

Documentation du projet Geocalculator

Sommaire

I/ Geocalculator

a) Introduction.....	2
b) Création des fichiers PHP.....	3-7
c) installation de Postgres.....	8-15
d) fichier geocalculator.bat et affichage des tables.....	16
e) Android (MainActivity, PageCarre, AfficheCarre, etc.).....	17-38
f) Stat.....	39-40
g) Diagramme de classe.....	41
h) Cube.....	42-46

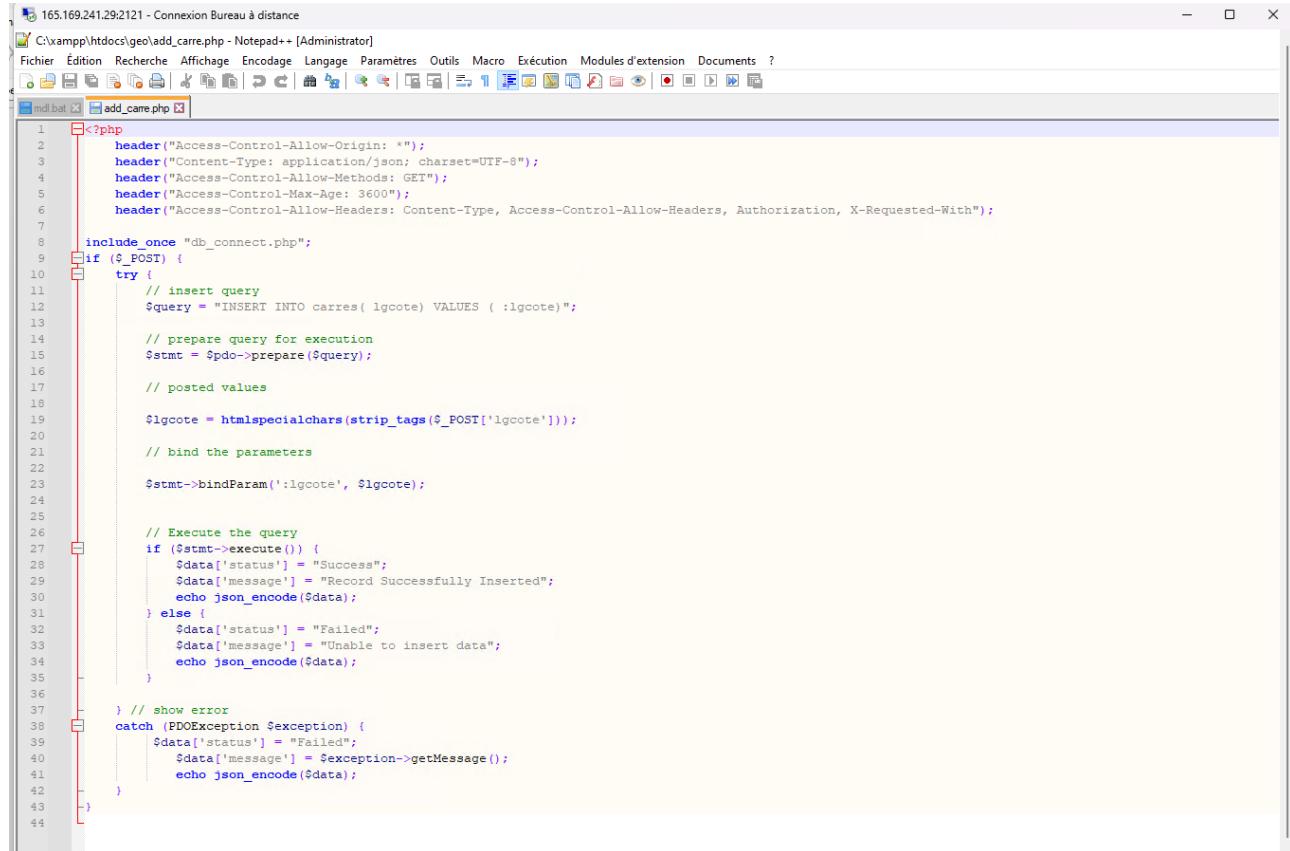
Geocalculator

Le but du projet geocalculator était de créer cinq formes tel que, un carré, un rectangle, un triangle, un cercle et un cube via un téléphone virtuel sur l’application Android. Ce projet consiste à calculer la surface, le périmètre et le volume selon la forme souhaitée. Pour calculer ces trois paramètres il faut entrer la longueur, la largeur, le rayon, et la longueur du coté selon ces différentes figures.

Afin de pouvoir calculer ces paramètres pour chaque forme, par exemple pour le carré, il faut suivre les étapes suivantes :

b) Création des fichiers PHP

- créer un fichier PHP permettant d'ajouter un carré en le nommant add_carre.php :

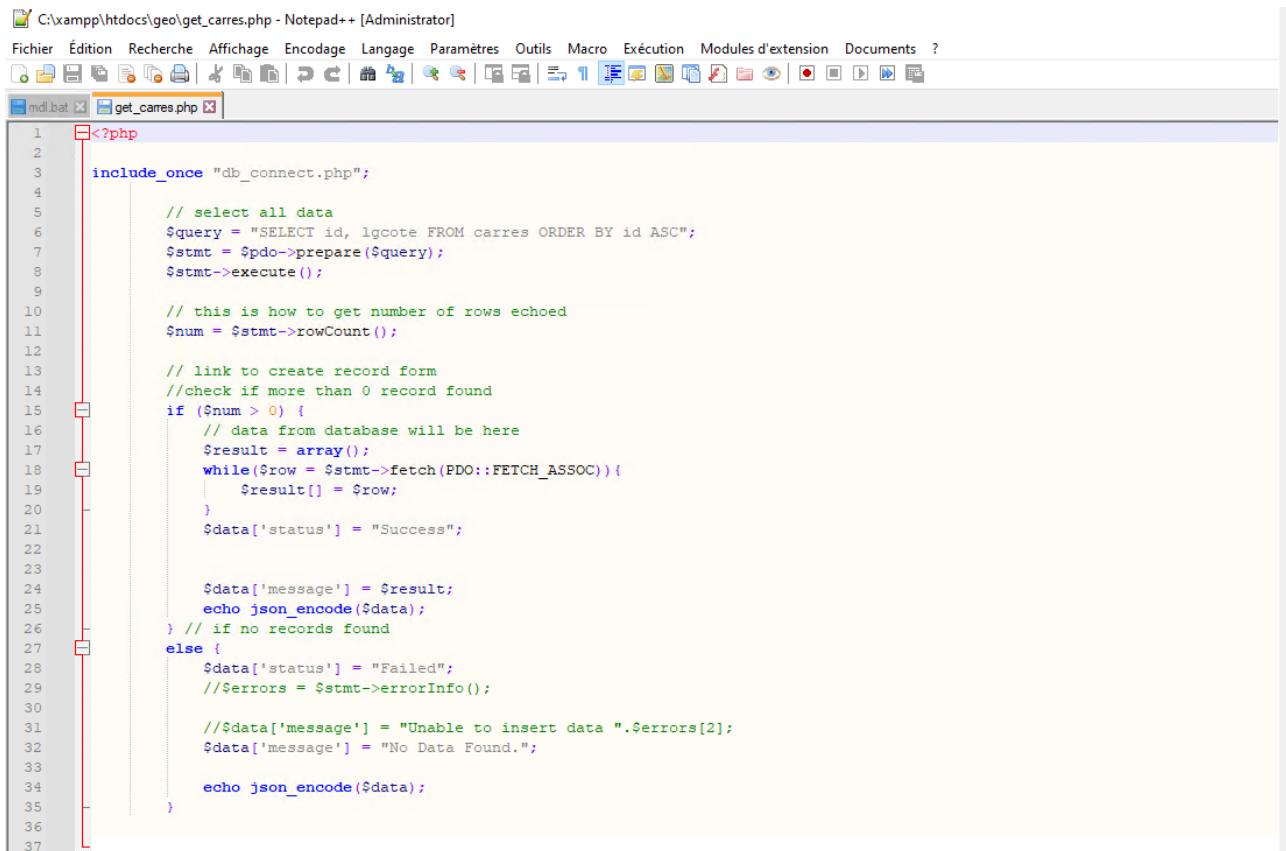


```

1 <?php
2     header("Access-Control-Allow-Origin: *");
3     header("Content-Type: application/json; charset=UTF-8");
4     header("Access-Control-Allow-Methods: GET");
5     header("Access-Control-Max-Age: 3600");
6     header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");
7
8     include_once "db_connect.php";
9     if ($_POST) {
10         try {
11             // insert query
12             $query = "INSERT INTO carres( lgcote ) VALUES ( :lgcote )";
13
14             // prepare query for execution
15             $stmt = $pdo->prepare($query);
16
17             // posted values
18
19             $lgcote = htmlspecialchars(strip_tags($_POST['lgcote']));
20
21             // bind the parameters
22
23             $stmt->bindParam(':lgcote', $lgcote);
24
25
26             // Execute the query
27             if ($stmt->execute()) {
28                 $data['status'] = "Success";
29                 $data['message'] = "Record Successfully Inserted";
30                 echo json_encode($data);
31             } else {
32                 $data['status'] = "Failed";
33                 $data['message'] = "Unable to insert data";
34                 echo json_encode($data);
35             }
36
37         } // show error
38         catch (PDOException $exception) {
39             $data['status'] = "Failed";
40             $data['message'] = $exception->getMessage();
41             echo json_encode($data);
42         }
43     }
44 }

```

- créer un fichier PHP permettant de récupérer les données du carré en le nommant get_carres.php :



The screenshot shows the Notepad++ interface with the file 'get_carres.php' open. The code is a PHP script that connects to a database, selects all data from the 'carres' table, and returns it as JSON. It includes error handling for no records found.

```
<?php
include_once "db_connect.php";
// select all data
$query = "SELECT id, lgcote FROM carres ORDER BY id ASC";
$stmt = $pdo->prepare($query);
$stmt->execute();

// this is how to get number of rows echoed
$num = $stmt->rowCount();

// link to create record form
//check if more than 0 record found
if ($num > 0) {
    // data from database will be here
    $result = array();
    while($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
        $result[] = $row;
    }
    $data['status'] = "Success";

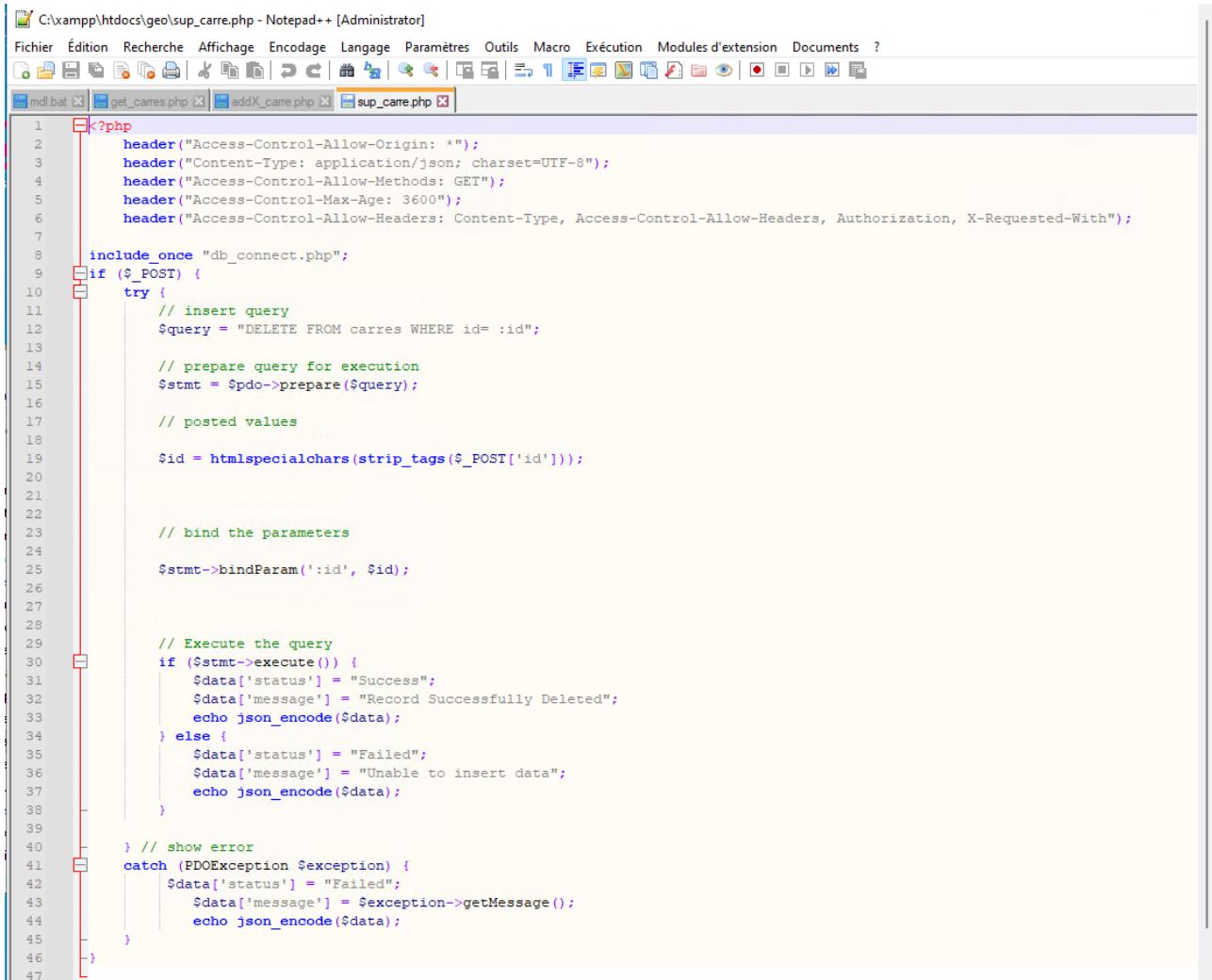
    $data['message'] = $result;
    echo json_encode($data);
} // if no records found
else {
    $data['status'] = "Failed";
    //errors = $stmt->errorInfo();

    //data['message'] = "Unable to insert data ".errors[2];
    $data['message'] = "No Data Found.';

    echo json_encode($data);
}
```

4

- créer un fichier PHP permettant de supprimer un carré en le nommant sup_carre.php :

