego rozpoczęcia postu na instagramie.

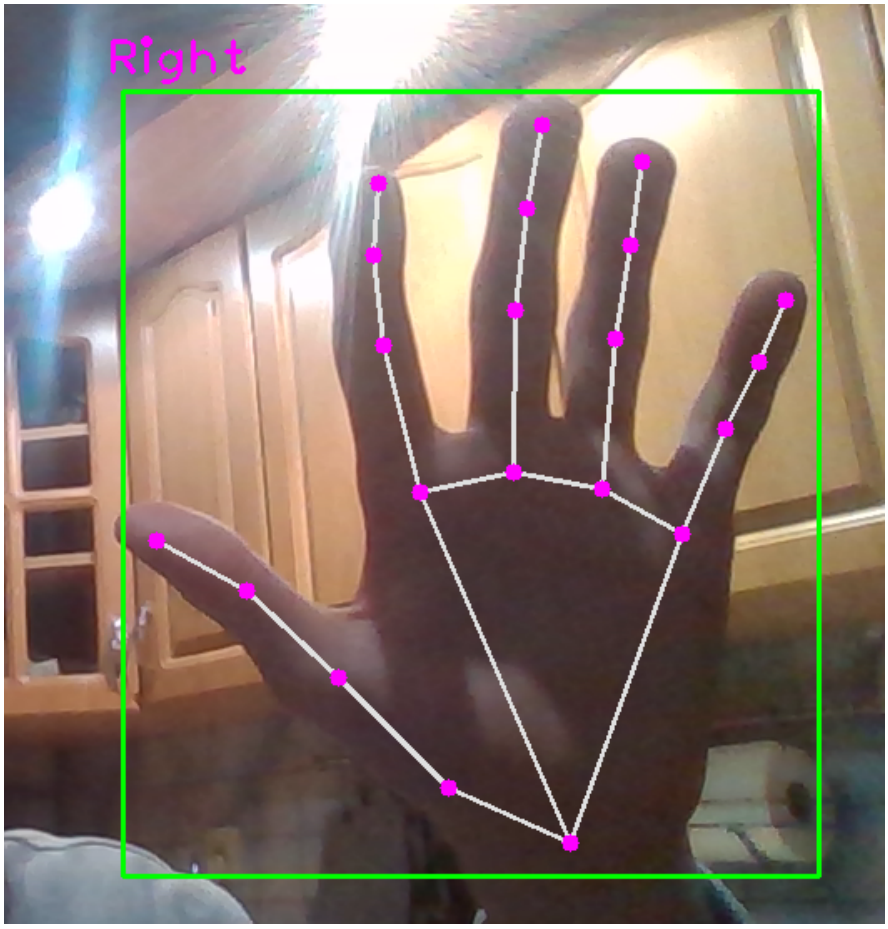
1. **Wykrywanie oraz rozpoznawanie twarzy:**
   Jedna z najważniejszych funkcji naszego projektu, działanie jej możliwe jest między innymi dzięki bibliotece OpenCv2 , która pozwala nam na naukę, dobrze radzącej sobie z naszym wyzwaniem, sieci neuronowej.



Film demonstrujący działanie rozpoznawania twarzy:
https://youtu.be/PzGrcY8rfqE

2. **Rozpoznawanie gestów oraz śledzenie położenia dłoni**

   Również jedna z ważniejszych funkcji naszego lustra, dzięki tej sieci neuronowej jesteśmy w stanie sterować kursorem w nowoczesny sposób.



3. **Wykrywanie następnego rozpoczęcia postu na instagramie.**

   Funkcja na pewno bardzo podobająca się w szczególności młodzieży, dzięki niej, lustro wyręcza nas w przewijaniu instagrama do następnego posta co przekłada się na zaoszczędzony czas.

Film przedstawiający całokształt głównej aplikacji lustra:
https://youtu.be/t3IonDM8ono

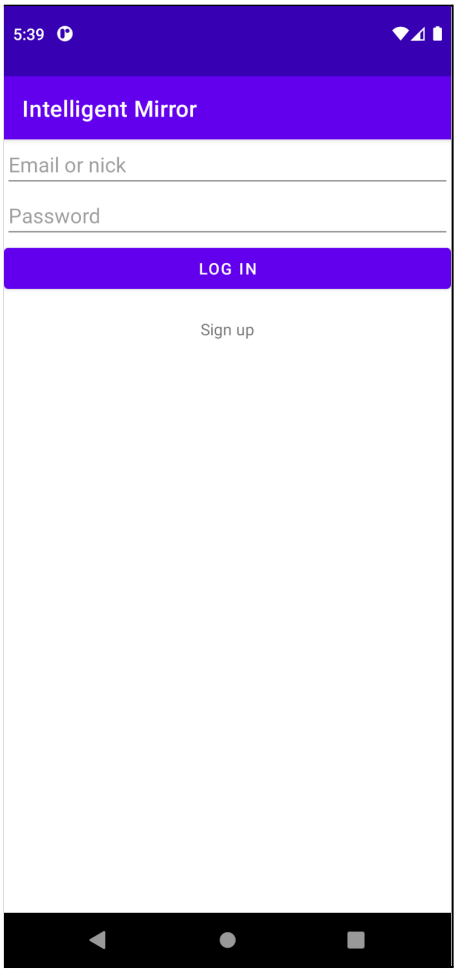
Intelligent Mirror – aplikacja mobilna

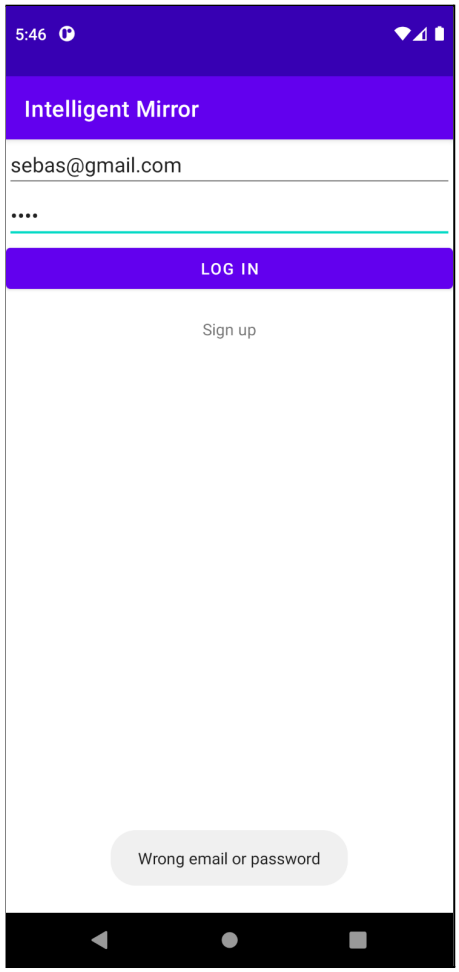
Narzędzie do automatyzacji kompilacji: Gradle
Minimalna wersja systemu Android: Lollipop 5.0
Wykorzystane języki: Java, XML
Kod źródłowy: https://github.com/JediSebas/InteligentMirror

MainActivity
Startowa strona aplikacji. Zawiera:

- 2 pola EditText potrzebne do wprowadzenia danych do logowania
- Przycisk Button, który pozwala na zalogowanie
- Napis TextView służący jako link do strony SignupActivity

Po kliknięciu przycisku dostajemy informacje czy wprowadziliśmy wszystkie dane oraz czy są prawidłowe. Jeśli wszystko się zgadza oraz mamy dostęp do sieci WiFi powinniśmy zostać zalogowani i przejść do HomeFragment.

