

Documentation: **Application Kivizz**

<b>I_Introduction</b>	<b>2</b>
<b>II_Installation et configuration</b>	<b>2</b>
<b>III_Utilisation</b>	<b>3</b>
<b>IV_Structure du code</b>	<b>3</b>
<b>V_Outils utilisés</b>	<b>3</b>

## I Introduction

L'application Kivizz a été développée dans le cadre d'une SAE, avec pour objectif de créer des applications communicantes. Kivizz propose une plateforme de quizz interactifs pour une expérience divertissante. Notre principal but était de fournir une application de quizz interactive sans prétention, simplement pour le plaisir des utilisateurs.

## II Installation et configuration

Pour le moment, notre application est uniquement disponible localement, mais nous prévoyons une publication future sur le Play Store. Pour arriver à cet objectif, il y aura plusieurs étapes nécessaires avant de déposer notre application:

-Création d'un Compte Développeur : nous devons créer un compte développeur sur le Google Play Console.

-Préparation de l'Application : nous allons devoir nous assurer que l'application est prête avec des captures d'écran attrayantes, une description claire et sans bugs majeurs.

-Génération d'un APK Signé : Nous utiliserons les outils Android pour créer un fichier APK sécurisé.

-Soumission sur le Google Play Console : Connexion à notre compte développeur android, téléversement de l'APK, remplissage des infos nécessaires et suivi des étapes de soumission.

-Validation et Approbation : Nous devons alors attendre que l'équipe de Google examine et approuve notre application.

-Promotion de l'Application : Nous commencerons à promouvoir notre application une fois approuvée.

-Suivi des Performances : Nous utiliserons des outils de suivi sur le Play Console pour voir comment notre application performe. Nous ferons des mises à jour régulières en fonction des retours des utilisateurs.

### III Utilisation

Nous allons maintenant vous donner un aperçu d'utilisation de notre application. Lorsque que l'on lance l'application, on se retrouve sur la page de création d'un compte.



Créer un compte

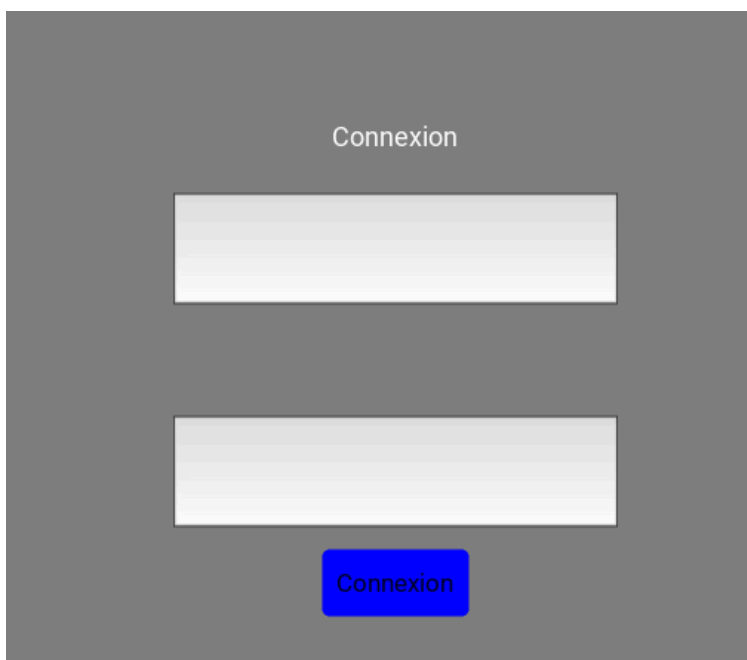
Utilisateur:

Une fois notre compte créé, on se dirige vers la page de connexion.

