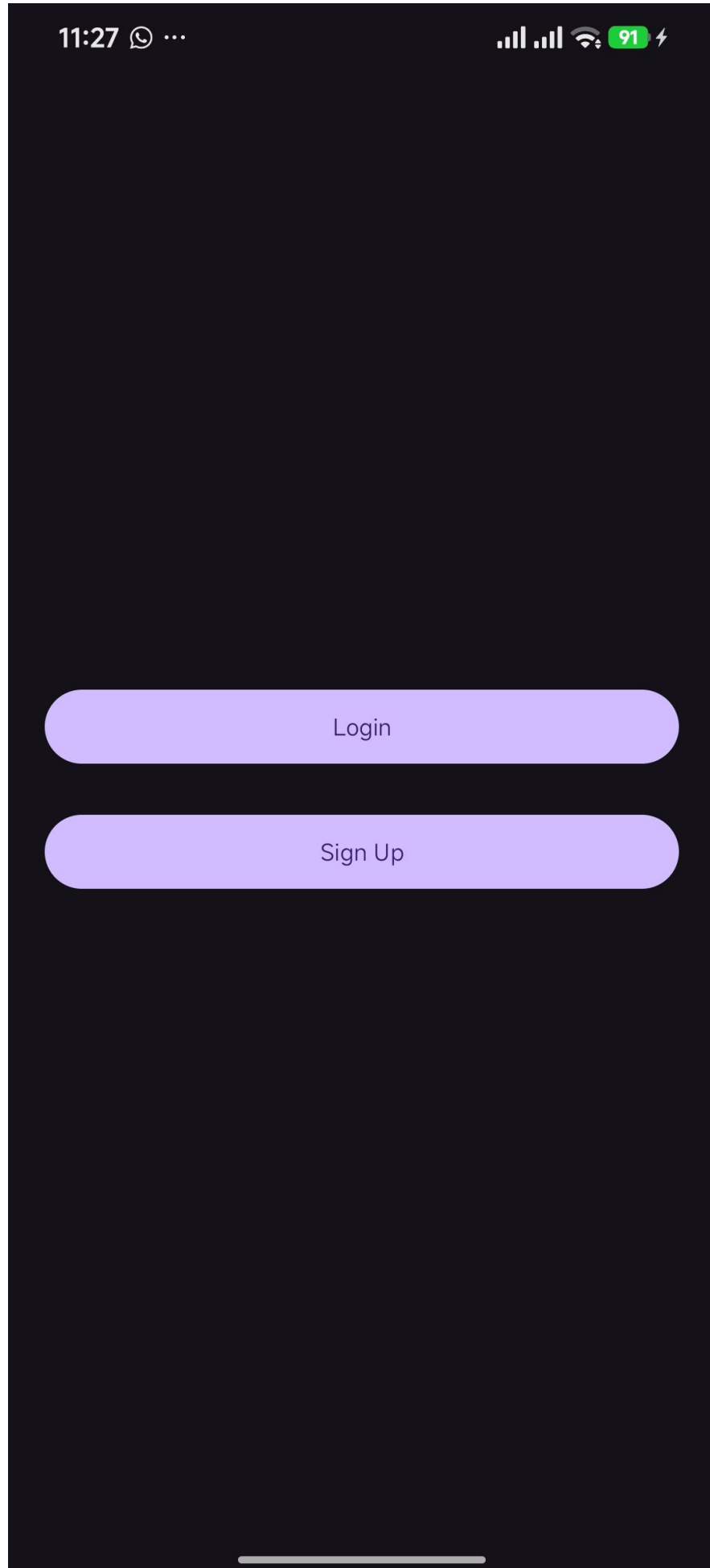


Développement Mobile

TP6

Crée par Yassir Ghazi
2-BTS DAI -2

- Interface principal de l'application



- *L'interface de registremnent*

11:28 🌙 ... 📶 91 🔋

yassir

20

yassir.com !

• • • • • • • • • •

Valider

🏠 GIF 📋 ⚙️ 💬 🎤

1 2 3 4 5 6 7 8 9 0

a z e r t y u i o p

q s d f g h j k l m

⬆️ w x c v b n ' ⬅️

?123 , 🌐 English . ✓

📶

11:27 🌙 ... 📶 91 🔋

yassir

Age !