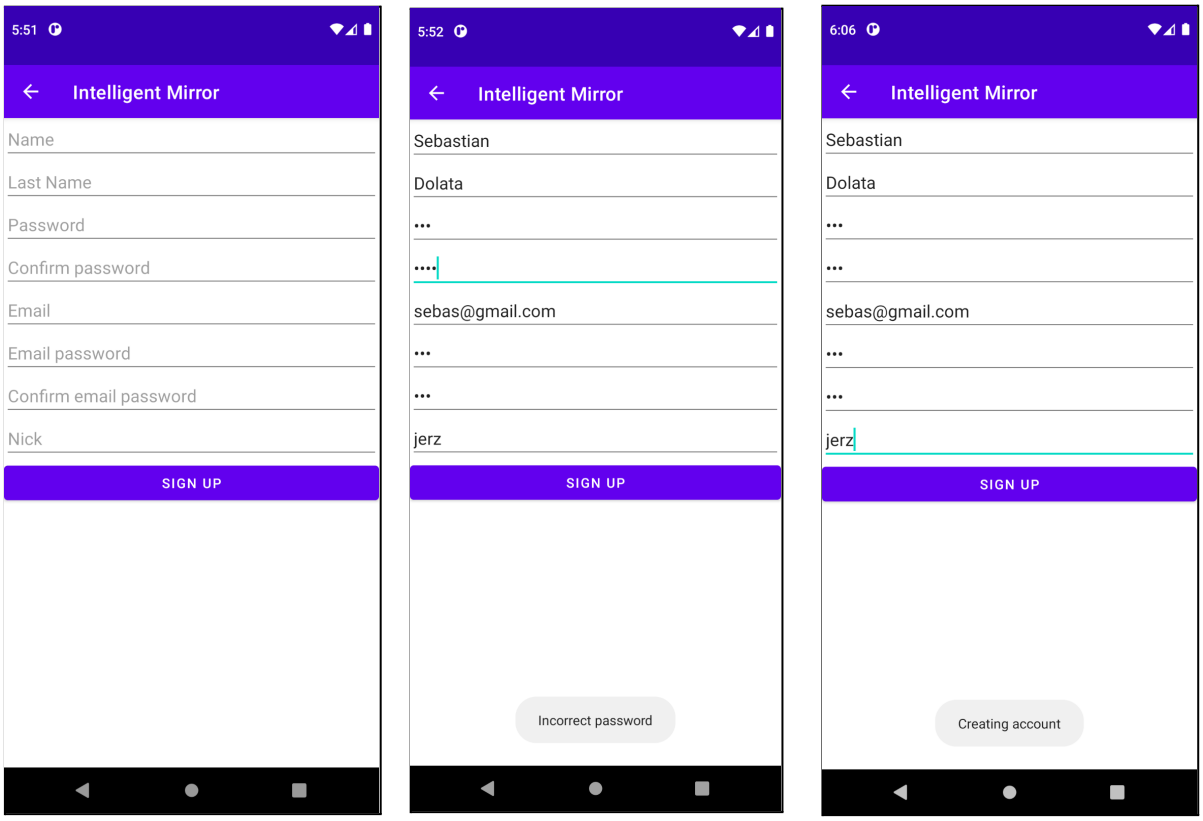




SignupActivity
Strona rejestracji. Zawiera:

- 8 pól EditText potrzebnych do założenia konta (imię, nazwisko, hasło, potwierdź hasło, hasło do emaila, potwierdź hasło do emaila, nick)
- Przycisk Button do utworzenia konta
- Strzałki powrotu do MainActivity w lewym górnym rogu

Naciśnięcie przycisku powoduje sprawdzenie czy wszystkie pola zostały wypełnione, następnie czy wartości pól ‘hasło` i `potwierdź hasło` są identyczne. Potem następuje weryfikacja, czy konto o podanym adresie e mail bądź nicku już istnieje. Jeśli wszystkie kryteria zostaną spełnione, dochodzi do utworzenia profilu.



MenuActivity

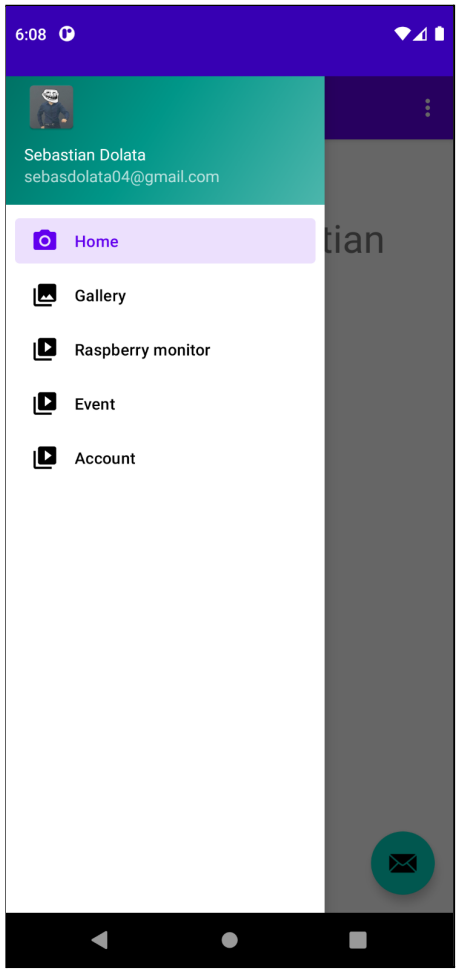
Pliki potrzebne do stworzenia menu, aby poruszać się po aplikacji. Zawiera:

- AppBarMenu, czyli wysuwane menu z lewego boku
- Pole Settings w prawym górnym rogu, aby wejść w ustawienia aplikacji (SettingsActivity)

W skład AppBarMenu wchodzi:

- HomeFragment
- GalleryActivity
- RaspberryFragment
- EventFragment
- AccountActivity
- Pasek z imieniem i nazwiskiem twórcy aplikacji, e mailem aby się z nim skontaktować oraz zdjęciem profilowym na serwisie GitHub (na tej stronie również jest cały kod źródłowy aplikacji)

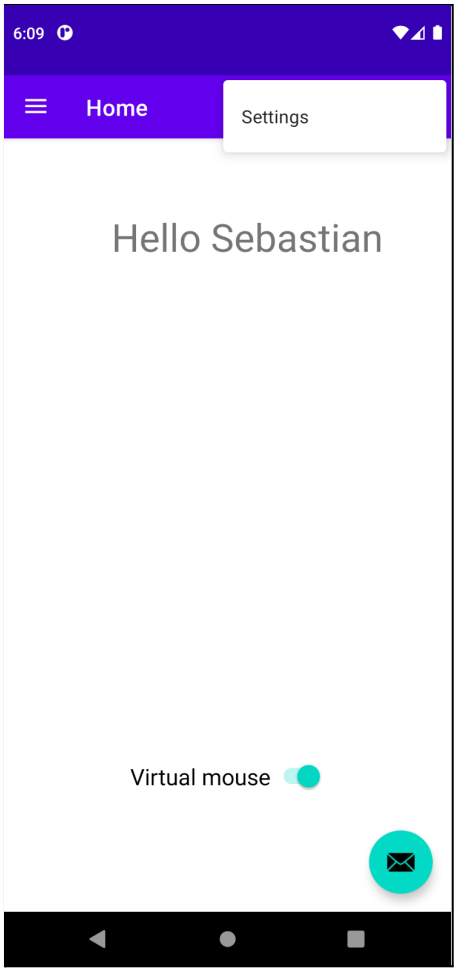
Aby otworzyć AppBarMenu należy nacisnąć ikonę przedstawiającą trzy poziome kreski w lewym górnym rogu. W przypadku pola Settings trzy kropki w pionowej linii znajdujące się w prawym górnym rogu.