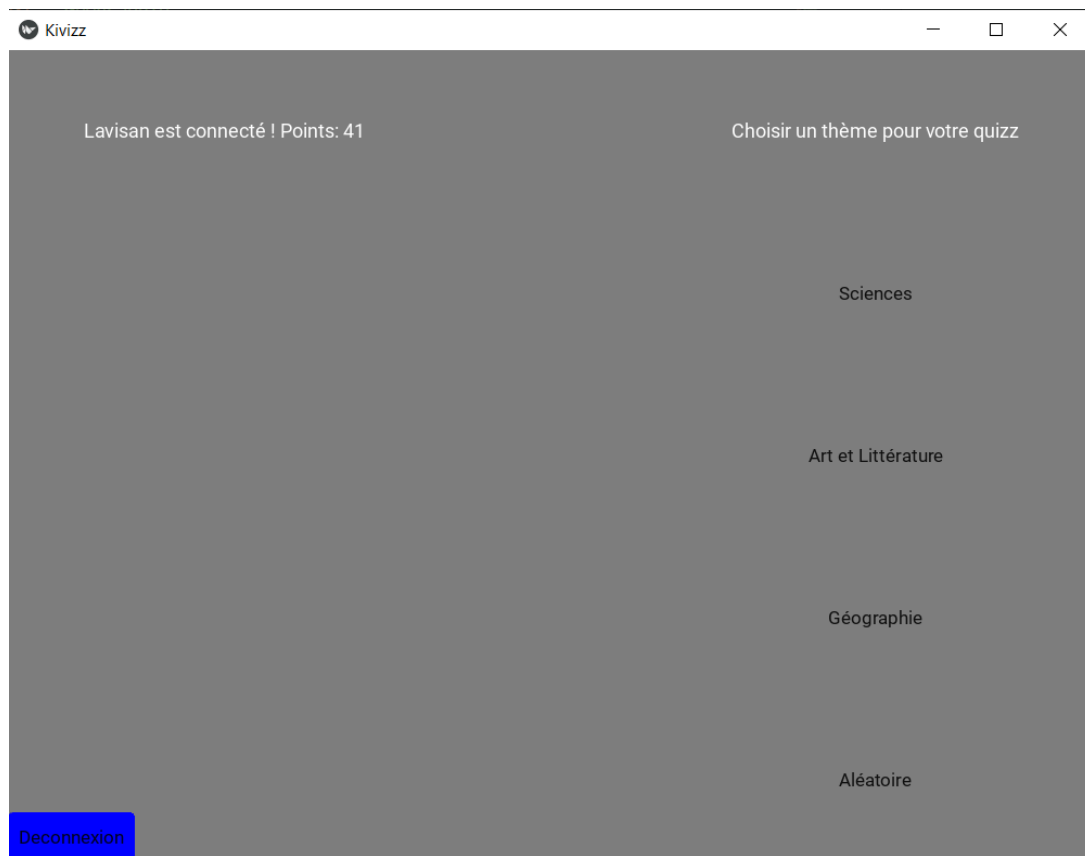


Page de connexion



Connexion

Lorsque l'utilisateur est connecté, son score est affiché sur la page d'accueil et il a le choix de réaliser des quizz sur différents thèmes, il peut aussi se déconnecter.



Pour chaque thème, il y a différents types de questions:

-des questions où l'utilisateur doit écrire sa réponse:



-des questions avec 4 propositions de réponses:

Quel est l'organe principal du système respiratoire?



Valider

Cerveau Foie Poumons Coeur

-des questions avec 2 propositions de réponses:

Quelle est la force qui maintient les objets sur la Terre?



Valider

Force centrifuge Gravité

Une fois le quizz terminé, le score de la page d'accueil s'actualise avec les points gagnés durant le quizz.

## IV Structure du code

### Architecture générale:

Le code utilise le langage KV, spécifique à Kivy, pour définir la structure de l'interface utilisateur.