Email

Mot de passe

Valider

🏠 yassir | Yassir | yasser 🎤

1 2 3 4 5 6 7 8 9 0

a z e r t y u i o p

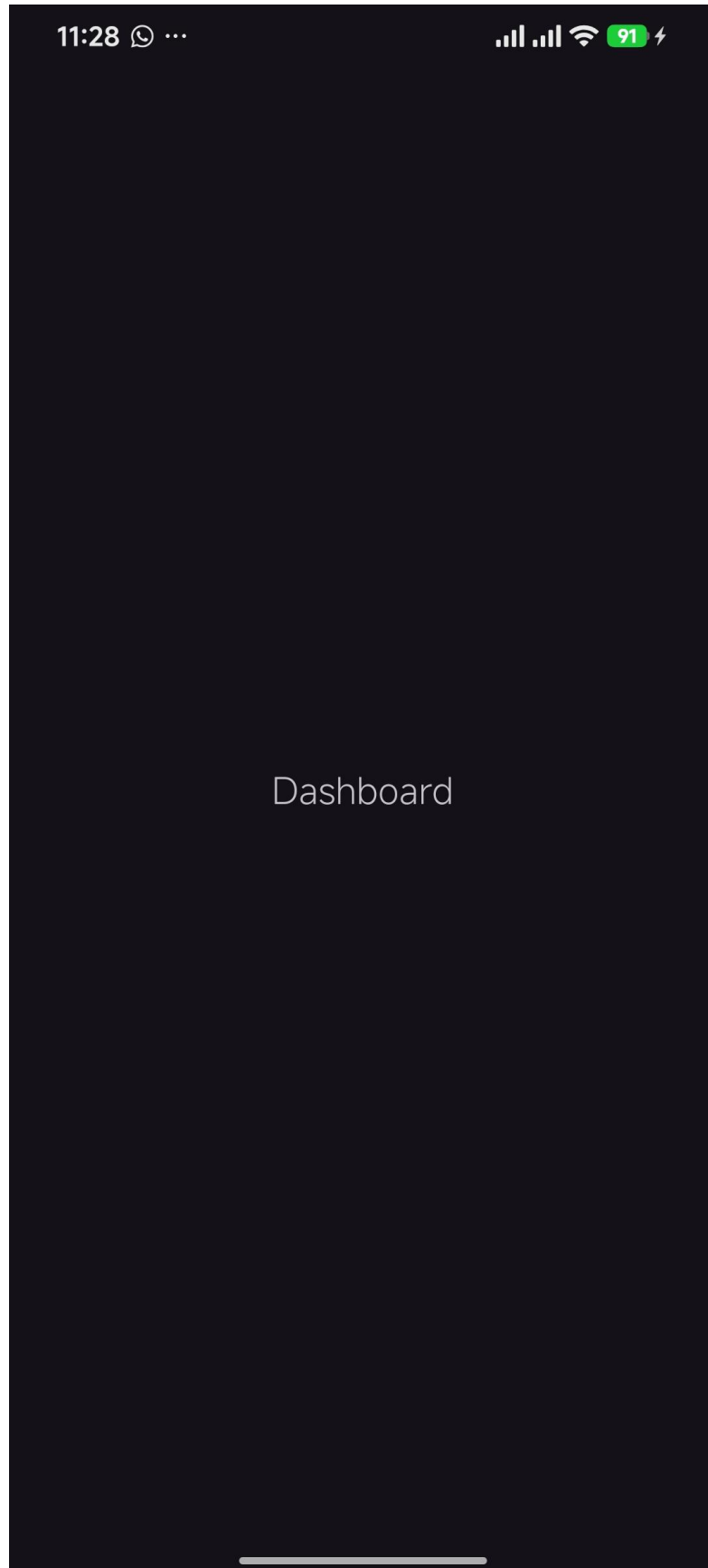
q s d f g h j k l m

⬆️ w x c v b n ' ⬅️

?123 😊 🌐 EN • FR . ⬅️

📶

- Interface après la login





Section 1 — Le Code

1. Méthode **verifierFormulaire()**

```
private boolean verifierFormulaire() {

    String nom = editNom.getText().toString().trim();
    String ageStr = editAge.getText().toString().trim();
    String email = editEmail.getText().toString().trim();
    String password = editPassword.getText().toString();

    if(nom.isEmpty()){
        editNom.setError("Nom obligatoire");
        return false;
    }

    if(ageStr.isEmpty()){
        editAge.setError("Age obligatoire");
        return false;
    }

    try {
        int age = Integer.parseInt(ageStr);

        if(age < 0 || age > 120){
            editAge.setError("Age invalide");
            return false;
        }

    } catch(NumberFormatException e){
        editAge.setError("Age numérique seulement");
        return false;
    }

    if(!email.contains("@")){
        editEmail.setError("Email invalide");
        return false;
    }

    if(password.length() < 6){
        editPassword.setError("Mot de passe doit contenir au moins 6 caractères");
        return false;
    }

    return true;
}
```