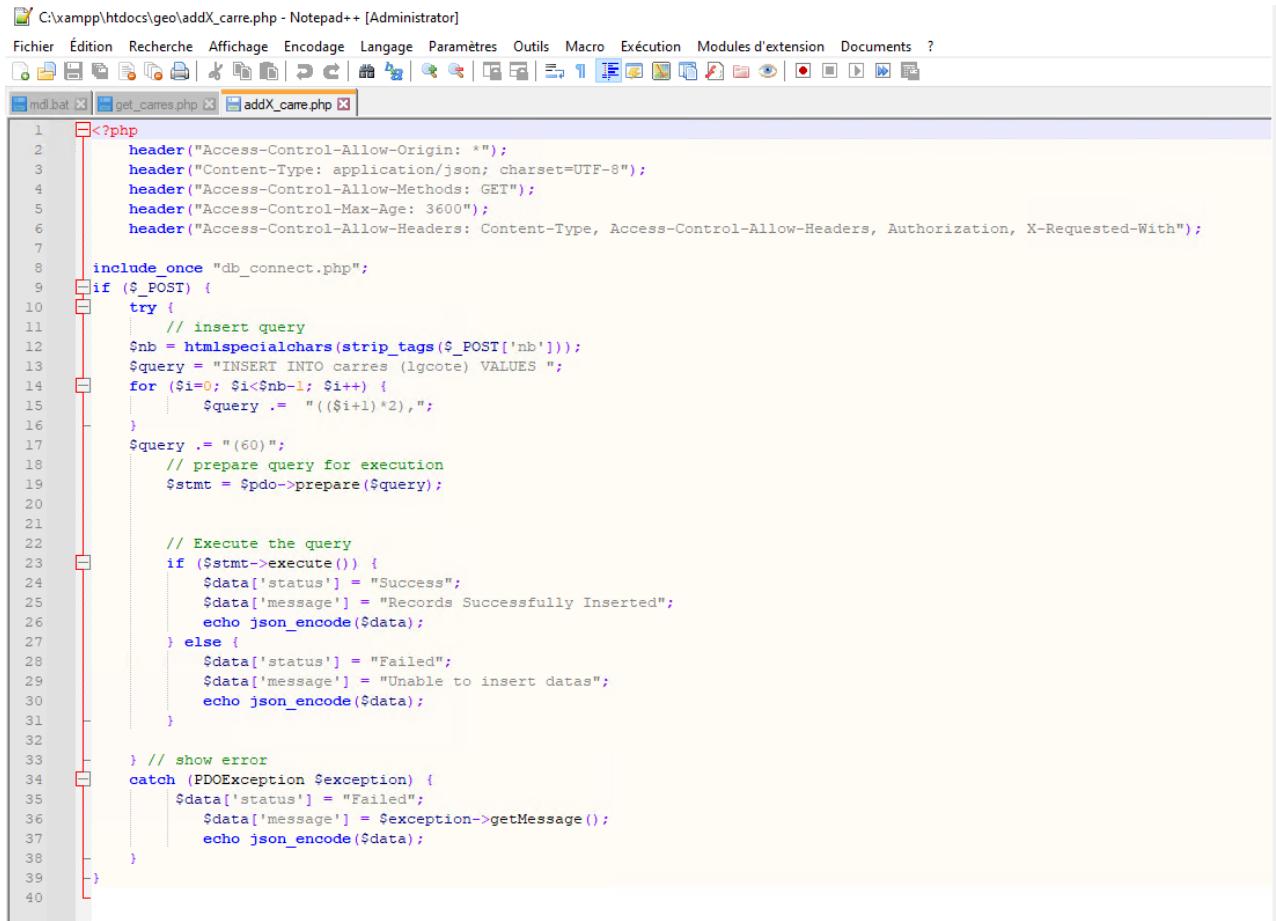


```

1 <?php
2     header("Access-Control-Allow-Origin: *");
3     header("Content-Type: application/json; charset=UTF-8");
4     header("Access-Control-Allow-Methods: GET");
5     header("Access-Control-Max-Age: 3600");
6     header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");
7
8     include_once "db_connect.php";
9     if ($_POST) {
10         try {
11             // insert query
12             $query = "DELETE FROM carres WHERE id= :id";
13
14             // prepare query for execution
15             $stmt = $pdo->prepare($query);
16
17             // posted values ...
18
19             $id = htmlspecialchars(strip_tags($_POST['id']));
20
21
22             // bind the parameters
23
24             $stmt->bindParam(':id', $id);
25
26
27
28             // Execute the query
29             if ($stmt->execute()) {
30                 $data['status'] = "Success";
31                 $data['message'] = "Record Successfully Deleted";
32                 echo json_encode($data);
33             } else {
34                 $data['status'] = "Failed";
35                 $data['message'] = "Unable to insert data";
36                 echo json_encode($data);
37             }
38
39
40             // show error
41             catch (PDOException $exception) {
42                 $data['status'] = "Failed";
43                 $data['message'] = $exception->getMessage();
44                 echo json_encode($data);
45             }
46         }
47     }

```

- créer un fichier PHP permettant d'ajouter plusieurs carres en même temps en le nommant addX_carre.php :



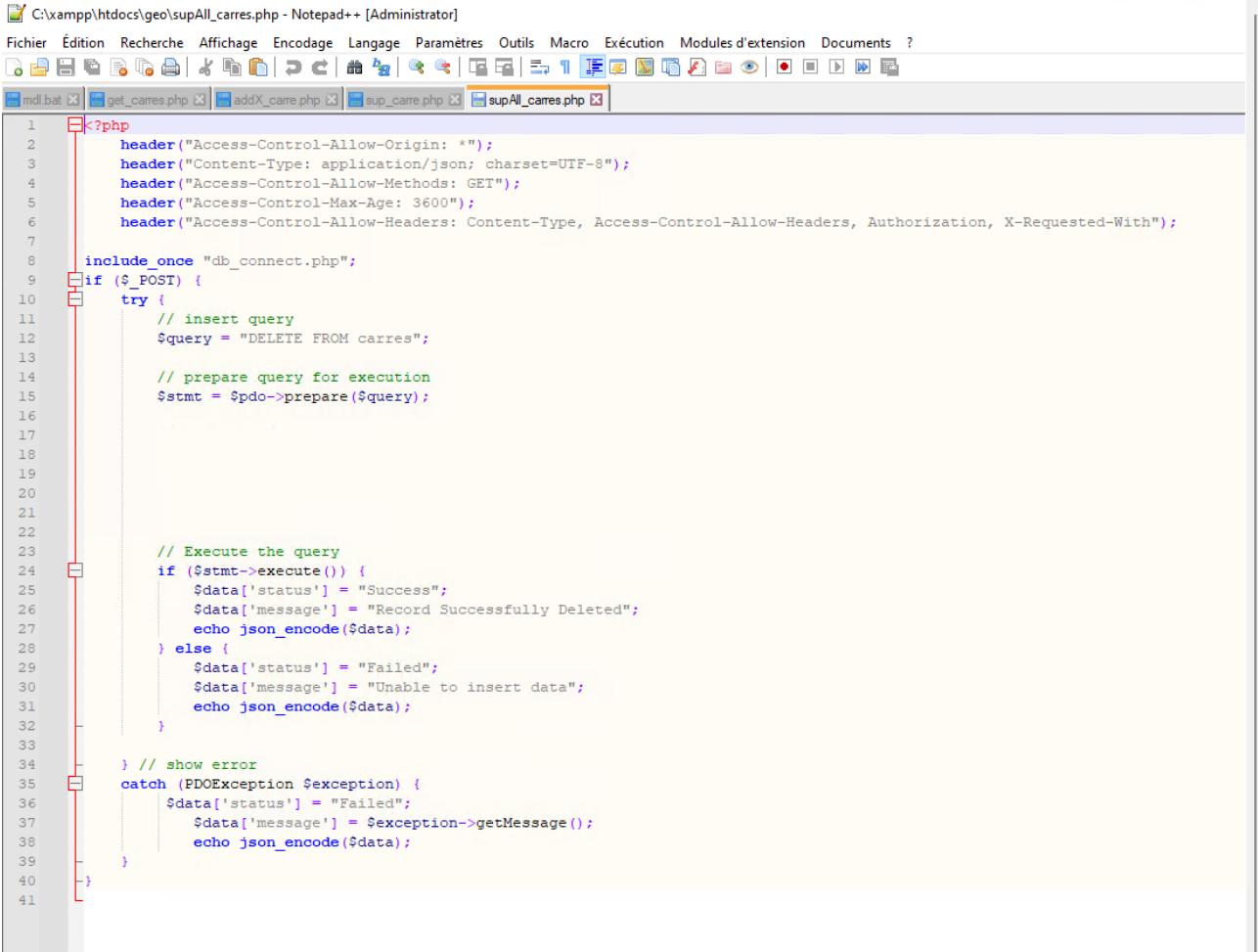
```

1 <?php
2     header("Access-Control-Allow-Origin: *");
3     header("Content-Type: application/json; charset=UTF-8");
4     header("Access-Control-Allow-Methods: GET");
5     header("Access-Control-Max-Age: 3600");
6     header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");
7
8     include_once "db_connect.php";
9     if ($_POST) {
10         try {
11             // insert query
12             $nb = htmlspecialchars(strip_tags($_POST['nb']));
13             $query = "INSERT INTO carres (lgcote) VALUES ";
14             for ($i=0; $i<$nb-1; $i++) {
15                 $query .= "($i+1)*2,";
16             }
17             $query .= "(60)";
18             // prepare query for execution
19             $stmt = $pdo->prepare($query);
20
21             // Execute the query
22             if ($stmt->execute()) {
23                 $data['status'] = "Success";
24                 $data['message'] = "Records Successfully Inserted";
25                 echo json_encode($data);
26             } else {
27                 $data['status'] = "Failed";
28                 $data['message'] = "Unable to insert datas";
29                 echo json_encode($data);
30             }
31
32         } // show error
33         catch (PDOException $exception) {
34             $data['status'] = "Failed";
35             $data['message'] = $exception->getMessage();
36         }
37     }
38
39 }
40

```

6

- créer un fichier PHP permettant d'ajouter plusieurs carres en même temps en le nommant supAll_carres.php :



The screenshot shows a Notepad++ window with the file 'supAll_carres.php' open. The code is a PHP script that handles a DELETE request to remove all records from a 'carres' database table. It includes header definitions for CORS, JSON content type, and various access control headers. It uses PDO to execute a delete query and return a JSON response indicating success or failure.

```
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: GET");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

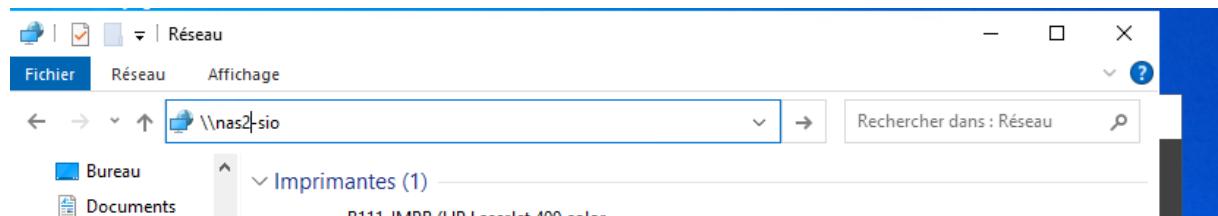
include_once "db_connect.php";
if ($_POST) {
    try {
        // insert query
        $query = "DELETE FROM carres";

        // prepare query for execution
        $stmt = $pdo->prepare($query);

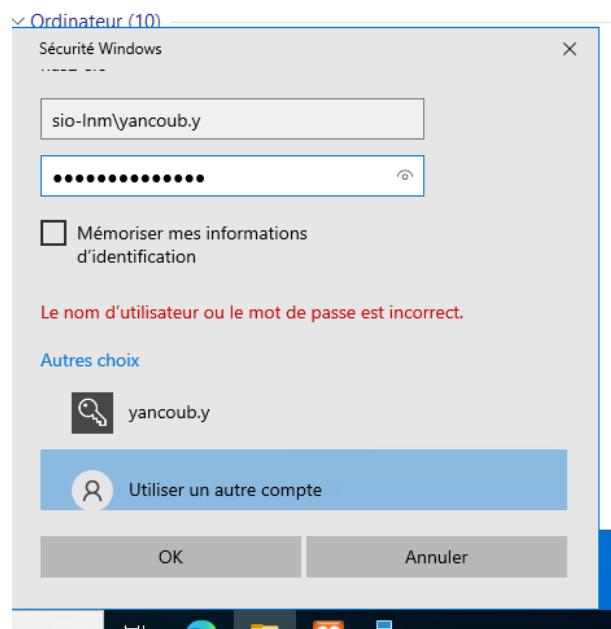
        // Execute the query
        if ($stmt->execute()) {
            $data['status'] = "Success";
            $data['message'] = "Record Successfully Deleted";
            echo json_encode($data);
        } else {
            $data['status'] = "Failed";
            $data['message'] = "Unable to insert data";
            echo json_encode($data);
        }
    } // show error
    catch (PDOException $exception) {
        $data['status'] = "Failed";
        $data['message'] = $exception->getMessage();
        echo json_encode($data);
    }
}
```

c) Installation de Postgres

- Pour installer Postgres on peut récupérer directement l'application sur le NAS :

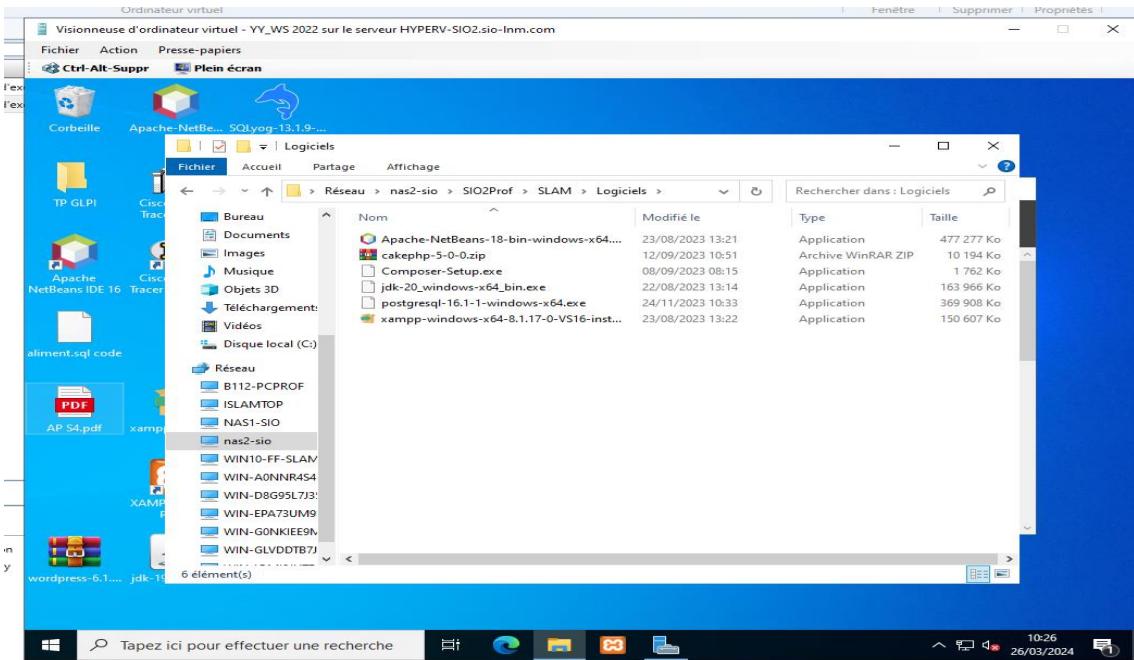


- Se connecter

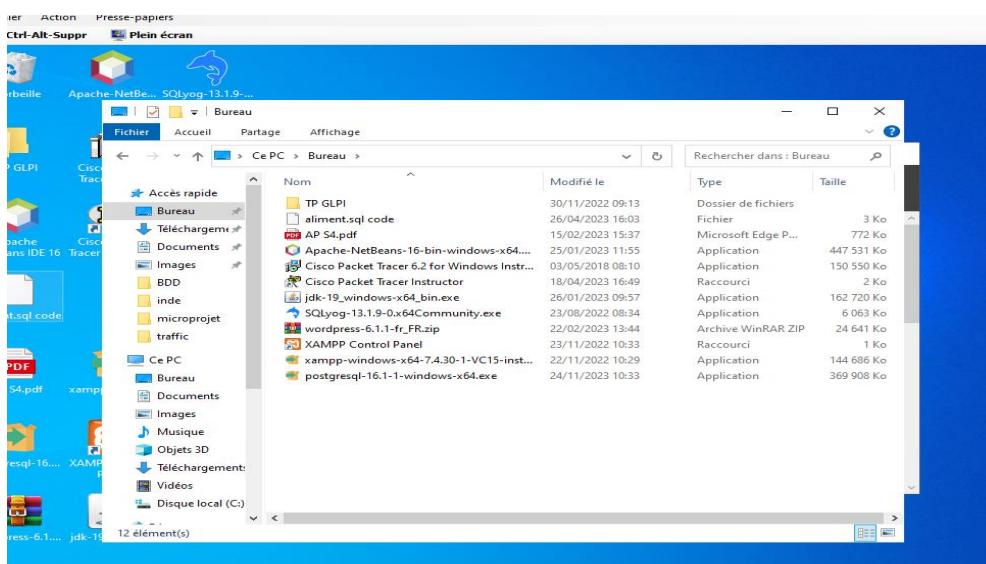


8

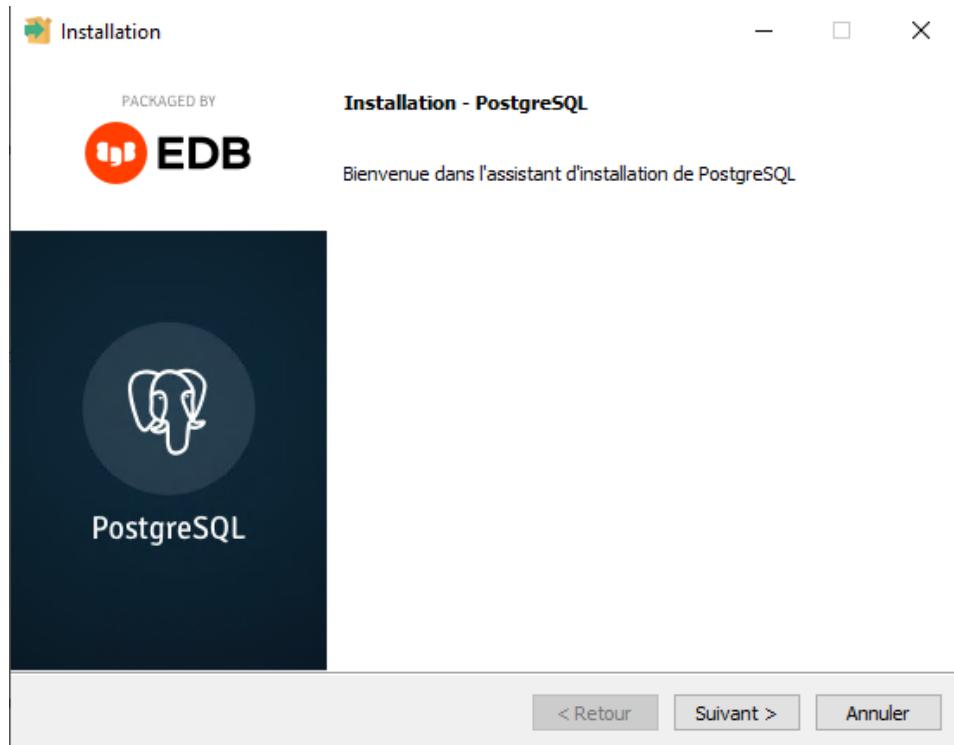
- On récupère le logiciel dans SIO2prof>SLAM>Logiciels