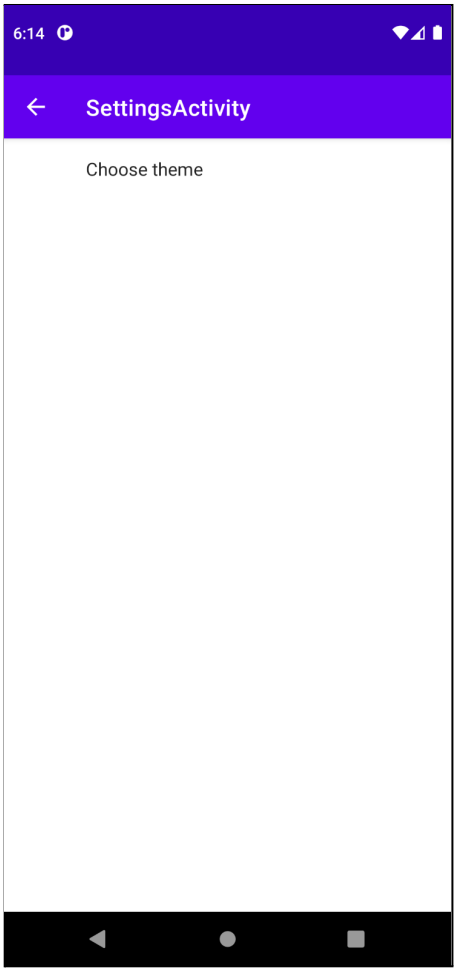


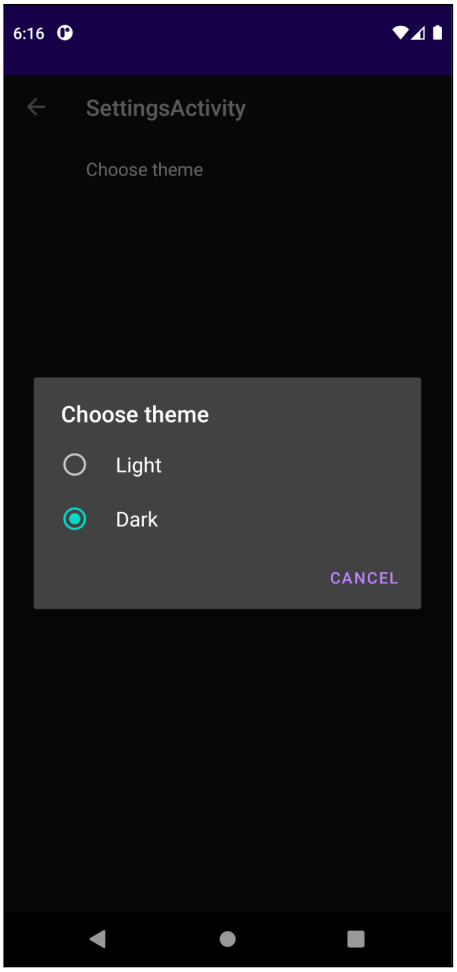
SettingsActivity

Strona ustawień aplikacji. Zawiera:

- ListPreference posiadające 2 Radio Buttony do zmiany motywu aplikacji
- Strzałkę powrotu do fragmentu, z którego weszliśmy w ustawienia





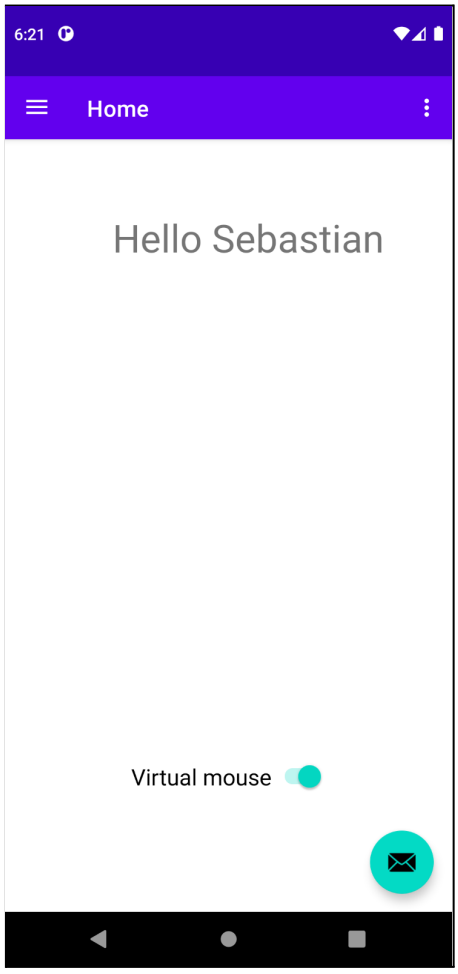



```java
public static class SettingsFragment extends PreferenceFragmentCompat {
    @Override
    public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {
        setPreferencesFromResource(R.xml.root_preferences, rootKey);
        ListPreference listPreference = findPreference( key: "theme_preference");
        listPreference.setOnPreferenceChangeListener((preference, newValue) -> {
            CharSequence text = listPreference.getEntry();
            String id = listPreference.getValue();

            switch(Integer.parseInt(id)) {
                case 2:
                    AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
                    MainActivity.changeTheme("light");
                    break;
                case 1:
                    AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
                    MainActivity.changeTheme("dark");
                    break;
            }

            return true;
        });
    }
}
```

HomeFragment

Pierwsza widoczna strona po zalogowaniu. Zawiera:

- Napis TextView z przywitaniem zalogowanego użytkownika
- Switch (znany również jako ToggleButton lub przełącznik) służący do włączania/wyłączania wirtualnej myszy na Inteligentnym Lustrze
- AppBarMenu oraz pole Settings

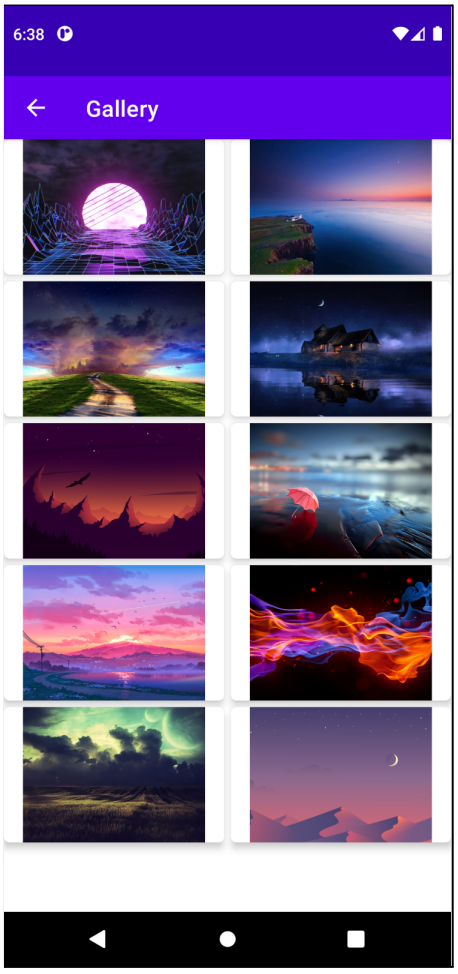



```
hello.setText("Hello " + Loggeduser.name);

JDBCvm jdbcvm = new JDBCvm( mode: 1);
jdbcvm.t.start();

try {
    jdbcvm.t.join();
} catch (InterruptedException e) {
    e.printStackTrace();
}

vmSwitch.setChecked(status == 1);

vmSwitch.setOnCheckedChangeListener((compoundButton, b) -> {
    JDBCvm jdbcVm = new JDBCvm( mode: 0);
    if (b) {
        jdbcVm.setVal((byte) 1);
    } else {
        jdbcVm.setVal((byte) 0);
    }
    jdbcVm.t.start();
});
```

```
private class JDBCvm implements Runnable {

    Thread t;
    byte val;
    int mode;

    JDBCvm(int mode) {
        t = new Thread( target: this);
        this.mode = mode;
    }

    void setVal(byte val) { this.val = val; }

    @Override
    public void run() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("DRIVER STILL WORKS BTW");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        if (mode == 0) {
            updateQuery();
        } else if (mode == 1) {
            getQuery();
        }
    }
```

GalleryActivity

Strona ze zdjęciami zrobionymi przez lustro. Dzięki dynamicznemu GridView (widokowi siatki) zdjęcia ukazują się w dwukolumnowej tabeli. Zdjęcia zapisywane w ścieżce /var/www/html/mirror są udostępniane przez serwer Apache jako link URL. Następnie korzystając z biblioteki *Universal Image Loader* są one wyświetlane. Kliknięcie w obraz powoduje przejście do DownloadActivity.




```
ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(getBaseContext())
        .threadPriority(Thread.NORM_PRIORITY - 2)
        .denyCacheImageMultipleSizesInMemory()
        .diskCacheFileNameGenerator(new Md5FileNameGenerator())
        .diskCacheSize(50 * 1024 * 1024)
        .tasksProcessingOrder(QueueProcessingType.LIFO)
        .writeDebugLogs()
        .build();

imageLoader = ImageLoader.getInstance();
imageLoader.init(config);

for (int i=0; i<imageUrls.size(); i++) {
    imageItems.add(new ImageItem());
}

GridViewAdapter adapter = new GridViewAdapter( context: this, imageItems);
gridView.setAdapter(adapter);

GettingBitmap gettingBitmap = new GettingBitmap();
gettingBitmap.t.start();

gridView.setOnItemClickListener((adapterView, view, i, l) -> {
    ImageItem item = (ImageItem) adapterView.getItemAtPosition(i);
    Intent intent = new Intent( packageContext: GalleryActivity.this, DownloadActivity.class);
    DownloadActivity.bitmap = item.getImage();
    DownloadActivity.fileName = item.getImageName();
    startActivity(intent);
});
```

```
public class GridViewAdapter extends ArrayAdapter {

    public GridViewAdapter(@NonNull Context context, ArrayList<ImageItem> imageItems) {
        super(context,  resource: 0, imageItems);
    }

    @Override
    public View getView(int position, @Nullable View convertView, @Nullable ViewGroup parent) {

        if (convertView == null) {
            convertView = LayoutInflater.from(getContext()).inflate(R.layout.grid_item, parent,  attachToRoot: false);
        }

        ImageItem imageItem = (ImageItem) getItem(position);
        ImageView imageView = convertView.findViewById(R.id.gridImage);

        imageView.setImageBitmap(imageItem.getImage());

        return convertView;
    }

}
```