Il crée une application de quizz avec plusieurs écrans gérés par ScreenManager.

```
KV = '''
ManageScreen:
    MenuScreen:
    GameScreen:
    LoggedScreen:
    SciencesScreen1:
    SciencesScreen2:
    SciencesScreen3:
    SciencesScreen4:
    SciencesScreen5:
```

Chaque écran est défini par une classe en python.

```
class SciencesScreen1(MDScreen):
```

Le contenu de chaque écran est défini dans la structure du langage KV:

```
<SciencesScreen1>:
    name: 'sciences'
    md_bg_color: 'grey'
    Label:
        text: "Quelle est la formule chimique de l'eau?"
        pos_hint: {'center_x': 0.5, 'center_y': 0.9}
    Image:
        source: "goutte.jpg"
        size_hint: 0.3, 0.3
        pos_hint: {'center_x': 0.5, 'center_y': 0.7}

    TextInput:
        id: sciencesun
        pos_hint: {'center_x': 0.5, 'center_y': 0.4}
        size_hint: 0.8, 0.2
```

## Modules et Classes :

Les modules importés incluent des éléments de l'interface utilisateur Kivy (App, Builder, ScreenManager, etc.), ainsi que des modules de KivyMD et Psycopg2 pour la gestion des écrans, des styles et la connexion à la base de données PostgreSQL.

```
from kivy.app import App
from kivy.lang import Builder
from kivy.uix.screenmanager import ScreenManager, Screen
from kivyMD.uix.screen import MDScreen
from kivy.uix.textinput import TextInput
from kivyMD.app import MDApp

import psycopg2
import time
```

## Connexion à la Base de Données :

Le code établit une connexion à une base de données PostgreSQL locale avec des identifiants spécifiques. Il crée également une table "utilisateur" pour stocker les données des utilisateurs.

```
conn = psycopg2.connect(
    host= "localhost",
    dbname= "postgres",
    user= "postgres",
    password= "tays2004",
    port= 5432
)

cur = conn.cursor()
cur.execute("""
CREATE TABLE IF NOT EXISTS utilisateur (
    utilisateur VARCHAR(25) PRIMARY KEY,
    motdepasse VARCHAR(25),
    points INT DEFAULT 0
)
""")
```

## Interface Utilisateur :

L'interface utilisateur est définie dans la structure KV avec plusieurs écrans, tels que le MenuScreen, le SciencesScreen1, etc. Chacun a des composants comme des étiquettes, des champs de texte, et des actions déclenchées par des méthodes Python, comme la vérification des réponses.

```
MDFlatButton:
    text: "Géographie"
    pos_hint: {'center_x': 0.8, 'center_y': 0.3}
    on_press: app.root.current = 'geo'
MDFlatButton:
    text: "Aléatoire"
    pos_hint: {'center_x': 0.8, 'center_y': 0.1}
    on_press: app.root.current = 'alea'
```

```
MDFlatButton:
    text: "Valider"
    pos_hint: {'center_x': 0.5, 'center_y': 0.2}
    size_hint: 0.3, 0.1
    md_bg_color: 'blue'
    on_press: root.next()
```

## Fonctionnement Général :

L'application utilise une approche à plusieurs écrans pour la navigation. La classe KivizzApp crée et gère l'application, charge le fichier KV, et contient une méthode **on\_stop** pour fermer la connexion à la base de données lors de la fermeture de l'application.

```
class KivizzApp(MDApp):
    def build(self):
        self.manager = ManageScreen()
        screen = Builder.load_string(KV)
        return screen
    def on_stop(self):
        cur.close()
        conn.close()
```

Puis on finit par run l'application:

```
if __name__=="__main__":
    KivizzApp().run()
```



## V Outils utilisés

### **Langage et Framework :**

- Langage de Programmation : Python
- Framework : Kivy

### **Bibliothèques Tierces :**

- KivyMD : Utilisé pour des composants d'interface utilisateur spécifiques à Material Design dans Kivy.
- Psycopg2 : Bibliothèque PostgreSQL pour la connexion et la manipulation de la base de données.
- Time : Bibliothèque standard Python pour la gestion du temps.

### **-Utilisation du Langage KV :**

Le langage KV est intégré dans le code pour décrire de manière déclarative la structure de l'interface utilisateur. Il simplifie la création de widgets et la définition de leurs interactions sans nécessiter de codage Python supplémentaire pour chaque élément.