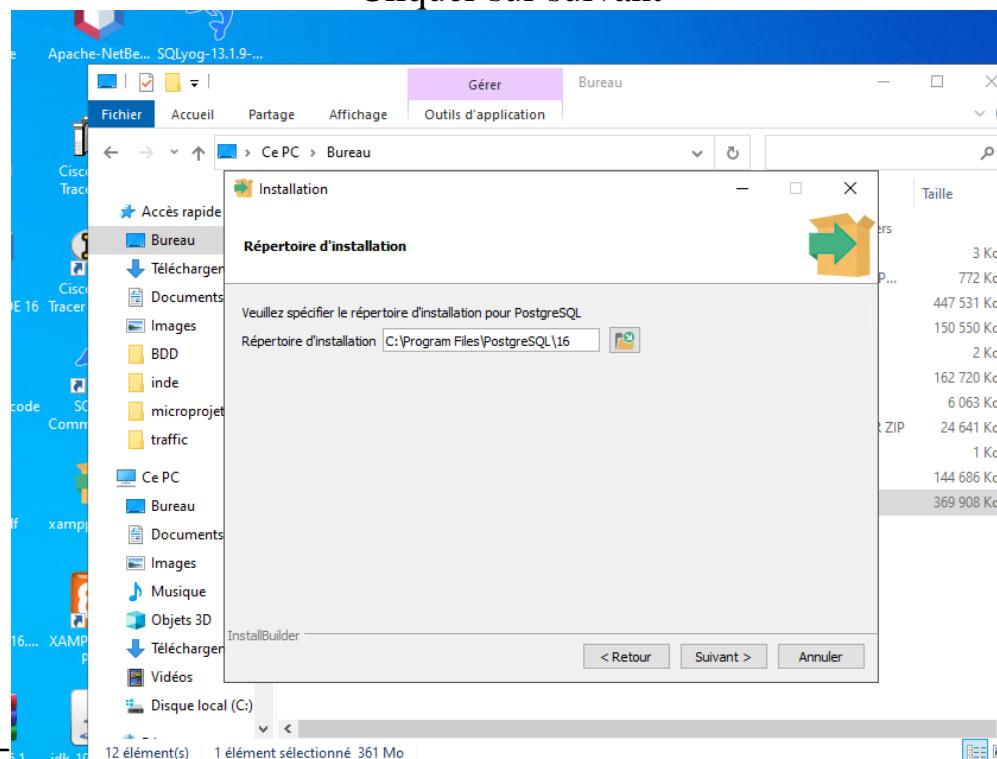
- On le copie-colle dans le bureau



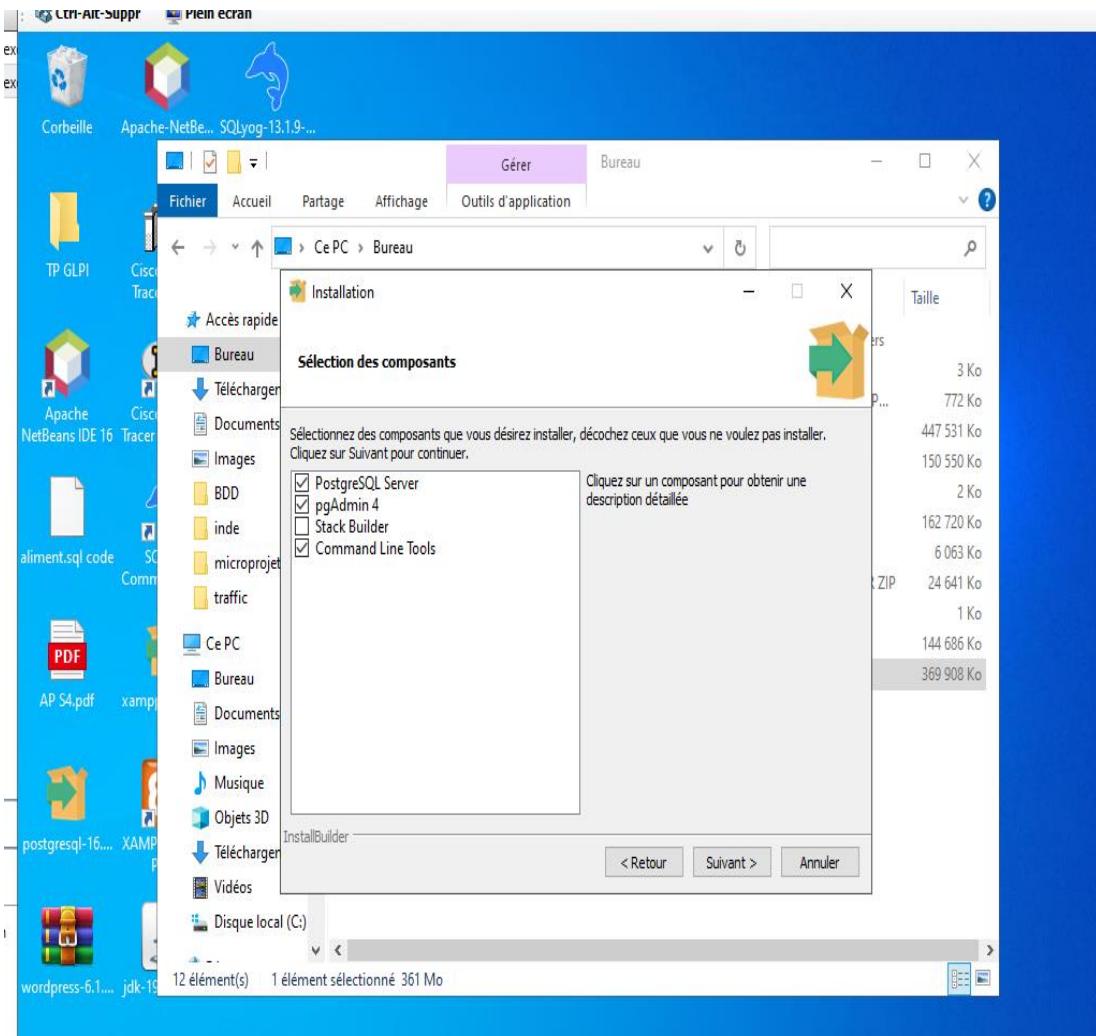
- On commence l'installation, puis suivant