DownloadActivity

Strona, na której widać cały obraz oraz można go usunąć, pobrać lub jednocześnie pobrać i usunąć. Zawiera:

- ImageView z wybranym zdjęciem
- 3 przyciski Button – delete (usuń), download (pobierz), download and delete (pobierz i usuń)

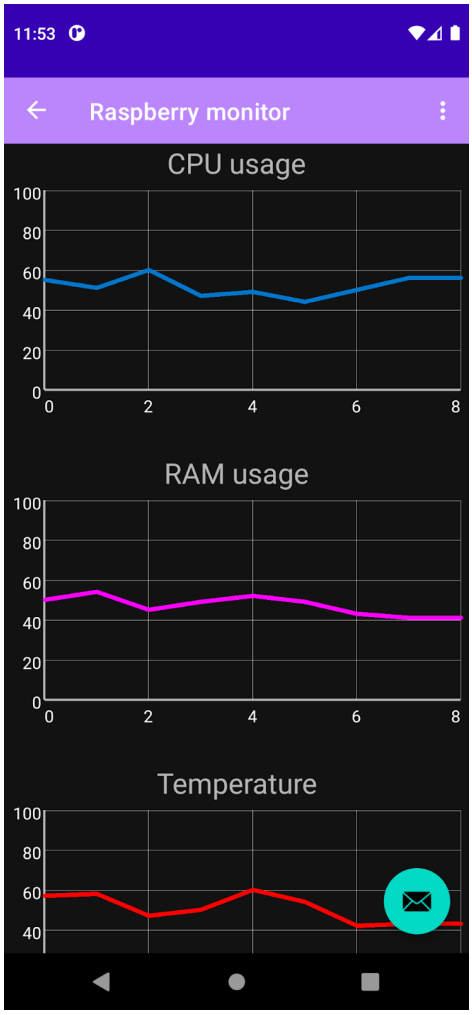
Pobieranie dokonuje się przez bibliotekę *Universal Image Loader.* Umożliwia ona pobieranie zdjęć posiadając tylko link URL. Usuwanie zachodzi poprzez protokół FTP i bibliotekę *ftp4j* i usunięcie rekordu z bazy danych MySQL.


```
private class FTPDelete implements Runnable {

    Thread t;

    FTPDelete() { t = new Thread( target: this); }

    @Override
    public void run() {
        FTPClient ftpClient = new FTPClient();
        try {
            ftpClient.connect(Loggeduser.ip, ConnectionData.PORT);
            ftpClient.login(ConnectionData.USER, ConnectionData.PASS);

            ftpClient.changeDirectory(ConnectionData.DIRECTORY_WITH_IMAGES);
            ftpClient.setType(FTPClient.TYPE_BINARY);
            ftpClient.setPassive(true);
            ftpClient.noop();

            ftpClient.deleteFile( path: ConnectionData.DIRECTORY_WITH_IMAGES + fileName);
        } catch (FTPIllegalReplyException | IOException | FTPException e) {
            e.printStackTrace();
        } finally {
            try {
                if (ftpClient.isConnected()) {
                    ftpClient.logout();
                    ftpClient.disconnect( sendQuitCommand: true);
                }
            } catch (IOException | FTPException | FTPIllegalReplyException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

```
private void download() {
    String directory = Environment.getExternalStorageDirectory().getAbsolutePath() + "/DCIM/Mirror";
    File myDir = new File(directory);
    myDir.mkdirs();

    File file = new File (myDir, fileName);
    if (file.exists ()) file.delete ();
    try {
        FileOutputStream out = new FileOutputStream(file);
        bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 90, out);
        out.flush();
        out.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        MediaScannerConnection.scanFile(getBaseContext(),
                new String[]{directory},  mimeTypes: null,
                (path, uri) -> {
                    Log.i( tag: "ExternalStorage",  msg: "Scanned " + path + ":");
                    Log.i( tag: "ExternalStorage",  msg: "-> uri=" + uri);
                });
    }
}
```

RaspberryFragment

Strona z wykresami kontrolującymi pracę podzespołów Raspberry Pi, na którym działa lustro. Wykorzystano bibliotekę *jjoe64 GraphView*.

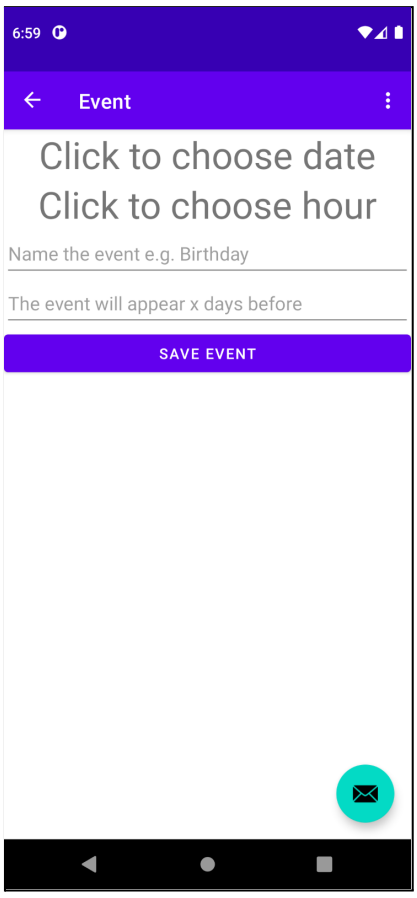


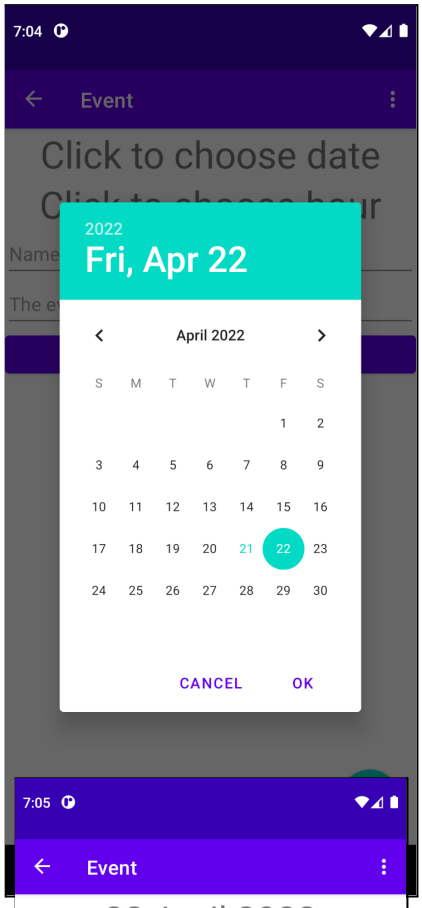

```xml
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/cpuTv"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="CPU usage"
            android:textAlignment="center"
            android:textSize="24sp" />

        <com.jjoe64.graphview.GraphView
            android:id="@+id/line_graph_cpu"
            android:layout_width="match_parent"
            android:layout_height="200dp" />

        <TextView
            android:id="@+id/ramTv"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="RAM usage"
            android:textAlignment="center"
            android:layout_marginTop="30dp"
            android:textSize="24sp" />