2. Méthode **validerFormulaire()** (avec try / catch)

```
private void validerFormulaire() {  
    try {  
        User user = new User(  
            editNom.getText().toString(),  
            Integer.parseInt(editAge.getText().toString()),  
            editEmail.getText().toString(),  
            editPassword.getText().toString()  
        );  
  
        boolean inserted = userDao.insertUser(user);  
  
        if(inserted){  
            Toast.makeText(this, "Inscription réussie",  
Toast.LENGTH_LONG).show();  
            finish();  
        } else {  
            Toast.makeText(this, "Email déjà utilisé",  
Toast.LENGTH_LONG).show();  
        }  
  
    } catch (Exception e){  
        Toast.makeText(this, "Erreur inattendue",  
Toast.LENGTH_LONG).show();  
    }  
}
```

3. Implémentation du **TextWatcher**

```
TextWatcher watcher = new TextWatcher() {  
  
    @Override  
    public void beforeTextChanged(CharSequence s, int start, int count,  
int after) {}  
  
    @Override  
    public void onTextChanged(CharSequence s, int start, int before, int  
count) {  
        btnValider.setEnabled(verifierFormulaire());  
    }  
  
    @Override  
    public void afterTextChanged(Editable s) {}  
};  
  
editNom.addTextChangedListener(watcher);  
editAge.addTextChangedListener(watcher);  
editEmail.addTextChangedListener(watcher);  
editPassword.addTextChangedListener(watcher);
```

4. Classe DatabaseHelper

public class DatabaseHelper extends
SQLiteOpenHelper {

```
    public DatabaseHelper(Context context) {  
        super(context, "tp9.db", null, 1);  
    }
```

```
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        db.execSQL("CREATE TABLE utilisateur(" +  
            "id INTEGER PRIMARY KEY  
AUTOINCREMENT," +  
            "nom TEXT," +  
            "age INTEGER," +  
            "email TEXT UNIQUE," +  
            "password TEXT)");  
    }
```

```
    @Override  
    public void onUpgrade(SQLiteDatabase db, int  
oldVersion, int newVersion) {  
        db.execSQL("DROP TABLE IF EXISTS  
utilisateur");  
        onCreate(db);  
    }  
}
```

5. Classe UserDao

public class UserDao {

DatabaseHelper dbHelper;

```
    public UserDao(Context context) {  
        dbHelper = new DatabaseHelper(context);  
    }
```

```

public boolean insertUser(User user){
    SQLiteDatabase db =
dbHelper.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put("nom", user.getNom());
    values.put("age", user.getAge());
    values.put("email", user.getEmail());
    values.put("password", user.getPassword());

    long result = db.insert("utilisateur", null, values);

    db.close();
    return result != -1;
}

public boolean login(String email, String password){

    SQLiteDatabase db =
dbHelper.getReadableDatabase();

    Cursor cursor = db.rawQuery(
        "SELECT * FROM utilisateur WHERE email=?
AND password=?",
        new String[]{email, password});

    boolean exists = cursor.getCount() > 0;

    cursor.close();
    db.close();

    return exists;
}
}

```


Section 2 — Les réponses aux questions

1. Pourquoi utilise-t-on try / catch ?

On utilise le bloc try / catch pour intercepter les exceptions et éviter le crash de l'application.

Par exemple, lors de la conversion d'un texte en entier (Integer.parseInt), une erreur peut se produire si l'utilisateur entre des caractères non numériques.

Le try / catch permet de gérer cette erreur proprement.

2. Quelle est la différence entre setError() et Toast ?

setError() affiche un message directement sous le champ concerné. Il est utilisé pour signaler une erreur spécifique liée à un champ précis.

Toast affiche un message temporaire à l'écran. Il est utilisé pour informer l'utilisateur d'un résultat global (succès ou erreur générale).

3. Pourquoi désactiver le bouton avant validation ?

Le bouton est désactivé pour empêcher l'utilisateur d'envoyer un formulaire invalide.

Cela améliore l'expérience utilisateur et réduit les erreurs.

4. Que se passe-t-il si on supprime la gestion des exceptions ?

Si on supprime la gestion des exceptions, l'application peut se fermer brutalement (crash) lorsqu'une erreur se produit, comme une mauvaise conversion de type.

Cela rend l'application instable et non professionnelle.