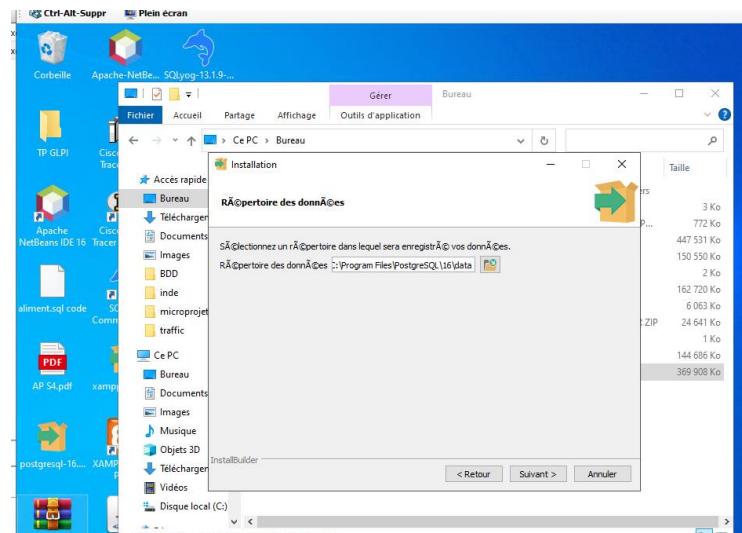
- Cliquer sur suivant



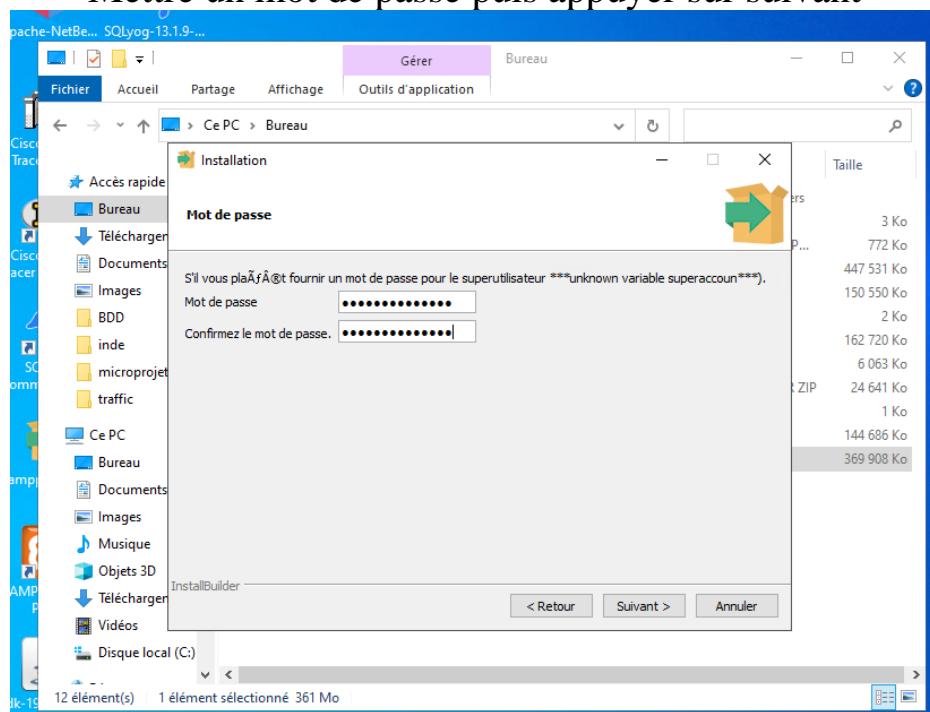
- Cliquer sur suivant



- Faire suivant

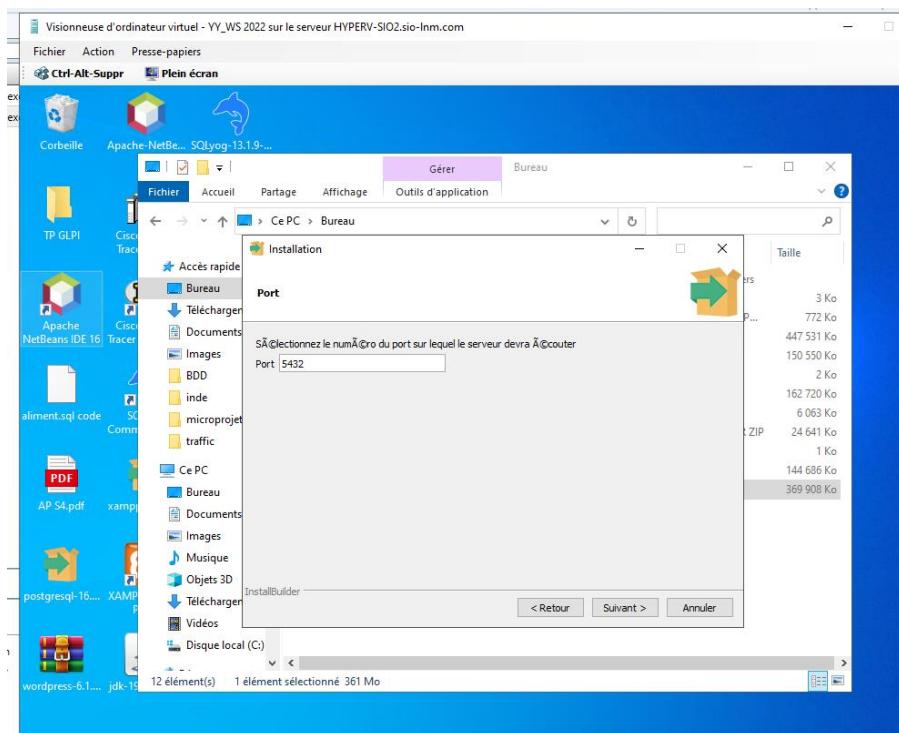


Mettre un mot de passe puis appuyer sur suivant

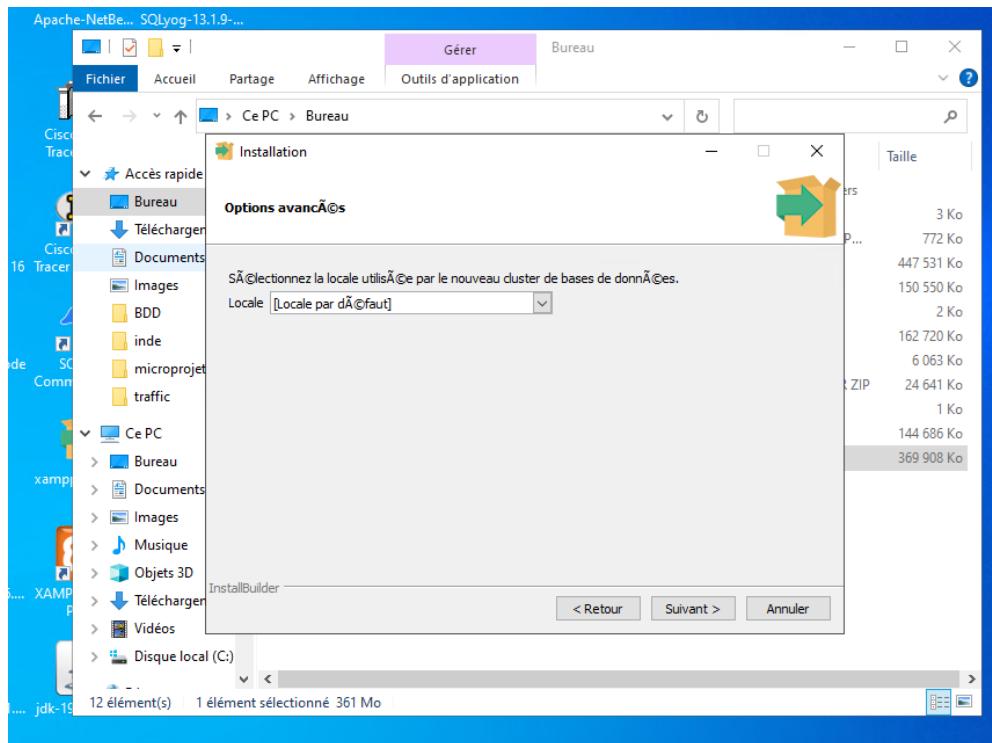


12

- Laisser le port par défaut



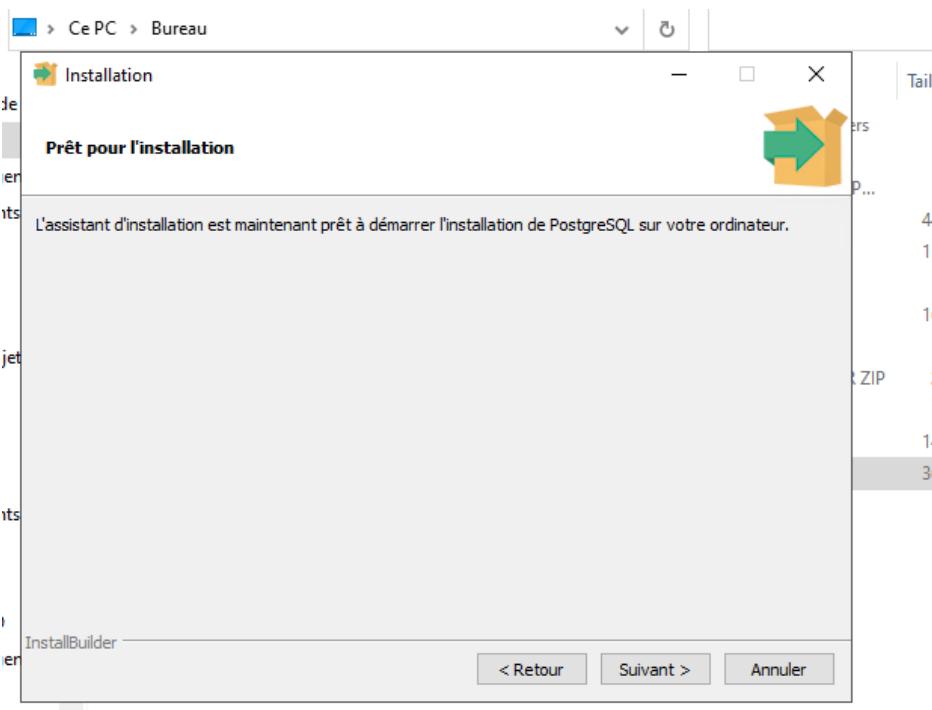
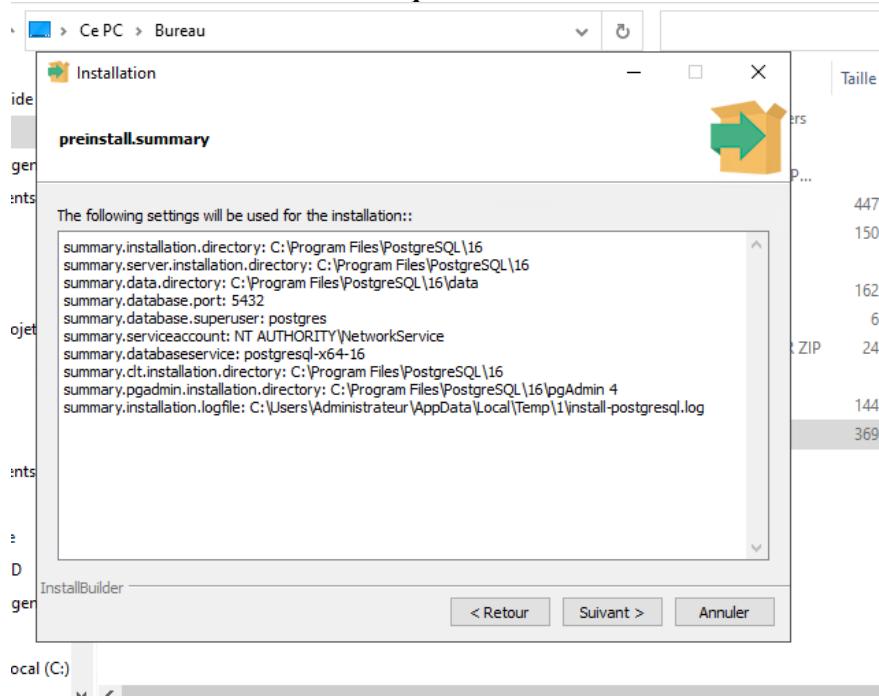
- Faire suivant



- Attendre et faire suivant

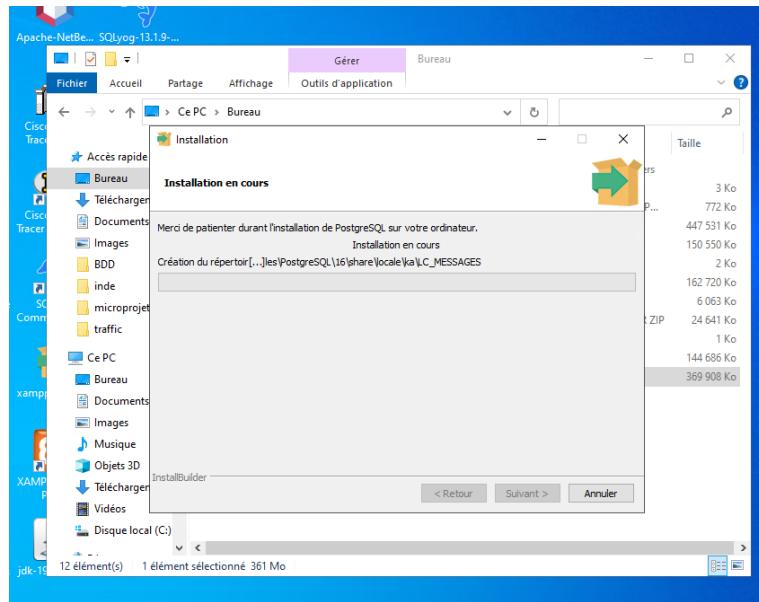
13

- Cliquer sur suivant

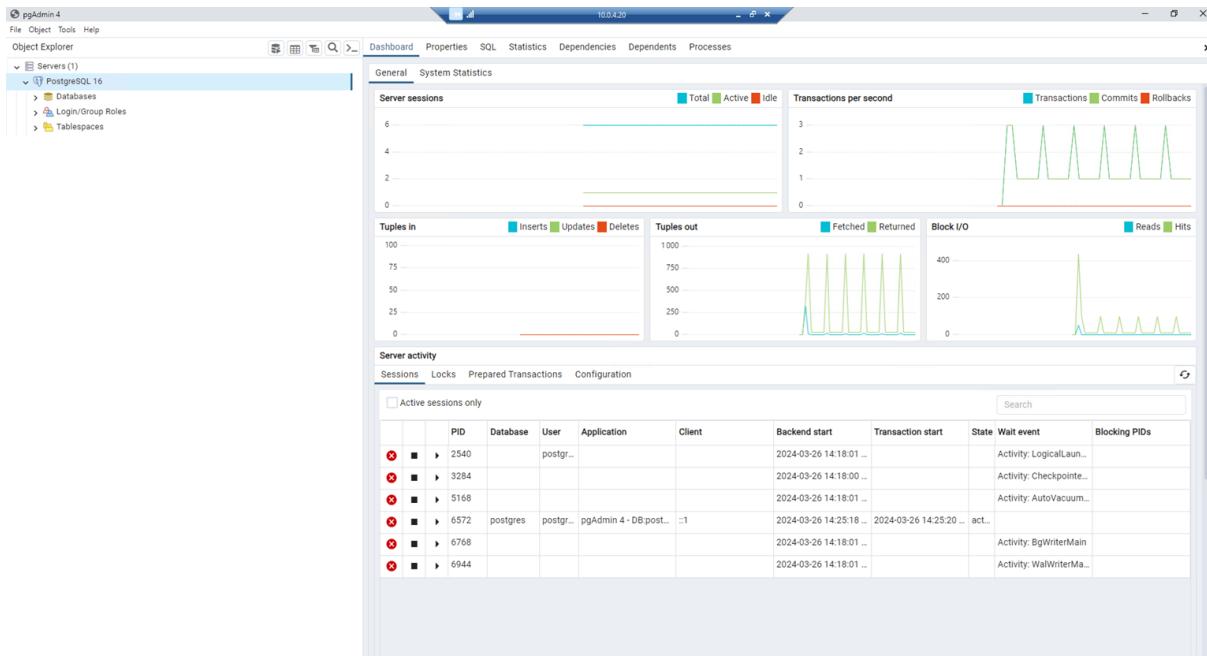


- Faire suivant

- Attendre l'installation



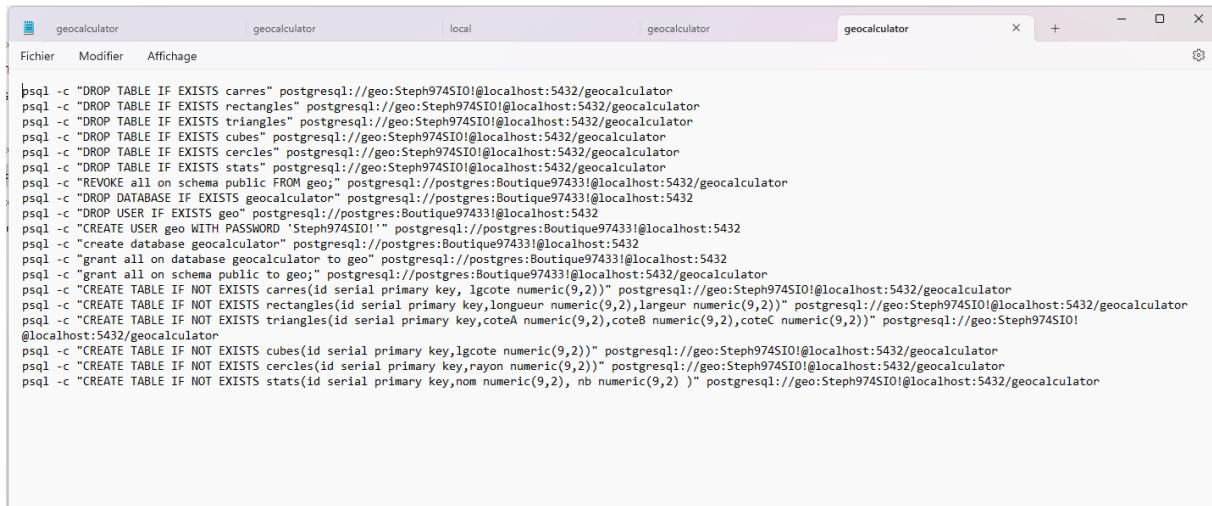
- Il faut mettre le mot de passe choisi lors de l'installation pour obtenir ce résultat



c) geocalculator.bat

Ce fichier permet de faire la création des données dans la base de données Postgres.

- création des différentes tables ;
- suppression des tables ;
- permet de donner et de supprimer les droits.

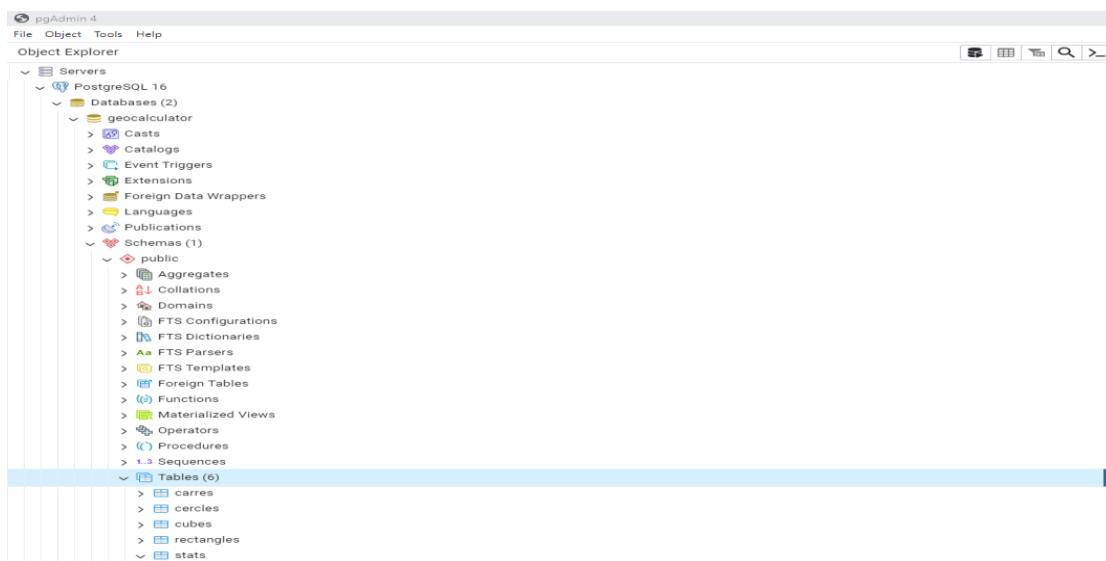


```

pgsql -c "DROP TABLE IF EXISTS carres" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "DROP TABLE IF EXISTS rectangles" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "DROP TABLE IF EXISTS triangles" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "DROP TABLE IF EXISTS cubes" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "DROP TABLE IF EXISTS stats" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "REVOKE all on schema public FROM geo;" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "DROP DATABASE IF EXISTS geocalculator" postgresql://postgres:Boutique97433!@localhost:5432
pgsql -c "CREATE USER geo WITH PASSWORD 'Steph974S10!'" postgresql://postgres:Boutique97433!@localhost:5432
pgsql -c "grant all on database geocalculator to geo" postgresql://postgres:Boutique97433!@localhost:5432
pgsql -c "CREATE TABLE IF NOT EXISTS carres(id serial primary key, lgcote numeric(9,2))" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "CREATE TABLE IF NOT EXISTS rectangles(id serial primary key,longueur numeric(9,2),largeur numeric(9,2))" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "CREATE TABLE IF NOT EXISTS triangles(id serial primary key,coteA numeric(9,2),coteB numeric(9,2),coteC numeric(9,2))" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "CREATE TABLE IF NOT EXISTS cubes(id serial primary key,lgcote numeric(9,2))" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "CREATE TABLE IF NOT EXISTS cercles(id serial primary key,rayon numeric(9,2))" postgresql://geo:Steph974S10!@localhost:5432/geocalculator
pgsql -c "CREATE TABLE IF NOT EXISTS stats(id serial primary key,nom numeric(9,2), nb numeric(9,2) )" postgresql://geo:Steph974S10!@localhost:5432/geocalculator

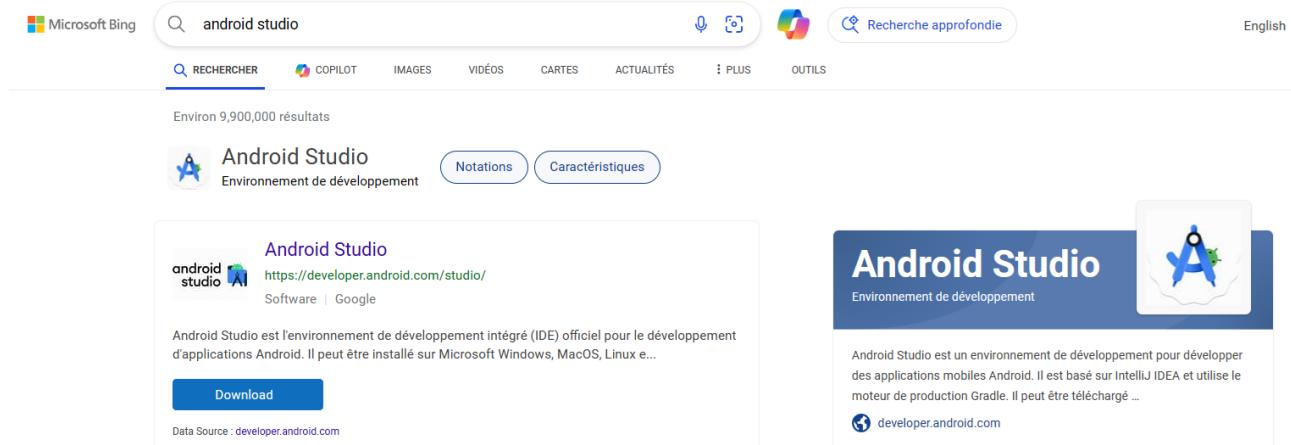
```

- Ici nous pouvons voir toutes les tables qui ont été créées dans la base de donnée geocalculator grâce au fichier geocalculator.bat :