        <com.jjoe64.graphview.GraphView
            android:id="@+id/line_graph_ram"
            android:layout_width="match_parent"
            android:layout_height="200dp" />
```

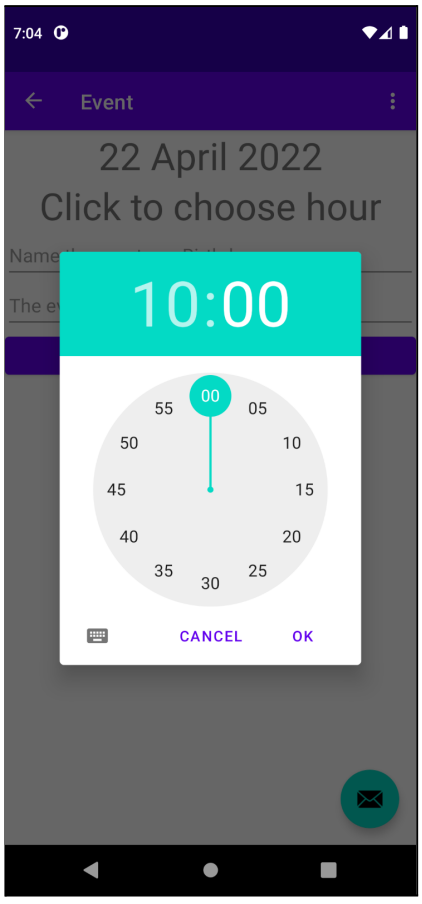
```java
void levelUpTable() {
    for (int j=0; j<4; j++) {
        graphArray[j].removeAllSeries();
        for (int i = 0; i < 8; i++) {
            pointArray[j][i] = new DataPoint(i, pointArray[j][i + 1].getY());
        }
        LineGraphSeries<DataPoint> rePaint = new LineGraphSeries<>(pointArray[j]);
        rePaint.setThickness(10);
        switch (j) {
            case 1:
                rePaint.setColor(Color.MAGENTA);
                break;
            case 2:
                rePaint.setColor(Color.RED);
                break;
            case 3:
                rePaint.setColor(Color.YELLOW);
                break;
        }
        graphArray[j].addSeries(rePaint);
    }
}
```

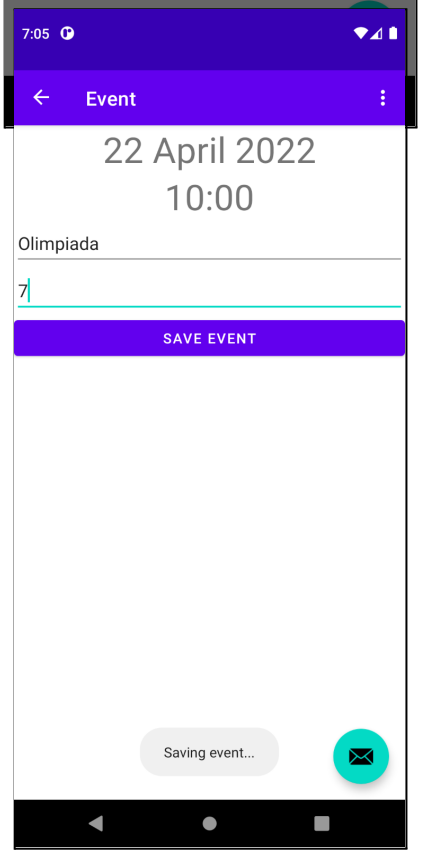
EventFragment
Strona, dzięki której możemy tworzyć nowe wydarzenia, które będą wyświetlane na lustrze.
Zawiera:

- 2x napis TextView z datą i godziną
- 2x pole EditText z nazwą wydarzenia i ilością dni, od kiedy wydarzenie ma być widoczne
- Przycisk Button aby zapisać wydarzenie





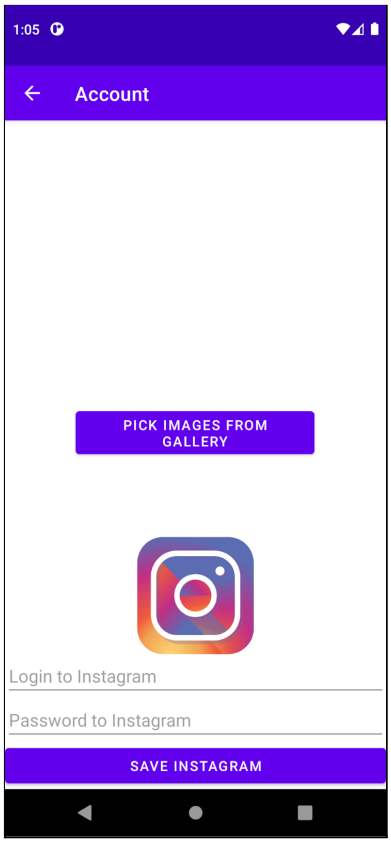


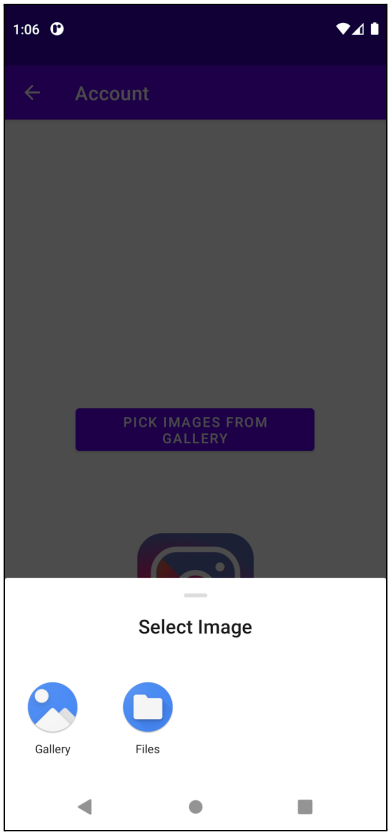


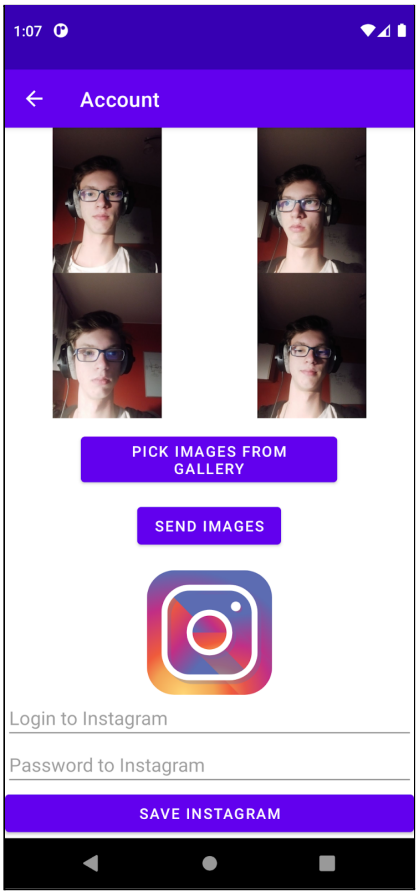
AccountActivity

Strona, do zarządzania kontem użytkownika. Zawiera:

- Przycisk Button do wybrania zdjęć z galerii
- Przycisk Button do przesłania wybranych zdjęć z galerii
- ImageView z ikonami loga Instagrama
- 2x pole EditText do dodania danych logowania do Instagrama
- Przycisk Button do zapisania tych danych







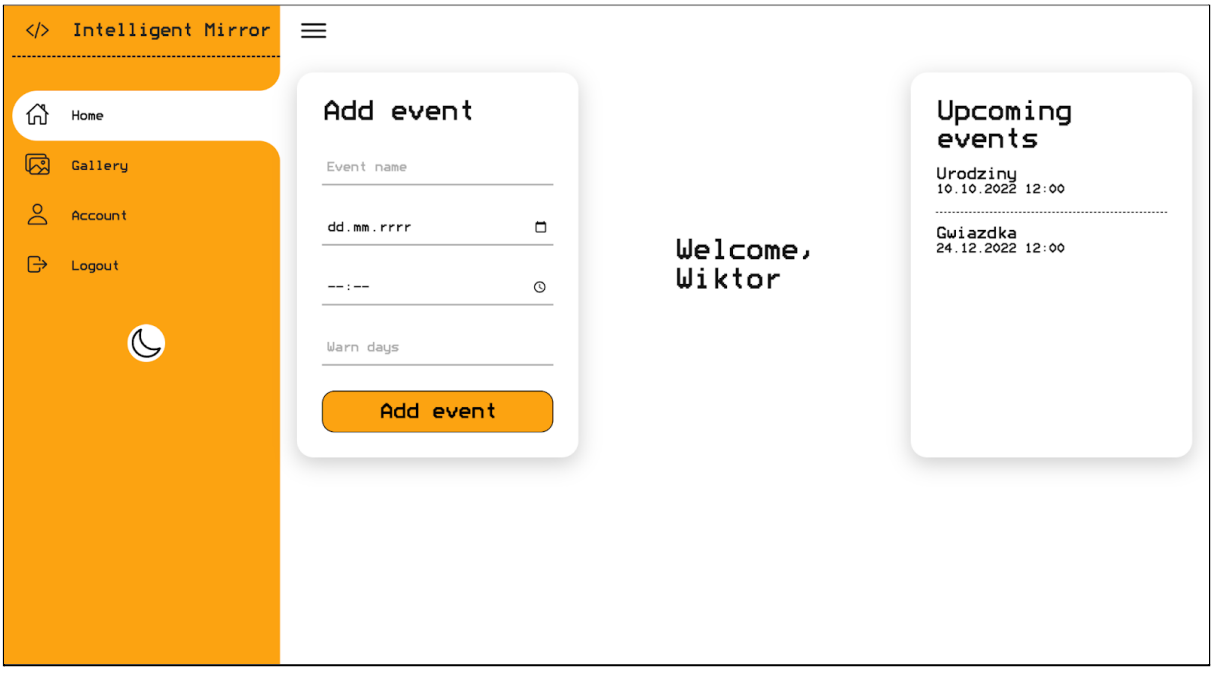
Aplikacja wymusza, aby wybrać co najmniej 4 zdjęcia, w przeciwnym razie przycisk do przesłania się nie pojawi.