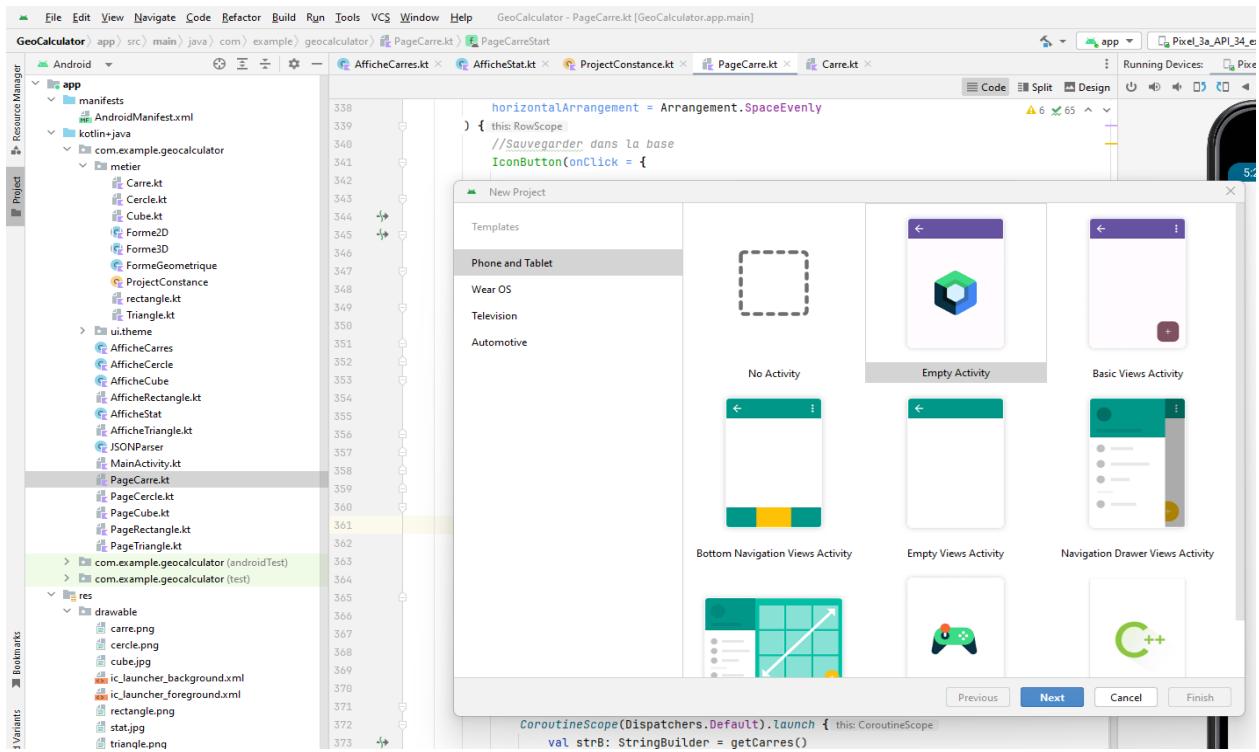
e) Android

- On peut installer Android sur internet :

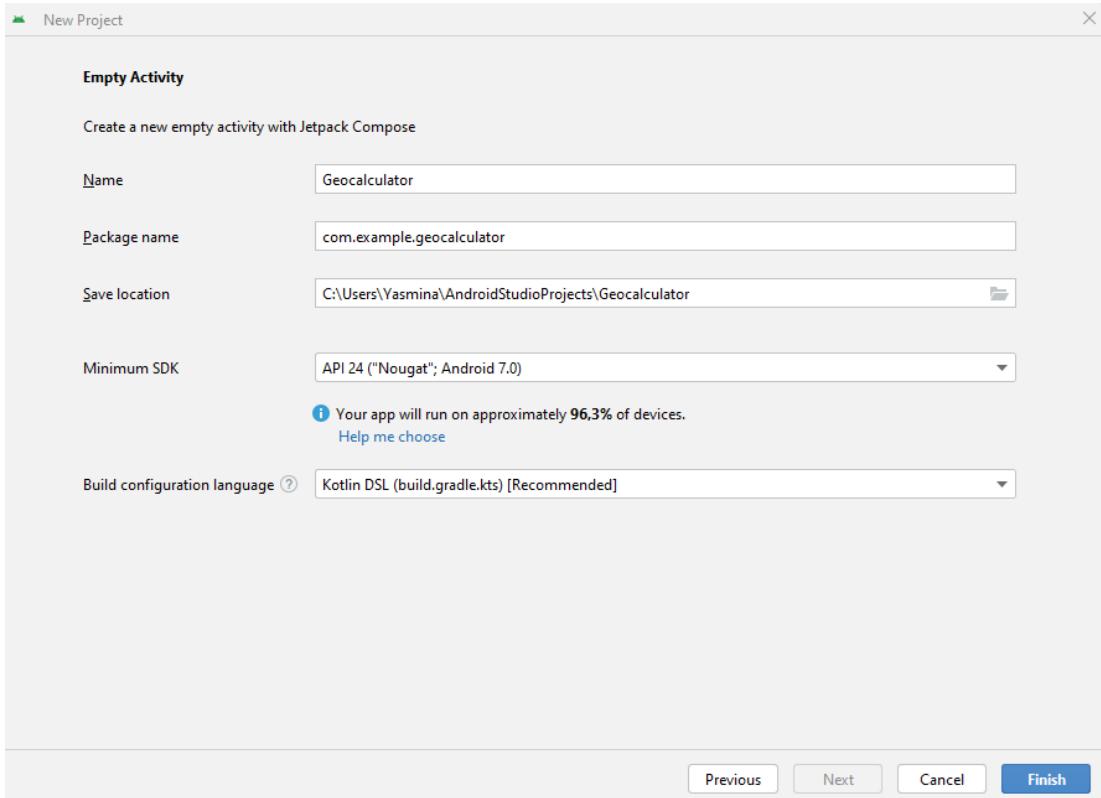


- Pour créer un nouveau projet il faut :

cliquer sur fil>New>New Project> Empty Activity (C'est ce qu'on a fait pour Geocalculator) puis suivant



- Mettre le nom du projet > Finish



- Dans le MainActivity

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            GeoCalculatorTheme {
                // A surface container using the 'background' color from the theme
                var Carre = Carref(nom: "Carre", perimetre: 10.0, surface: 0.0, cote: 0.0)
                val valPerimetree = Carre.calculPerimetree()
                Carre.perimetree = valPerimetree
                val valSurface = Carre.calculSurface()
                Carre.surface = valSurface

                Log.d(tag: "debug variable", msg: "valeur perimetree =" + Carre.perimetree)
                Log.d(tag: "debug variable", msg: "valeur surface =" + Carre.surface)

                acceuil()
            }
        }
    }
}
```

- Pour afficher le titre principal sur la page d'accueil :

```
@Preview(showBackground = true)
@Composable
fun acceuil(
    modifier: Modifier = Modifier
        .wrapContentSize(Alignment.TopCenter)
        .padding(20.dp)
) {
    Column(
        modifier = modifier
    ) { this: ColumnScope
        val context = LocalContext.current

        //Titre principal
        Text(
            text = stringResource("Bienvenue sur l'application de calculs sur les formes g..."),
            color = Color.Red,
            fontSize = 30.sp,
            textAlign = TextAlign.Center
        )

        //Espace
        Spacer(modifier = Modifier.height(30.dp))
    }
}
```

- Pour afficher le sous titre sur la page d'accueil :

```
//Espace
Spacer(modifier = Modifier.height(30.dp))

//sous titre
Text() {
    text = stringResource("Cliquez sur une forme geometrique pour acceder a la pag..."),
    color = Color.Gray,
    fontSize = 30.sp,
    textAlign = TextAlign.Center
}
Spacer(modifier = Modifier.height(30.dp))
```

- Pour afficher deux formes sur une même ligne il faut les mettre dans un seul *Row*. Ci-dessous nous avons comme image le carré et le rectangle.



- On a ensuite le cercle et le triangle toujours dans le même *Row* ;

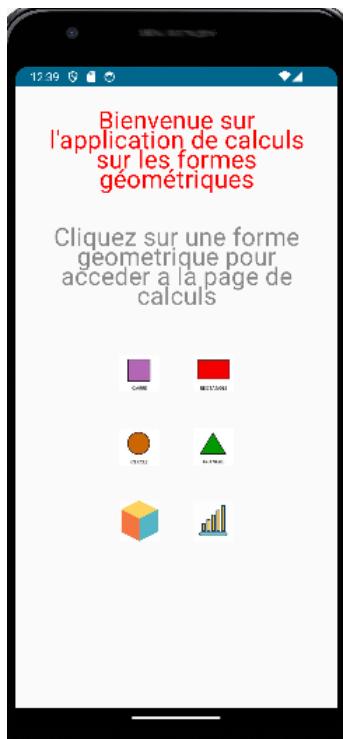


- Et enfin les deux dernières images qui sont le cube et l'image stat qui permettra d'afficher les statistiques des différentes formes.

```
Row(
    Modifier.fillMaxWidth(),
    Arrangement.Center,
    Alignment.CenterVertically
) { this: RowScope

    Image(
        painter = painterResource(R.drawable.cube),
        contentDescription = "image cube",
        modifier
            .size(50.dp)
            .clickable {
                context.startActivity(Intent(context, PageCube::class.java))
            }
    )
    Image(
        painter = painterResource(R.drawable.stat),
        contentDescription = "image stat",
        modifier
            .size(50.dp)
            .clickable {
                CoroutineScope(Dispatchers.Default).launch { thisCoroutineScope
                    val strB: StringBuilder = getStats()
                    withContext(Dispatchers.Main) { thisCoroutineScope
                        Log.i("tag: " + "apres get stat", strB.toString())
                        val intent = Intent(context, AfficheStat::class.java)
                        // now by putExtra method put the value in key, value pair key is
                        // message_key by this key we will receive the value, and put the string
                        intent.putExtra("name: " + "stat", strB.toString())
                        // start the Intent
                        ContextCompat.startActivity(context, intent, Bundle.EMPTY)
                    }
                }
            }
    )
}
```

- On obtient donc ce résultat :



- Dans la classe Carré

```
class Carré : Forme2D {
    private var cote: Double

    constructor(nom: String?, perimetre: Double, surface: Double, cote: Double) : super(
        nom,
        perimetre,
        surface
    ) {
        this.cote = cote
    }

    constructor(cote: Double) : super( nom: "carré", perimetre: 0.0, surface: 0.0) {
        this.cote = cote
    }

    override fun calculPerimetre(): Double {
        return cote * 4
    }

    override fun calculSurface(): Double {
        return cote * cote
    }
}
```

- Voici un bout de code permettant d'ajouter un carré, il faut mettre exactement le nom que le fichier PHP qu'on veut utiliser

```
//Fonction pour insérer des données
suspend fun addCarré(lgcote: String): JSONObject {
    //La variable JSON qui contient la réponse de l'ajout reçue du serveur WEB
    var job = JSONObject()

    //les paramètres à envoyer en post au serveur web pour insérer dans la BDD
    val uriBuilder = Uri.Builder() Uri.Builder()
        .appendQueryParameter("lgcote", lgcote) Uri.Builder!
        .build()

    //Formatage des paramètres
    val params = uriBuilder.toString().replace(
        oldValue: "?",
        newValue: ""
    ) // Remove the "?" from the beginning of the parameters ?name=Jack&salary=8054&age=45
    val postData = params.toByteArray(StandardCharsets.UTF_8)

    //on lance le job indépendant qui insère les données
    val insertionJob = GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope
        Log.d( tag: "dans le scope", msg: "globalscope")
        //url contient l'adresse du site à exécuter
        val url = URL( spec: "http://" + ProjectConstance.URL_SERVEUR + "/geo/add_carré.php")
        Log.e( tag: "-- URL SAISIE ---", url.toString())
        //le traitement de la connection au site
        val httpURLConnection = url.openConnection() as HttpURLConnection
        httpURLConnection.requestMethod = "POST"
        httpURLConnection.setRequestProperty(
            "Content-Type",
            "application/x-www-form-urlencoded"
        ) // The format of the content we're sending to the server
        httpURLConnection.setRequestProperty(
            "Accept",
            "application/json"
        ) // The format of response we want to get from the server
        httpURLConnection.doInInput = true
    }
}
```

- Pour récupérer les données du carré :

```
//Fonction pour récupérer les carrés
suspend fun getCarres(): StringBuilder {
    //la variable qui contiendra tous les carrés
    val response = StringBuilder()

    //on lance je job indépendant qui insère les données
    val recuperationJob = GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope
        //url contient l'adresse du site à exécuter
        val url = URL(spec: "http://"+ ProjectConstance.URL_SERVEUR +"/geo/get_carres.php")
        Log.e( tag: "-- URL get SAISIE ---", url.toString())
        //le traitement de la connection au site
        val httpURLConnection = url.openConnection() as HttpURLConnection
        httpURLConnection.setRequestProperty(
            "Accept",
            "application/json"
        ) // The format of response we want to get from the server
        httpURLConnection.requestMethod = "GET"
        httpURLConnection.doInput = true
        httpURLConnection.doOutput = false
        // Check if the connection is successful
        val responseCode = httpURLConnection.responseCode
        Log.d( tag: "reponse code", msg: "reader: " + responseCode.toString())
        if (responseCode == HttpURLConnection.HTTP_OK) {
            withContext(Dispatchers.Main) { this: CoroutineScope
                //on récupère la réponse du serveur web concernant la recuperation
                val `in`: InputStream = BufferedInputStream(httpURLConnection.inputStream)
                val reader = BufferedReader(InputStreamReader(`in`))
                Log.d( tag: "JSON Parser", msg: "reader: " + reader.toString())
                var line: String?
                while (reader.readLine().also { line = it } != null) {
                    response.append(line)
                }
                Log.d( tag: "JSON Parser", msg: "reponse: " + response.toString()) ^withContext
            }
        } else {
            Log.e( tag: "HTTPURLCONNECTION_ERROR", responseCode.toString())
        }
    }
    //On attend que la recuperation se fasse pour passer à la suite
    recuperationJob.join()
}
```

- Pour supprimer le carré :

```
//Fonction pour supp des données
suspend fun suppCarre(id: String): JSONObject {
    //La variable JSON qui contient la réponse de l'ajout reçue du serveur WEB
    var obj = JSONObject()

    //les paramètres à envoyer en post au serveur web pour insérer dans la BDD
    val uriBuilder = Uri.Builder() UriBuilder
        .appendQueryParameter("id", id) UriBuilder
        .build()

    //Formatage des paramètres
    val params = uriBuilder.toString().replace(
        oldValue: "?",
        newValue: ""
    ) // Remove the "?" from the beginning of the parameters ?name=Jack&salary=8054&age=45
    val postData = params.toByteArray(StandardCharsets.UTF_8)

    //on lance je job indépendant qui insère les données
    val suppJob = GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope
        Log.d( tag: "dans le scope", msg: "GlobalScope")
        //url contient l'adresse du site à exécuter
        val url = URL(spec: "http://" + ProjectConstance.URL_SERVEUR +"/geo/sup_carre.php")

        //le traitement de la connection au site
        val httpURLConnection = url.openConnection() as HttpURLConnection
        httpURLConnection.requestMethod = "POST"
        httpURLConnection.setRequestProperty(
            "Content-Type",
            "application/x-www-form-urlencoded"
        ) // The format of the content we're sending to the server
        httpURLConnection.setRequestProperty(
            "Accept",
            "application/json"
        ) // The format of response we want to get from the server
        httpURLConnection.doInput = true
        httpURLConnection.doOutput = true
        val dataOutputStream = DataOutputStream(httpURLConnection.getOutputStream)
        dataOutputStream.write(postData)

        // Check if the connection is successful
    }
```

- Pour supprimer plusieurs carrés en même temps :

```
//Fonction pour supp des données
suspend fun suppAllCarre(): JSONObject {
    //La variable JSON qui contient la réponse de l'ajout reçue du serveur WEB
    var job1 = JSONObject()

    //les paramètres à envoyer en post au serveur web pour insérer dans la BDD
    val urlBuilder = Uri.Builder().appendQueryParameter("all", "1").build()

    //Formatage des paramètres
    val params = urlBuilder.toString().replace(
        oldValue: "?",
        newValue: ""
    ) // Remove the "?" from the beginning of the parameters ?name=Jack&salary=8054&age=45
    val postData = params.toByteArray(StandardCharsets.UTF_8)

    //on lance le job indépendant qui insère les données
    val suppJob = GlobalScope.launch(Dispatchers.IO) { thisCoroutineScope
        Log.d(tag: "dans le scope", msg: "globalscope")
        //url contient l'adresse du site à exécuter
        val url = URL(spec: "http://" + ProjectConstance.URL_SERVEUR + "/geo/supAll_carres.php")

        //le traitement de la connection au site
        val httpURLConnection = url.openConnection() as HttpURLConnection
        httpURLConnection.requestMethod = "POST"
        httpURLConnection.setRequestProperty(
            "Content-Type",
            "application/x-www-form-urlencoded"
        ) // The format of the content we're sending to the server
        httpURLConnection.setRequestProperty(
            "Accept",
            "application/json"
        ) // The format of response we want to get from the server
        httpURLConnection.doInput = true
        httpURLConnection.doOutput = true
        val dataOutputStream = DataOutputStream(httpURLConnection.getOutputStream)
        dataOutputStream.write(postData)

        // Check if the connection is successful
        val responseCode = httpURLConnection.responseCode
    }
}
```

- Pour ajouter plusieurs carrés en même temps :

```
suspend fun addXCarre(nb: String): JSONObject {
    //La variable JSON qui contient la réponse de l'ajout reçue du serveur WEB
    var job1 = JSONObject()

    //les paramètres à envoyer en post au serveur web pour insérer dans la BDD
    val urlBuilder = Uri.Builder().appendQueryParameter("nb", nb).build()

    //Formatage des paramètres
    val params = urlBuilder.toString().replace(
        oldValue: "?",
        newValue: ""
    ) // Remove the "?" from the beginning of the parameters ?name=Jack&salary=8054&age=45
    val postData = params.toByteArray(StandardCharsets.UTF_8)

    //on lance le job indépendant qui insère les données
    val suppJob = GlobalScope.launch(Dispatchers.IO) { thisCoroutineScope
        Log.d(tag: "dans le scope", msg: "globalscope")
        //url contient l'adresse du site à exécuter
        val url = URL(spec: "http://" + ProjectConstance.URL_SERVEUR + "/geo/addX_carre.php")

        //le traitement de la connection au site
        val httpURLConnection = url.openConnection() as HttpURLConnection
        httpURLConnection.requestMethod = "POST"
        httpURLConnection.setRequestProperty(
            "Content-Type",
            "application/x-www-form-urlencoded"
        ) // The format of the content we're sending to the server
        httpURLConnection.setRequestProperty(
            "Accept",
            "application/json"
        ) // The format of response we want to get from the server
        httpURLConnection.doInput = true
        httpURLConnection.doOutput = true
        val dataOutputStream = DataOutputStream(httpURLConnection.getOutputStream)
        dataOutputStream.write(postData)

        // Check if the connection is successful
        val responseCode = httpURLConnection.responseCode
    }
}
```

- La PageCarre

```
class PageCarre : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            GeoCalculatorTheme {
                // A surface container using the 'background' color from the
                if (Build.VERSION.SDK_INT > 9) {
                    val policy = ThreadPolicy.Builder().permitAll().build()
                    StrictMode.setThreadPolicy(policy)
                }
                PageCarreStart()
            }
        }
    }
}
```

- Le code permettant de calculer le périmètre et la surface du carré grâce à la longueur du côté.

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Preview(showBackground = true)
@Composable
fun PageCarreStart() {

    //Pour les changement d'activité et les messages Toast
    val context = LocalContext.current

    //les variables à récupérer/modifier
    val txtPerimetre = remember { mutableStateOf(TextFieldValue(text: "Calculez moi")) }
    val txtSurface = remember { mutableStateOf(TextFieldValue(text: "Calculez moi")) }
    var coteInput by remember { mutableStateOf(value: "") }
```

- Le bout de code qui permet d'afficher toutes les autres formes en bas de la page pour qu'on puisse cliquer dessus pour afficher directement leur page au lieu de retourner sur la page d'accueil.

```
Scaffold(modifier = Modifier,
    bottomBar = {
        BottomAppBar(actions = {
            Image(
                painter = painterResource(id = R.drawable.rectangle),
                contentDescription = "rectangle",
                contentScale = ContentScale.Crop,
                modifier = Modifier
                    .size(60.dp)
                    .clip(RoundedCornerShape(16.dp))
                    .clickable {
                        //on lance une nouvelle activité PageCarre
                        context.startActivity(Intent(context, PageRectangle::class.java))
                    }
            )
            Image(
                painter = painterResource(id = R.drawable.cercle),
                contentDescription = "cercle",
                contentScale = ContentScale.Crop,
                modifier = Modifier
                    .size(60.dp)
                    .clip(RoundedCornerShape(16.dp))
                    .clickable {
                        //on lance une nouvelle activité PageCarre
                        context.startActivity(Intent(context, PageCercle::class.java))
                    }
            )
            Image(
                painter = painterResource(id = R.drawable.triangle),
                contentDescription = "triangle",
                contentScale = ContentScale.Crop,
                modifier = Modifier
                    .size(60.dp)
                    .clip(RoundedCornerShape(16.dp))
                    .clickable {
                        //on lance une nouvelle activité PageCarre
                        context.startActivity(Intent(context, PageTriangle::class.java))
                    }
            )
        })
    }
)
```

- Le code permettant d'afficher le titre de la page du carré :

```
Row(
    modifier = Modifier
        .fillMaxWidth()
        .padding(top = 10.dp),
    verticalAlignment = Alignment.CenterVertically
) { this: RowScope
    Text(
        text = stringResource("Bienvenue sur la page du carré"),
        color = Color.Red,
        fontSize = 30.sp,
        textAlign = TextAlign.Center
}
```

- Le code qui permet d'afficher le carré central :

```
Column{
    modifier = Modifier
        .padding(start = 30.dp, top = 0.dp, end = 30.dp, bottom = 30.dp)
        .fillMaxSize()
        .wrapContentSize(Alignment.Center)
} { this: ColumnScope

    Row(
        Modifier.fillMaxWidth(),
        Arrangement.Center,
        Alignment.CenterVertically,
    ) { this: RowScope
        Image(
            painter = painterResource(R.drawable.carre),
            contentDescription = "image carre",
            contentScale = ContentScale.Fit,
            modifier = Modifier
                .size(150.dp)
                .border(
                    BorderStroke(5.dp, Color.Gray),
                    RectangleShape
                )
        )
    }
}
```

- Le code qui nous permettent d'entrer la longueur du côté :

```
//Espacement
Spacer(modifier = Modifier.height(40.dp))

Row(
    modifier = Modifier
        .fillMaxWidth(),
    verticalAlignment = Alignment.CenterVertically
) { this: RowScope
    Text(
        text = "Côté :",
        color = Color.Black,
        //modifier = Modifier.height(60.dp),
        textAlign = TextAlign.Center
    )

    Spacer(modifier = Modifier.padding(10.dp))
    //Le input pour le poids

    OutlinedTextField(
        value = coteInput,
        onValueChange = { coteInput = it },
        label = { Text(text: "Longueur du côté") },
        keyboardOptions = KeyboardOptions.Default.copy(keyboardType = KeyboardType.Number)
)
}
```

- Ligne de code qui permet de calculer le périmètre automatiquement :

```
//Espacement
Spacer(modifier = Modifier.height(40.dp))

//Ligne pour le périmètre
Row(
    modifier = Modifier
        .fillMaxWidth(),
    verticalAlignment = Alignment.CenterVertically,
    horizontalArrangement = Arrangement.SpaceAround
) { this: RowScope
    //Calculer le périmètre
    ElevatedButton(onClick = {
        var carre = Carré(nom: "carre", périmètre: 0.0, surface: 0.0, coteInput.toDouble())
        val valPérimètre = carre.calculPérimètre()
        carre.périmètre = valPérimètre
        txtPérimètre.value = (TextFieldValue(carre.périmètre.toString()))
    })
    this: RowScope
    Text(
        text: "Calculer le périmètre",
        color = Color.Black
    )
}

//Espacement
Spacer(modifier = Modifier.padding(25.dp))

//Text pour le résultat du périmètre
Text(
    text = txtPérimètre.value.text,
    color = Color.Gray
)
```

- Ligne de code permettant de calculer la surface :

```
//Espace
Spacer(modifier = Modifier.height(10.dp))

//Ligne pour la surface
Row(
    modifier = Modifier
        .fillMaxWidth(),
    verticalAlignment = Alignment.CenterVertically,
    horizontalArrangement = Arrangement.SpaceAround
) { this: RowScope
    //Calculer la surface
    ElevatedButton(onClick = {
        var carre = Carré( nom: "carre", périmètre: 0.0, surface: 0.0, coteInput.toDouble())
        val valSurface = carre.calculSurface()
        carre.surface = valSurface
        txtSurface.value = (TextFieldValue(carre.surface.toString()))
    })
    this: RowScope
    Text(
        text: "Calculer la surface",
        color = Color.Black
    )
}

//Espace
Spacer(modifier = Modifier.padding(25.dp))

//Afficher la surface
Text(
    text = txtSurface.value.text,
    color = Color.Gray,
    textAlign = TextAlign.Right
)
```

- Ligne de code qui permet d'enregistrer un carré :

```
//Ligne pour les autres boutons
Row(
    modifier = Modifier
        .fillMaxWidth(),
    verticalAlignment = Alignment.CenterVertically,
    horizontalArrangement = Arrangement.SpaceEvenly
) { this: RowScope
    //Sauvegarder dans la base
    IconButton(onClick = {
        CoroutineScope(Dispatchers.Main).launch { this: CoroutineScope
            val job: JSONObject = addCarre(coteInput)
            withContext(Dispatchers.Main) { this: CoroutineScope
                Log.i( tag: "après insert", job.toString())
                if (job.getString( name: "status").equals("Success")) {
                    println("affiche toast")
                    Handler(Looper.getMainLooper()).post {
                        Toast.makeText(context, text: "insert ok", Toast.LENGTH_LONG).show()
                    }
                } else {
                    Handler(Looper.getMainLooper()).post {
                        Toast.makeText(context, text: "Erreur d'insertion", Toast.LENGTH_LONG)
                            .show()
                    }
                }
            }
        }
    })
    Icon(
        Icons.Filled.FavoriteBorder,
        contentDescription = "Localized description",
    )
}
```

- Ligne de code permettant d'afficher le carré enregistrer :

```
//Espacement
Spacer(modifier = Modifier.padding(25.dp))

//Afficher la liste
IconButton(onClick = {
    CoroutineScope(Dispatchers.Default).launch { this: CoroutineScope
        val strB: StringBuilder = getCarres()
        withContext(Dispatchers.Main) { this: CoroutineScope
            Log.i( tag: "apres get carre", strB.toString())
            val intent = Intent(context, AfficheCarres::class.java)
            // now by putExtra method put the value in key, value pair key is
            // message_key by this key we will receive the value, and put the string
            intent.putExtra( name: "carre", strB.toString())
            // start the Intent
            startActivity(context, intent, Bundle.EMPTY)
        }
    }
}) {
    Icon(
        Icons.Filled.List,
        contentDescription = "Localized description",
    )
}
```

- Résultat de la page carré :



- AfficheCarre

```
//On déclare une classe qui contient les données de Carre à afficher
private data class DataCarre(val id: Int, val lgcote: Double)
```

```
//On cree une variable qui contient la liste des carrés à afficher
private var lesCarres = listOf<DataCarre>()
```

```
class AfficheCarres : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(
            GeoCalculatorTheme {
                //on crée une liste de carres vide à chaque lancement de la page
                lesCarres = emptyList()

                //on récupere les données envoyées ici dans la variable intent
                val intent = intent

                //on transforme les données recues au format JSON
                if (JSONObject(
                    intent.getStringExtra(name: "carre").toString()
                    .getString(name: "status").equals("Failed")
                ) {
                    DisplayNoDataCarres()
                } else {
                    //on transforme les données recues au format JSON
                    val lesCarresJSON =
                        JSONArray(
                            JSONObject(
                                intent.getStringExtra(name: "carre").toString()
                                .getString(name: "message")
                            )
                        )
                    Log.i(tag: "apres get carre", lesCarresJSON.toString())

                    //A partir du JSON on crée la liste d'objets DataCarre à afficher
                    for (i in 0 ≤ until < lesCarresJSON.length()) {
                        //on récupere chaque element JSON qu'on stocke dans la variable item
                        val item = lesCarresJSON.getString(i)

                        //On rajoute au fur et a mesure l'item dans la liste à afficher
                        lesCarres += DataCarre(
                            JSONObject(item).get("id").toString().toInt(),
                            JSONObject(item).get("lgcote").toString().toDouble()
                        )
                    }
                    //on affiche pour vérifier le nombre
                    println(lesCarres.size)

                    DisplayList()
                }
            }
        )
    }
}
```

- Dans cette capture d'écran il y a la taille des colonnes, le titre et le bouton retour.

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@Preview(showBackground = true)
@OptIn(ExperimentalMaterial3Api::class)
@Composable
private fun DisplayList() {

    //Variable contenant la proportion de taille des colonnes
    val column1Weight = .1f
    val column2Weight = .3f
    val column3Weight = .2f
    val column4Weight = .2f

    val context = LocalContext.current
    var nbInput by remember { mutableStateOf(value: "") }
    //La top bar
    Scaffold(
        modifier = Modifier,
        topBar = {
            TopAppBar(
                title = [
                    Text(
                        text: "Tous les carres enregistrés",
                        maxLines = 1,
                        overflow = TextOverflow.Ellipsis
                    )
                ],
                actions = { this: RowScope
                    ElevatedButton(
                        onClick = {
                            //on appel la fonction dans on click dessus
                            val intent = Intent(context, PageCarre::class.java)
                            ContextCompat.startActivity(context, intent, Bundle.EMPTY)
                        }
                    ) { this: RowScope
                        Text(
                            text: "Retour",
                            color = Color.Red
                        )
                    }
                },
            )
        }
    )
}
```

- ça permet de supprimer tous les carrés enregistrés :

```
    },
    bottomBar = {
        BottomAppBar(modifier = Modifier
            .fillMaxWidth(),
            actions = { this: RowScope
                Button(
                    onClick = {
                        val builder: AlertDialog.Builder = AlertDialog.Builder(context)
                        builder
                            .setMessage("Voulez-vous vraiment supprimer tous les carrés ?")
                            .setIcon(R.drawable.ic_delete)
                            .setTitle("Suppression des carrés")
                            .setPositiveButton(text: "Oui") { dialog, which ->
                                println("on supp")
                                val suppAllJob = CoroutineScope(Dispatchers.Main).launch { this: CoroutineScope
                                    val job: JSONObject =
                                        suppAllCarre()
                                    withContext(Dispatchers.Main) { this: CoroutineScope
                                        Log.i(tag: "après supp", job.toString())
                                        if (job)
                                            .getString(name: "status")
                                            .equals("Success")
                                    } {
                                        println("affiche toast")
                                        Handler(Looper.getMainLooper()).post {
                                            Toast
                                                .makeText(
                                                    context,
                                                    text: "Supp ok",
                                                    Toast.LENGTH_LONG
                                                )
                                                .show()
                                        }
                                    }
                                }
                            }
                        }
                    }
                )
            }
        )
    }
}
```

- Si les carrés n'ont pas été supprimés ça affiche erreur de suppression grâce à ce code :

```
    .show()
}
} else {
    Handler(Looper.getMainLooper()).post {
        Toast
            .makeText(
                context,
                text: "Erreur de suppression",
                Toast.LENGTH_LONG
            )
            .show()
}
}
```

- Permet d'ajouter plusieurs carrés en même temps :

```
CoroutineScope(Dispatchers.Default).launch { this: CoroutineScope
    //on attend que le job de suppression se fasse
    addXJob.join()
    val strB: StringBuilder = getCarres()
    withContext(Dispatchers.Main) { this: CoroutineScope
        Log.i( tag: "apres supp carre", strB.toString())

        val intent =
            Intent(context, AfficheCarres::class.java)
        // now by putExtra method put the value in key, value pair key is
        // message_key by this key we will receive the value, and put the string
        intent.putExtra( name: "carre", strB.toString())
        // start the Intent
        ContextCompat.startActivity(
            context,
            intent,
            Bundle.EMPTY
        )
    }
},
),
) { this: RowScope
    Text(text = "Ajouter")
}
},
},
```

Pour afficher le tableau :

```
//Le big tableau de données
LazyColumn(
    Modifier.padding(start = 30.dp, top = 60.dp, end = 30.dp, bottom = 30.dp)
) { this:LazyListScope

    //entête du tableau
    item { this:LazyItemScope
        Row(
            Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.Center,
            verticalAlignment = Alignment.CenterVertically
        ) { this:RowScope
            TableCell(
                text = "Id",
                weight = column1Weight,
                alignment = TextAlign.Left,
                title = true
            )
            TableCell(
                text = "Longueur du côté",
                weight = column2Weight,
                alignment = TextAlign.Right,
                title = true
            )
            TableCell(
                text = "Périmètre",
                weight = column3Weight,
                alignment = TextAlign.Center,
                title = true
            )
            TableCell(
                text = "Surface",
                weight = column4Weight,
                alignment = TextAlign.Center,
                title = true
            )
        }
    }
    //Le trait rouge
    Divider(
        color = Color.Red,
        thickness = 5.dp,
        modifier = Modifier
    )
}
```

- Pour supprimer un seul carré