```
private class JDBCSaveIgSp implements Runnable {

    Thread t;
    String login, password;

    JDBCSaveIgSp(String login, String password) {
        t = new Thread( target: this);
        this.login = login;
        this.password = password;
    }

    @Override
    public void run() {
        String QUERY = "UPDATE `user` SET `instagram_login` = '" + login + "', `instagram_password` = '" + password + "' WHERE `user`.`id` = " + Loggeduser.id + ";";
        try {
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("DRIVER STILL WORKS BTW");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        try {
            Connection conn = DriverManager.getConnection(ConnectionData.DB_URL, ConnectionData.USER, ConnectionData.PASS);
            Statement stmt = conn.createStatement();
            stmt.executeUpdate(QUERY);
        } catch (SQLException throwables) {
            System.out.println("HERE IS SOMETHING WRONG");
            throwables.printStackTrace();
        }
    }
}
```

Intelligent Mirror – aplikacja webowa

Narzędzie do automatyzacji kompilacji: Maven
Wykorzystane języki: Java (Spring Boot, Hibernate), HTML, CSS, JS
Integracja szablonu strony ze Spring Bootem: Thymeleaf
Kod źródłowy: https://github.com/JediSebas/Webmirror

Strona internetowa służy do tworzenia kont, planowania wydarzeń, przeglądania wykonanych na lustrze zdjęć i podawania danych logowania do aplikacji Instagram.

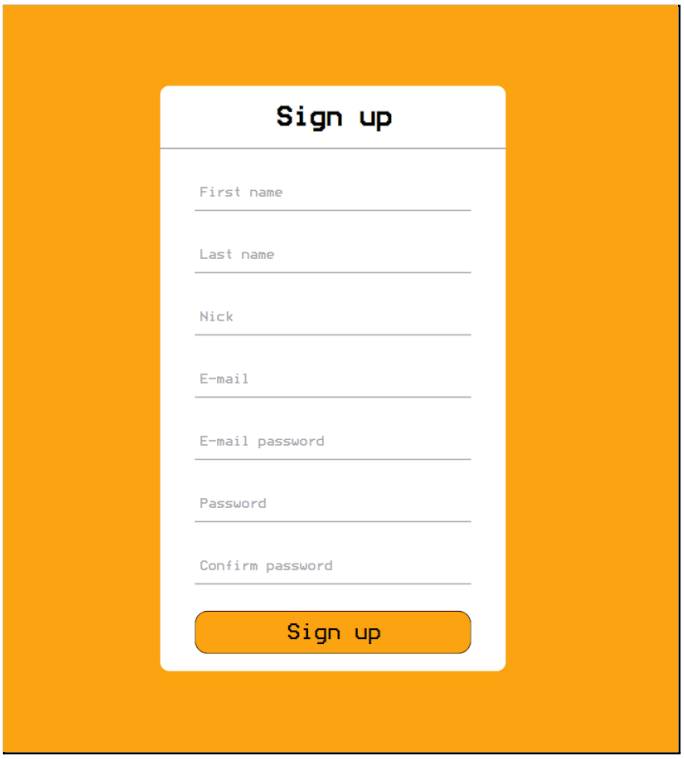


Front-end
Kolory:

#FCA311	#262322	#292929	#141414	#E5E5E5

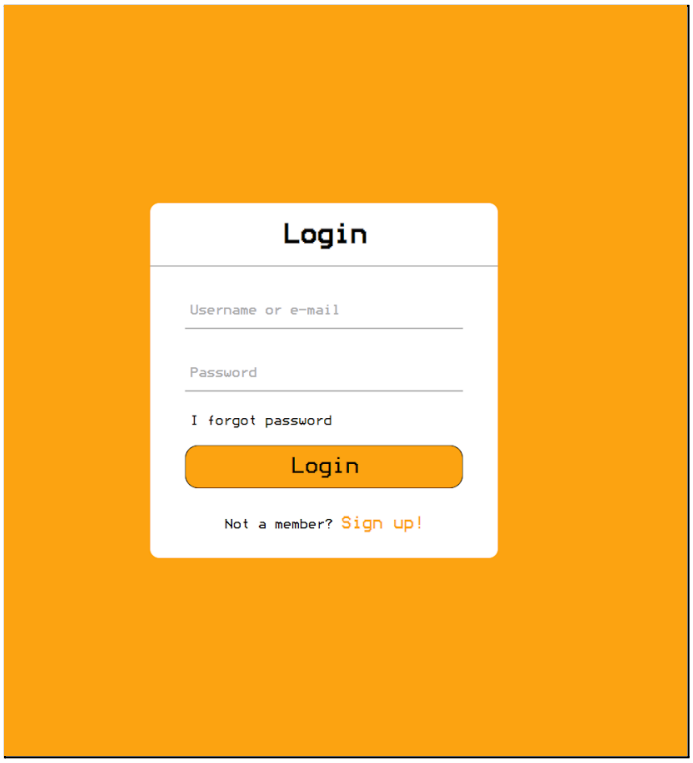
Czcionka
Modeseven

Ikony
Ikony open source - Ionicons
https://ionic.io/ionicons



Rejestracja

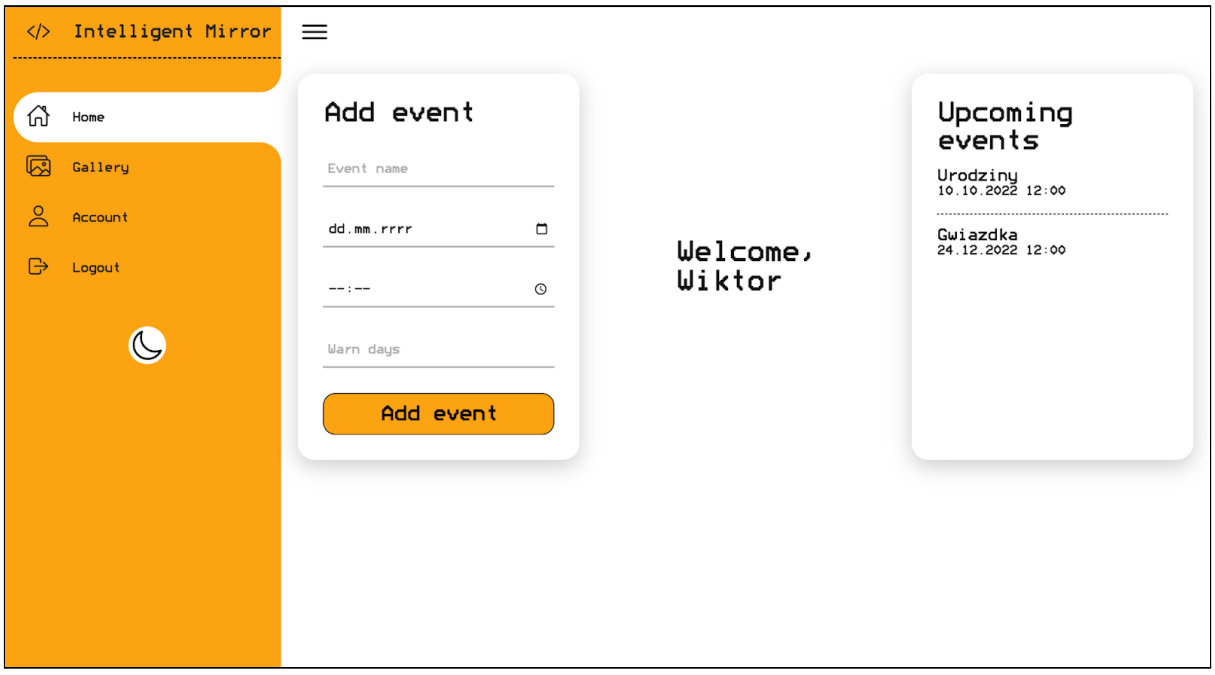
Aby lustro było w stanie nas rozpoznawać, należy założyć konto i podać takie informacje jak: imię, nazwisko, nick, e-mail, hasło do emaila



Logowanie

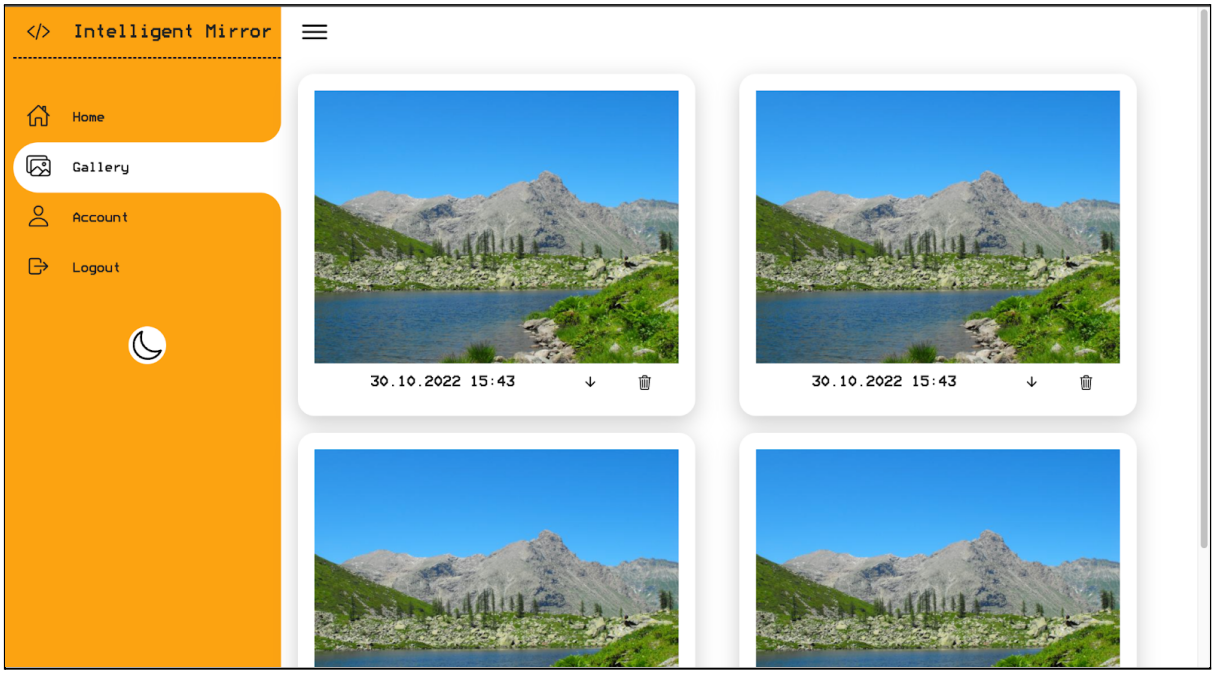
Jeśli chcemy dokonać konfiguracji naszego konta, bądź też pobrać zdjęcia które zrobiliśmy za pomocą lustra, logujemy się na już istniejące konto podając Nick lub email konta oraz hasło do niego.

Strona główna



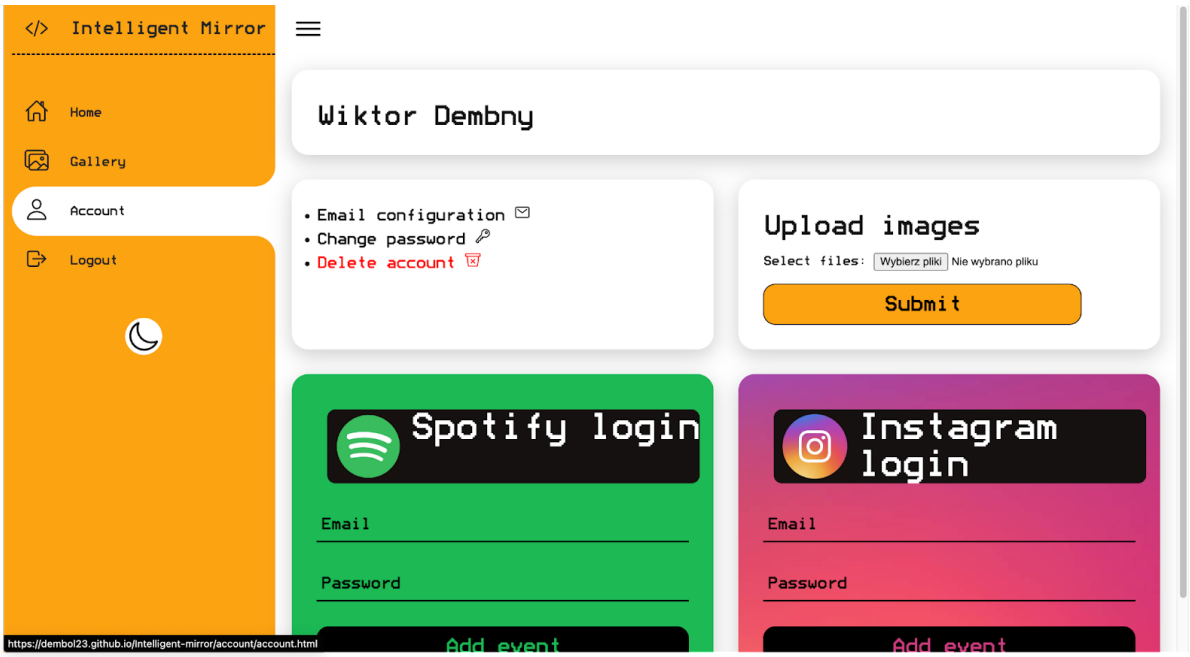
Strona powitalna po zalogowaniu umożliwia nam zaplanowanie nowych wydarzeń, zobaczenie już utworzonych oraz ich usunięcie.

Galeria



W tym miejscu pojawiają się wszystkie wykonane na lustrze zdjęcia. Każde z nich można pobrać lub usunąć z bazy danych za pomocą przycisków poniżej. Strona jest responsywna i dopasowuje zdjęcia do rozmiaru okna przeglądarki.

Konto



W tej zakładce możliwe jest zmiana maila, hasła, usunięcie swojego konta, a także dodanie danych logowania do Instagrama oraz Spotify. W kafelku *Upload images* użytkownik dodaje zdjęcia służące do przyszłego rozpoznawania twarzy

Back-end

Wykorzystane zależności (dependencies):

- Spring Data JPA
- Thymeleaf
- Spring Web
- Spring Boot DevTools
- MySQL Driver
- Spring Integration

Architektura aplikacji:
Klasa User – odtworzenie zawartości bazy danych, aby Hibernate mógł dokonywać w niej zmian. (analogicznie klasa Event)