```
itemsIndexed(lesCarres) { this:LazyItemScope _, unCarre ->
    Row(
        Modifier.fillMaxWidth()
            .selectable(
                selected = unCarre.id == 1,
                onClick = {
                    val builder: AlertDialog.Builder = AlertDialog.Builder(context)
                    builder
                        .setMessage("Voulez-vous vraiment supprimer le carré d'ID : " + unCarre.id)
                        .setIcon(R.drawable.ic_delete)
                        .setTitle("Suppression d'un carré")
                        .setPositiveButton(text = "Oui") { dialog, which ->
                            println("on supp")
                            val suppJob = CoroutineScope(Dispatchers.Main).launch { this:CoroutineScope
                                val job: JSONObject =
                                    suppCarre(unCarre.id.toString())
                                withContext(Dispatchers.Main) { this:CoroutineScope
                                    Log.i( tag: "après Insert", job.toString())
                                    if (job)
                                        .getString( name: "status")
                                        .equals("Success")
                                    ) {
                                        println("affiche toast")
                                        Handler(Looper.getMainLooper()).post {
                                            Toast
                                                .makeText(
                                                    context,
                                                    text: "Supp ok",
                                                    Toast.LENGTH_LONG
                                                )
                                                .show()
                                        }
                                    } else {
                                        Handler(Looper.getMainLooper()).post {
                                            Toast
                                                .makeText(
                                                    context,
                                                    text: "Erreur de suppression",
                                                    Toast.LENGTH_LONG
                                                )
                                                .show()
                                        }
                                    }
                                }
                            }
                        }
                    )
                }
            )
    )
}
```

- Lorsqu'il y a aucune donnée à afficher :

```
@Preview(showBackground = true)
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DisplayNoDataCarres() {
    ...
    ) { it: PaddingValues
    }
    Row(
        modifier = Modifier
            .fillMaxSize()
            .padding(20.dp),
        verticalAlignment = Alignment.CenterVertically,
    ) { this:RowScope
        Text(
            text = "Aucune donnée à afficher",
            color = Color.Red,
            fontSize = 30.sp,
            textAlign = TextAlign.Center
        )
    }
}
```

- Résultat de AfficheCarre :

A screenshot of a smartphone displaying a list of registered squares. The screen shows a header "Tous les carres enregistrés" and a "Retour" button. Below is a table with the following data:

ID	Longueur du côté	Périmètre	Surface
31	6.0	24.0	36.0
32	2.0	8.0	4.0
33	4.0	16.0	16.0
34	6.0	24.0	36.0
35	8.0	32.0	64.0
36	10.0	40.0	100.0
37	60.0	240.0	3600.0

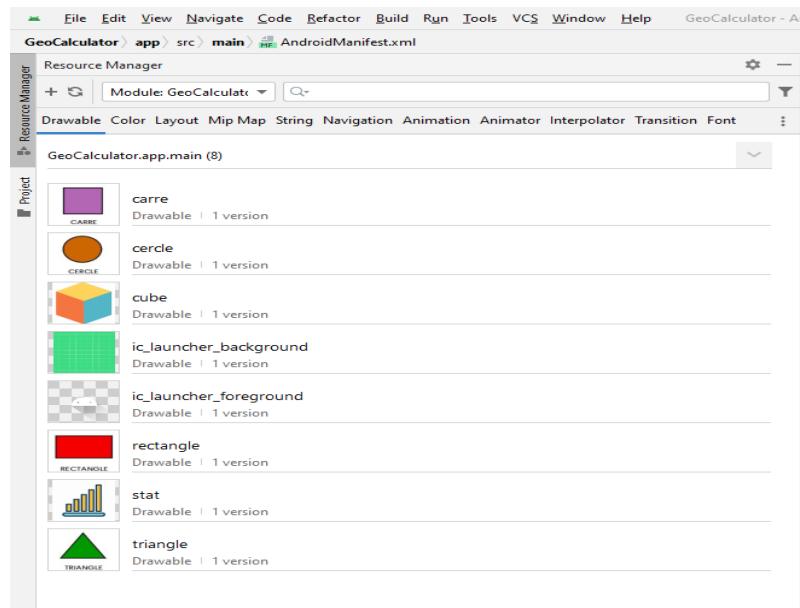
At the bottom, there are buttons for "Supprimer tout" (Delete all), "Nb : [input field]", and "Ajouter" (Add).

- Dans AndroidManifest on doit retrouver ceci :

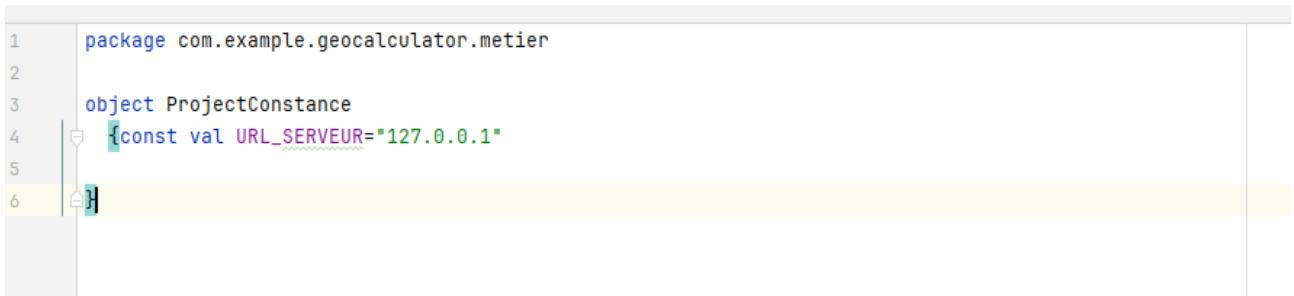
```
<activity
    android:name=".PageCarre"
    android:exported="false"
    android:label="PageCarre"
    android:theme="@style/Theme.GeoCalculator" />
<activity
    android:name=".PageCercle"
    android:exported="false"
    android:label="PageCercle"
    android:theme="@style/Theme.GeoCalculator" />
<activity
    android:name=".PageTriangle"
    android:exported="false"
    android:label="PageTriangle"
    android:theme="@style/Theme.GeoCalculator" />
<activity
    android:name=".AfficheCarres"
    android:exported="false"
    android:label="AfficheCarres"
    android:theme="@style/Theme.GeoCalculator" />
<activity
    android:name=".PageCube"
    android:exported="false"
    android:label="PageCube"
    android:theme="@style/Theme.GeoCalculator" />
<activity
    android:name=".MainActivity"
    android:exported="true"
    android:label="GeoCalculator"
    android:theme="@style/Theme.GeoCalculator">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>

</application>
</manifest>
```

- Pour récupérer les images on doit le cliquer sur ressources manager dans Android > +> import drawable et choisir le fichiers



- Dans ProjectConstance mettre l'adresse IP de sa machine qui permet de récupérer les données :

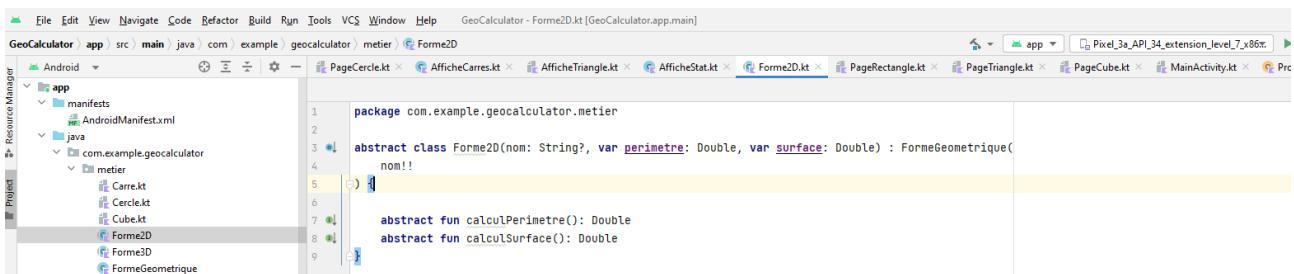


```

1 package com.example.geocalculator.metier
2
3 object ProjectConstance
4     {const val URL_SERVEUR="127.0.0.1"
5
6 }

```

- Dans Forme 2D

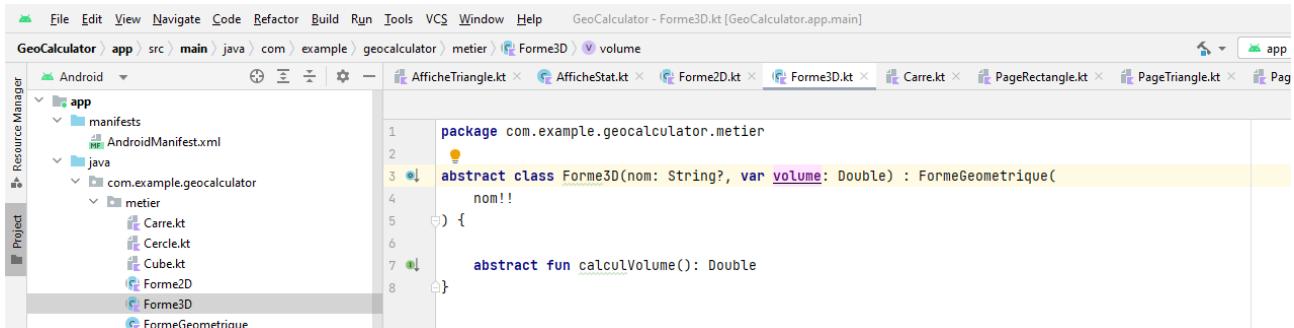


```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help GeoCalculator - Forme2D.kt [GeoCalculator.app.main]
GeoCalculator > app > src > main > java > com > example > geocalculator > metier > Forme2D
Resource Manager Project
Android app manifests AndroidManifest.xml
java com.example.geocalculator metier
Carre.kt Cercle.kt Cube.kt Forme2D Forme3D FormeGeometrique
Forme2D.kt
1 package com.example.geocalculator.metier
2
3 abstract class Forme2D(nom: String?, var perimetre: Double, var surface: Double) : FormeGeometrique(
4     nom!!
5
6
7     abstract fun calculPerimetre(): Double
8     abstract fun calculSurface(): Double
9

```

- Dans Forme 3D

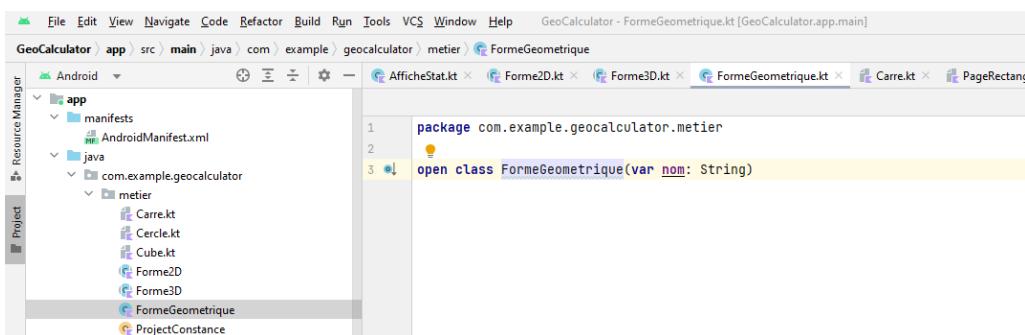


```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help GeoCalculator - Forme3D.kt [GeoCalculator.app.main]
GeoCalculator > app > src > main > java > com > example > geocalculator > metier > Forme3D
Resource Manager Project
Android app manifests AndroidManifest.xml
java com.example.geocalculator metier
Carre.kt Cercle.kt Cube.kt Forme2D Forme3D FormeGeometrique
Forme3D.kt
1 package com.example.geocalculator.metier
2
3 abstract class Forme3D(nom: String?, var volume: Double) : FormeGeometrique(
4     nom!!
5
6
7     abstract fun calculVolume(): Double
8

```

- Dans Forme Géométrique



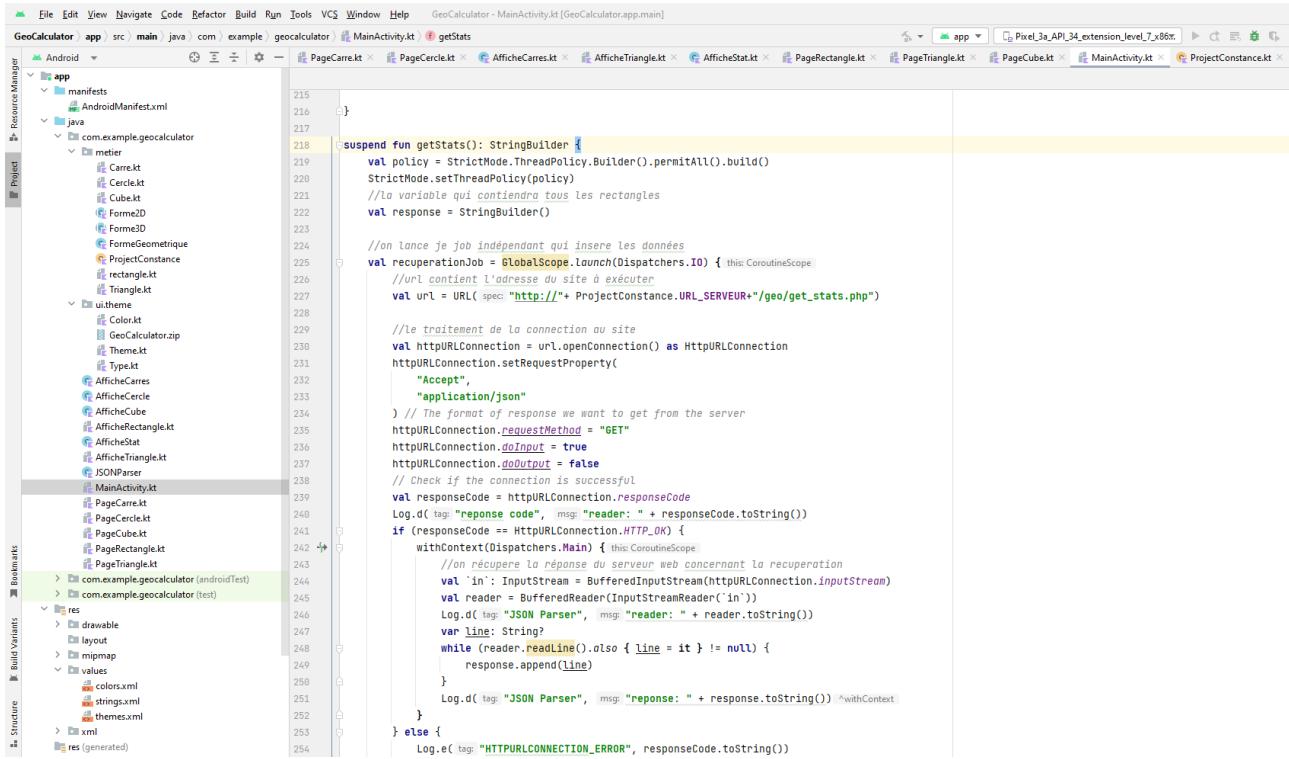
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help GeoCalculator - FormeGeometrique.kt [GeoCalculator.app.main]
GeoCalculator > app > src > main > java > com > example > geocalculator > metier > FormeGeometrique
Resource Manager Project
Android app manifests AndroidManifest.xml
java com.example.geocalculator metier
Carre.kt Cercle.kt Cube.kt Forme2D Forme3D FormeGeometrique
ProjectConstance
FormeGeometrique.kt
1 package com.example.geocalculator.metier
2
3 open class FormeGeometrique(var nom: String)