```
@Entity
@Table(name = "user", schema = "user")
public class User implements Serializable {
    4 usages
    @Id
    @GeneratedValue(strategy= GenerationType.AUTO,generator="native")
    @GenericGenerator(name = "native",strategy = "native")
    @Column(name = "id", nullable = false)
    private Long id;
    4 usages
    private String name;
    4 usages
    private String lastname;
    4 usages
    private String password;
    4 usages
    private String email;
```

Interfejs UserRepository – tworzy się w nim metody, dzięki którym dokonuje się operacji na bazie danych. Można stosować 2 języki zapytań: oryginalny SQL i zmodyfikowany JPQL czerpiący z obiektowej filozofii Javy. (analogicznie interfejs EventRepository)

```
2 usages  JediSebas +1
@Repository
public interface UserRepository extends JpaRepository<User, Long> {

    3 usages  JediSebas
    @Query("SELECT u FROM User u WHERE u.email = ?1")
    Optional<User> findUserByEmail(String email);

    2 usages  JediSebas
    @Query("SELECT u FROM User u WHERE u.nick = ?1")
    Optional<User> findUserByNick(String nick);

    1 usage  JediSebas
    @Query("SELECT password from User u WHERE u.email = ?1")
    String findPasswordByEmail(String email);
```

Klasa UserService – pozwala na wykorzystanie interfejsu UserRepository do dalszych działań. (analogicznie klasa EventService)

```
2 usages  JediSebas +1
@Service
public class UserService {

    19 usages
    public final UserRepository userRepository;

    JediSebas
    @Autowired
    public UserService(UserRepository userRepository) { this.userRepository = userRepository; }

    1 usage  JediSebas
    public List<User> getUsers() { return userRepository.findAll(); }

    1 usage  JediSebas
    public int addNewUser(User user) {
        Optional<User> userOptional = userRepository.findUserByEmail(user.getEmail());
        Optional<User> userOptional2 = userRepository.findUserByNick(user.getNick());
        if (userOptional.isPresent() || userOptional2.isPresent()) {
            if (userOptional.isPresent() && userOptional2.isPresent()) {
                return 1;
            } else if (userOptional.isPresent()) {
                return 2;
            } else {
                return 3;
            }
        }
        userRepository.save(user);
        return 0;
    }
```

Klasa UserContoller – umożliwia wykorzystanie serwisów (UserService i EventService) do wyświetlania zawartości na stronie oraz modyfikowanie zawartości bazy danych. Adnotacje *@GetMapping* oraz *@PostMapping* świadczą o wykorzystywaniu protokołu *HTTP* w komunikacji.

```
@Controller
public class UserController {

    9 usages
    public final UserService userService;
    4 usages
    public final EventService eventService;

    2 usages
    public static String ip = "192.168.0.50";

    JediSebas
    @Autowired
    public UserController(UserService userService, EventService eventService) {
        this.userService = userService;
        this.eventService = eventService;
    }

    JediSebas
    @GetMapping("/user")
    @ResponseBody
    public List<User> getUsers() {
        return userService.getUsers();
    }

    JediSebas
    @GetMapping("/signup")
    public String signupView(Model model) {
        model.addAttribute( attributeName: "user", new User());
        return "signup";
    }
```

Thymeleaf integruje kod HTML z Springiem dzięki przedrostkom *th:* przy niektórych atrybutach. Na poniższym zdjęciu fragment formularza do rejestracji.

```
<form th:action="@{/register}" th:object="${user}" method="post">
    <div class="text-field">
        <input id="name" name="name" type="text" th:field="*{name}" required>
        <span></span>
        <label>First name</label>
    </div>
    <div class="text-field">
        <input id="lastname" name="lastname" type="text" th:field="*{lastname}" required>
        <span></span>
        <label>Last name</label>
    </div>
    <div class="text-field">
        <input id="nick" name="nick" type="text" th:field="*{nick}" required>
        <span></span>
        <label>Nick</label>
    </div>
    <div class="text-field">
        <input id="email" name="email" type="email" th:field="*{email}" required>
        <span></span>
        <label>E-mail</label>
    </div>
    <div class="text-field">
        <input id="emailpassword" name="emailpassword" type="password" th:field="*{emailpassword}" required>
        <span></span>
        <label>E-mail password</label>
    </div>
    <div class="text-field">
        <input id="password" name="password" type="password" th:field="*{password}" required>
        <span></span>
        <label>Password</label>
```

Tutaj fragment galerii.

```
<div class="tileBox">
    <div class="tile" th:each="picture: ${pictures}">
        <img th:src="${picture.url}" alt="date">
        <div class="description">
        <div class="date"></div>
        <form th:action="@{/downloadPicture/{picture.name}(picture.name=${picture.name})}" method="post"><
        <form th:action="@{/deletePicture/{picture.name}(picture.name=${picture.name})}" method="post"><bu
    </div>
</div>
```

```
JediSebas
@GetMapping("/home/gallery/{name}")
public String galleryView(Model model, @PathVariable String name) {
    model.addAttribute( attributeName: "helloname", name);
    if (LoggedUser.isLogged && name.equals(LoggedUser.name)) {
        List<String> pictures = userService.getPictures(LoggedUser.id);
        System.out.println(pictures);
        List<String> picturesFinal = new ArrayList<>();
        for (String str: pictures) {
            picturesFinal.add("http://"+ip+"/mirror/" + str);
        }
        List<Picture> pictureList = new ArrayList<>();
        for (int i=0; i<pictures.size(); i++) {
            pictureList.add(new Picture(pictures.get(i), picturesFinal.get(i)));
        }
        model.addAttribute( attributeName: "pictures", pictureList);
        return "gallery";
    } else {
        return "redirect:/index/";
    }
}
```

Zostały dodane zabezpieczenia, aby nikt niepowołany nie dostał się do naszych danych bez zalogowania.

Struktura bazy danych

Nazwa: mirror
Tabele:
user

	#	Nazwa	Typ	napisów	Atrybuty	Null	domyślne	Komentarze	Dodatkowo
☐	1	id 🔑	int(11)			Nie	*Brak*		AUTO_INCREMENT
☐	2	name	text	utf8_polish_ci		Nie	*Brak*		
☐	3	lastname	text	utf8_polish_ci		Nie	*Brak*		
☐	4	password	text	utf8_polish_ci		Nie	*Brak*		
☐	5	email	text	utf8_polish_ci		Nie	*Brak*		
☐	6	emailpassword	text	utf8_polish_ci		Nie	*Brak*		
☐	7	nick	text	utf8_polish_ci		Nie	*Brak*		
☐	8	instagram_login	text	utf8_polish_ci		Tak	*NULL*		
☐	9	instagram_password	text	utf8_polish_ci		Tak	*NULL*		
☐	10	time_event	bit(1)			Nie	*Brak*		
☐	11	weather_event	bit(1)			Nie	*Brak*		
☐	12	gmail_event	bit(1)			Nie	*Brak*		
☐	13	quote_event	bit(1)			Nie	*Brak*		
☐	14	calendar_event	bit(1)			Nie	*Brak*		
☐	15	photos_event	bit(1)			Nie	*Brak*		
☐	16	instagram_event	bit(1)			Nie	*Brak*		
☐	17	time_x	int(11)			Nie	*Brak*		
☐	18	weather_x	int(11)			Nie	*Brak*		
☐	19	gmail_x	int(11)			Nie	*Brak*		
☐	20	quote_x	int(11)			Nie	*Brak*		
☐	21	calendar_x	int(11)			Nie	*Brak*		
☐	22	photos_x	int(11)			Nie	*Brak*		
☐	23	time_y	int(11)			Nie	*Brak*		
☐	24	weather_y	int(11)			Nie	*Brak*		
☐	25	gmail_y	int(11)			Nie	*Brak*		
☐	26	quote_y	int(11)			Nie	*Brak*		
☐	27	calendar_y	int(11)			Nie	*Brak*		
☐	28	photos_y	int(11)			Nie	*Brak*		
☐	29	spotify_x	int(11)			Nie	*Brak*		
☐	30	spotify_y	int(11)			Nie	*Brak*		
☐	31	spotify_event	bit(1)			Nie	*Brak*		
☐	32	vm	bit(1)			Nie	*Brak*		

event

	#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo
☐	1	id	int(11)			Nie	*Brak*		AUTO_INCREMENT
☐	2	userid	int(11)			Nie	*Brak*		
☐	3	name	text	utf8_polish_ci		Nie	*Brak*		
☐	4	howlong	int(11)			Nie	*Brak*		
☐	5	date	datetime			Nie	*Brak*		

pictures

	#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarze	Dodatkowo
☐	1	id	int(11)			Nie	*Brak*		AUTO_INCREMENT
☐	2	userid	int(11)			Nie	*Brak*		
☐	3	name	text	utf8_polish_ci		Nie	*Brak*		

Serwer Apache

Katalog: /var/www/html/mirror/
Są tam zapisywane zdjęcia zrobione przez lustro, aby potem mieć do nich dostęp za pomocą linku.

Serwer FTP

Katalog: /IntelligentMirror/data/
Za pomocą protokołu FTP przesyłamy zdjęcia do tego katalogu. Zdjęcia te służą do wykrywania naszej twarzy przez lustro.

Specyfikacja:

- Pobór mocy: 50 W
- Całkowity rozmiar: 60x60x180cm
- Rozmiar monitora: 24 cale
- Element sterujący: Raspberry Pi 4b
- Pamięć ram: 2 GB
- Pamięć twarda: 32 GB
- Procesor: Broadcom BCM2711 quad-core 64-bitowy

Nasz projekt lustra wykorzystuje 10 tysięcy linii back-endu oraz 3 tysiące front-endu. Był pisany przez okres półtora roku. Zajęło nam to ponad 1200 roboczogodzin. Program został napisany w ***całości*** *przez nas. Obejrzeć go w całej okazałości można na Githubie:*

- *Główna aplkacja: https://github.com/SzymonMarciniak/IntelligentMirror*
- *Aplikacja mobilna: https://github.com/JediSebas/InteligentMirror*
- *Strona www: https://github.com/JediSebas/Webmirror*