```

f) Stat

- Dans MainActivity copier-coller la fonction pour récupérer les données



```
215
216
217
218    suspend fun getStats(): String {
219        val policy = StrictMode.ThreadPolicy.Builder().permitAll().build()
220        StrictMode.setThreadPolicy(policy)
221        //La variable qui contiendra tous les rectangles
222        val response = StringBuilder()
223
224        //on lance le job indépendant qui insere les données
225        val recuperationJob = GlobalScope.launch(Dispatchers.IO) { thisCoroutineScope
226            //url contient l'adresse du site à exécuter
227            val url = URL(spec: "http://" + ProjectConstance.URL_SERVEUR+"/geo/get_stats.php")
228
229            //le traitement de la connection au site
230            val httpURLConnection = url.openConnection() as HttpURLConnection
231            httpURLConnection.setRequestProperty(
232                "Accept",
233                "application/json"
234            ) // The format of response we want to get from the server
235            httpURLConnection.requestMethod = "GET"
236            httpURLConnection.doInput = true
237            httpURLConnection.doOutput = false
238            // Check if the connection is successful
239            val responseCode = httpURLConnection.responseCode
240            Log.d(tag: "response code", msg: "reader: " + responseCode.toString())
241            if (responseCode == HttpURLConnection.HTTP_OK) {
242                withContext(Dispatchers.Main) { thisCoroutineScope
243                    //on récupère la réponse du serveur web concernant la recuperation
244                    val `in` : InputStream = BufferedInputStream(httpURLConnection.inputStream)
245                    val reader = BufferedReader(InputStreamReader(`in`))
246                    Log.d(tag: "JSON Parser", msg: "reader: " + reader.toString())
247                    var line: String?
248                    while (reader.readLine().also { line = it } != null) {
249                        response.append(line)
250                    }
251                    Log.d(tag: "JSON Parser", msg: "reponse: " + response.toString()) ^withContext
252                }
253            } else {
254                Log.e(tag: "HTTPURLCONNECTION_ERROR", responseCode.toString())
255            }
256        }
257    }
258}
```

- Créer AfficheStat

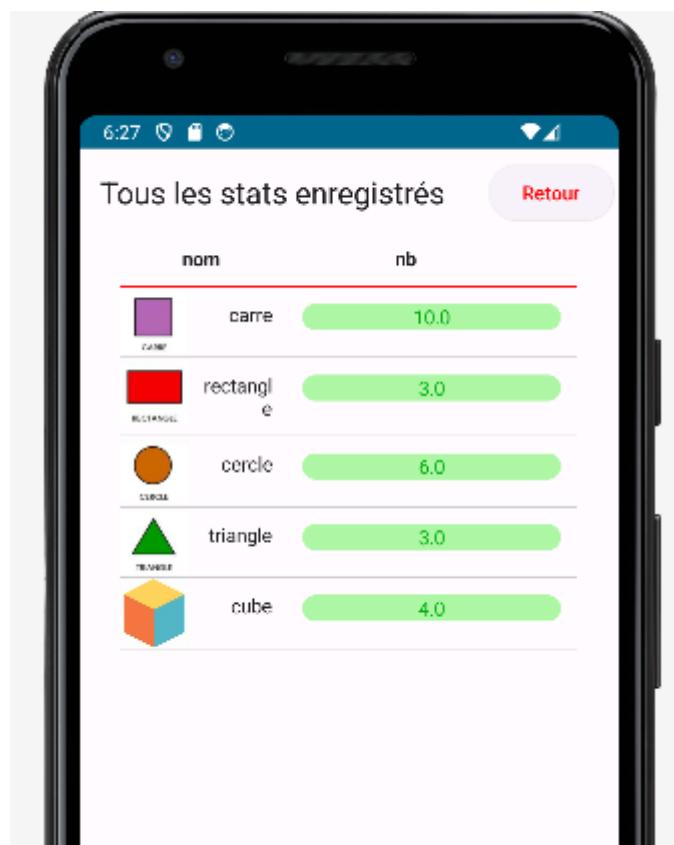
Copier coller un affiche d'une autre forme existante et modifier en fonction

```

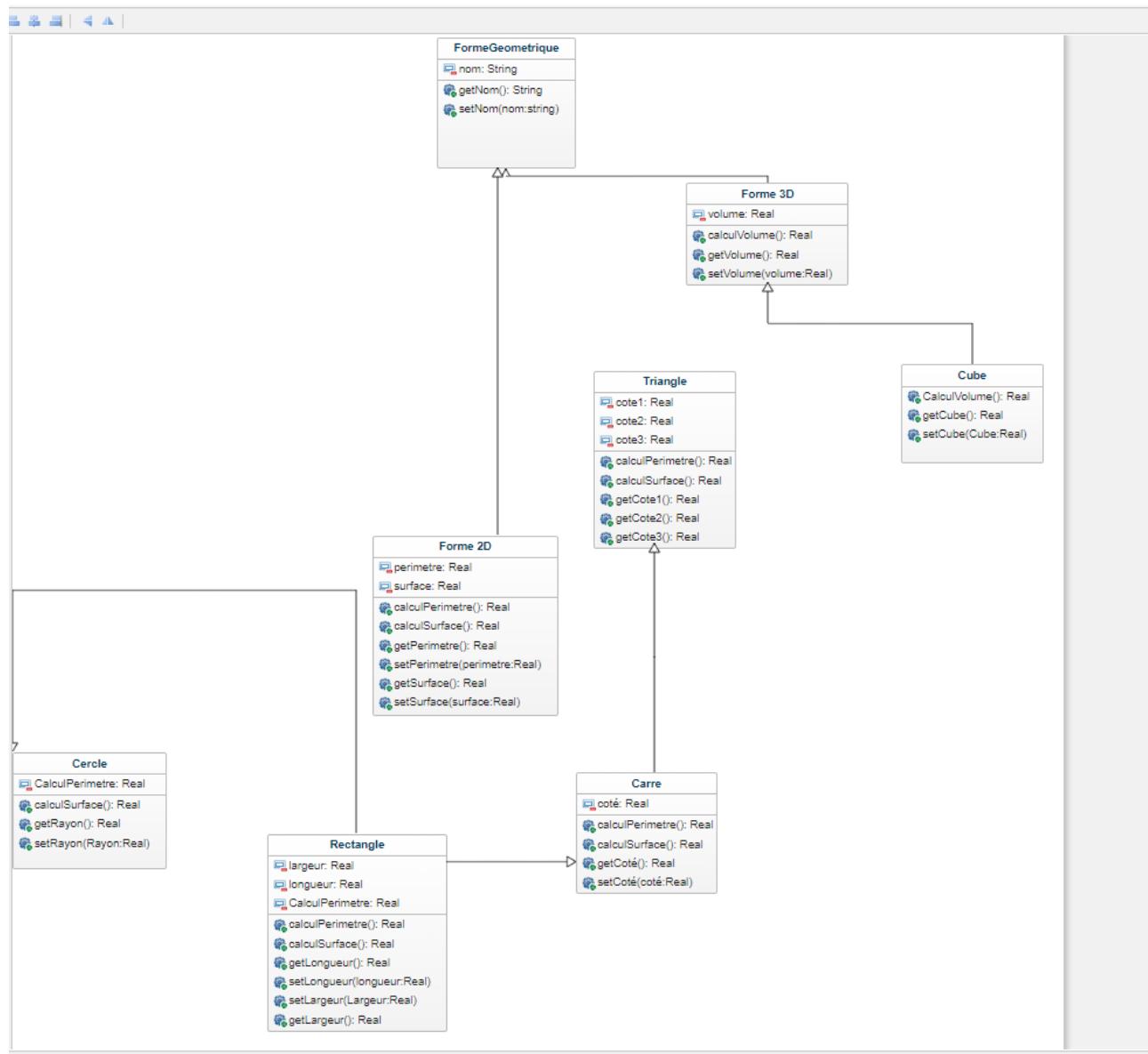
//Les lignes data
//Pour chaque element de lesCarres, on l'appelle unCarre et on fait le traitement
itemsIndexed{lesStats) { this:LazyItemScope _, unStat ->
    val img: Int
    val int: Intent
    when (unStat.nom) {
        "carre" -> {
            img=R.drawable.carre
            int=Intent(context, PageCarre::class.java)
        }
        "rectangle" -> {
            img=R.drawable.rectangle
            int=Intent(context, PageRectangle::class.java)
        }
        "cercle" -> {
            img=R.drawable.cercle
            int=Intent(context, PageCercle::class.java)
        }
        "triangle" -> {
            img=R.drawable.triangle
            int=Intent(context, PageTriangle::class.java)
        }
        "cube" -> {
            img=R.drawable.cube
            int=Intent(context, PageCube::class.java)
        }
        else->{
            img = R.drawable.stat
            int = Intent(context, MainActivity::class.java)
        }
    }
}

```

- Faire un run pour obtenir ce résultat

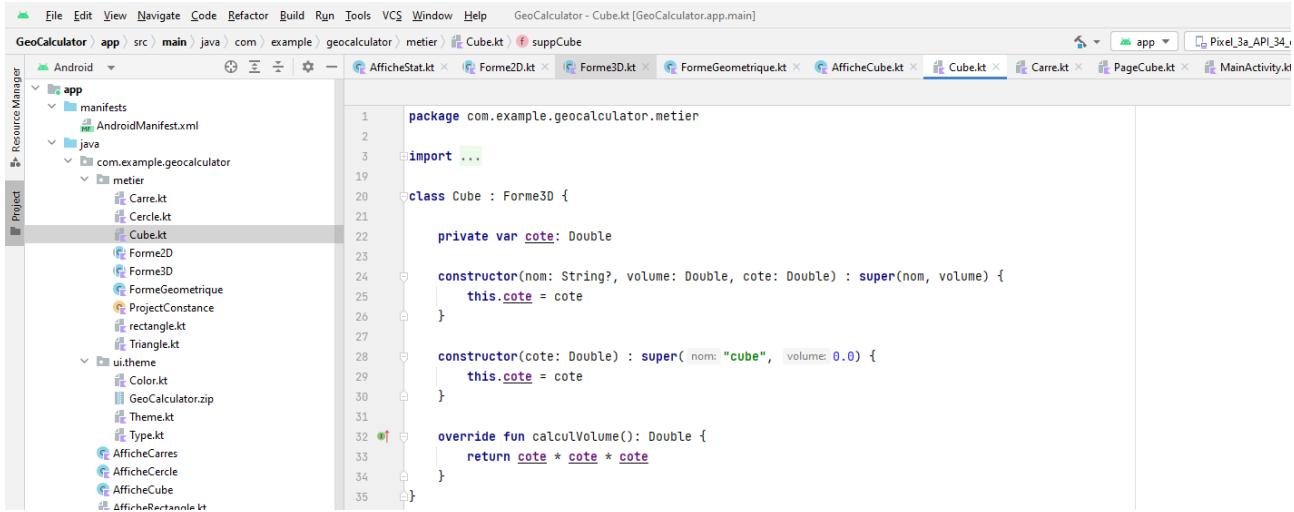


g) Diagramme de classe



h) Cube

- Dans la classe Cube



The screenshot shows the Android Studio interface with the code editor open to the `Cube.kt` file. The code defines a class `Cube` that implements the `Forme3D` interface. It has two constructors: one taking `nom`, `volume`, and `cote` as parameters, and another taking only `cote`. Both constructors call the super constructor with the provided parameters. The `calculVolume()` method returns the volume calculated by multiplying `cote` by itself three times.

```
package com.example.geocalculator.metier

import ...

class Cube : Forme3D {

    private var cote: Double

    constructor(nom: String?, volume: Double, cote: Double) : super(nom, volume) {
        this.cote = cote
    }

    constructor(cote: Double) : super( nom: "cube", volume: 0.0) {
        this.cote = cote
    }

    override fun calculVolume(): Double {
        return cote * cote * cote
    }
}
```

- Pour supprimer un cube faire comme dans la classe Carré :

```
//on lance je job indépendant qui insère les données
val suppJob = GlobalScope.launch(Dispatchers.IO) { thisCoroutineScope
    Log.d(tag: "dans le scope", msg: "globalScope")
    //url contient l'adresse du site à exécuter
    val url = URL(spec: "http://" + ProjectConstance.URL_SERVEUR + "/geo/sup_cube.php")

    //le traitement de la connection au site
    val httpURLConnection = url.openConnection() as HttpURLConnection
    httpURLConnection.requestMethod = "POST"
    httpURLConnection.setRequestProperty(
        "Content-Type",
        "application/x-www-form-urlencoded"
    ) // The format of the content we're sending to the server
    httpURLConnection.setRequestProperty(
        "Accept",
        "application/json"
    ) // The format of response we want to get from the server
    httpURLConnection.doInput = true
    httpURLConnection.doOutput = true
    val dataOutputStream = DataOutputStream(httpURLConnection.getOutputStream)
    dataOutputStream.write(postData)

    // Check if the connection is successful
    val responseCode = httpURLConnection.responseCode
    Log.d(tag: "response code", msg: "reader: " + responseCode.toString())
    if (responseCode == HttpURLConnection.HTTP_OK) {
        withContext(Dispatchers.Main) { thisCoroutineScope
            //on récupère la réponse du serveur web concernant l'insertion
            val in: InputStream = BufferedInputStream(httpURLConnection.getInputStream)
            val reader = BufferedReader(InputStreamReader(in))
            Log.d(tag: "JSON Parser", msg: "reader: " + reader.toString())
            val response = StringBuilder()
            var line: String?
            while (reader.readLine().also { line = it } != null) {
                response.append(line)
            }
        }
    }
}
```

- Pour supprimer tous les cubes :

```

//Fonction pour supp des données
suspend fun suppAllCube(): JSONObject {
    //La variable JSON qui contient la réponse de l'ajout reçue du serveur WEB
    var jobi = JSONObject()

    //les paramètres à envoyer en post au serveur web pour insérer dans la BDD
    val uriBuilder = Uri.Builder() UriBuilder
        .appendQueryParameter("all", "1") Uri.Builder
        .build()

    //Formatage des paramètres
    val params = uriBuilder.toString().replace(
        oldValue: "?",
        newValue: ""
    ) // Remove the "?" from the beginning of the parameters ?name=Jack&salary=8054&age=45
    val postData = params.toByteArray(StandardCharsets.UTF_8)

    //on lance le job indépendant qui insère les données
    val suppJob = GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope
        Log.d(tag: "dans le scope", msg: "globalscope")
        //url contient l'adresse du site à exécuter
        val url = URL(spec: "http://" + ProjectConstance.URL_SERVEUR +"/geo/supAll_cubes.php")

        //le traitement de la connection au site
        val httpURLConnection = url.openConnection() as HttpURLConnection
        httpURLConnection.requestMethod = "POST"
        httpURLConnection.setRequestProperty(
            "Content-Type",
            "application/x-www-form-urlencoded"
        ) // The format of the content we're sending to the server
        httpURLConnection.setRequestProperty(
            "Accept",
            "application/json"
        )
    }
}

```

- Pour ajouter plusieurs cubes en même temps :

```

01 suspend fun addXCube(nb: String): JSONObject {
02     //La variable JSON qui contient la réponse de l'ajout reçue du serveur WEB
03     var jobi = JSONObject()
04
05     //les paramètres à envoyer en post au serveur web pour insérer dans la BDD
06     val uriBuilder = Uri.Builder() UriBuilder
07         .appendQueryParameter("nb", nb) Uri.Builder
08         .build()

09     //Formatage des paramètres
10     val params = uriBuilder.toString().replace(
11         oldValue: "?",
12         newValue: ""
13     ) // Remove the "?" from the beginning of the parameters ?name=Jack&salary=8054&age=45
14     val postData = params.toByteArray(StandardCharsets.UTF_8)

15     //on lance le job indépendant qui insère les données
16     val suppJob = GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope
17         Log.d(tag: "dans le scope", msg: "globalscope")
18         //url contient l'adresse du site à exécuter
19
20         val url = URL(spec: "http://" + ProjectConstance.URL_SERVEUR +"/geo/addX_cube.php")

21         //le traitement de la connection au site
22         val httpURLConnection = url.openConnection() as HttpURLConnection
23         httpURLConnection.requestMethod = "POST"
24         httpURLConnection.setRequestProperty(
25             "Content-Type",
26             "application/x-www-form-urlencoded"
27         ) // The format of the content we're sending to the server
28         httpURLConnection.setRequestProperty(
29             "Accept",
30             "application/json"
31         ) // The format of response we want to get from the server
32         httpURLConnection.doInput = true
33         httpURLConnection.doOutput = true
34     }
35 }
36

```

- Pour récupérer les données du cube :

```
suspend fun getCube(): StringBuilder {
    //La variable qui contiendra tous les cubes
    val response = StringBuilder()

    //on lance le job indépendant qui insère les données
    val recuperationJob = GlobalScope.launch(Dispatchers.IO) { thisCoroutineScope
        //url contient l'adresse du site à exécuter
        val url = URL(spec: "http://"+ ProjectConstance.URL_SERVEUR +"/geo/get_cubes.php")

        //le traitement de la connection au site
        val httpURLConnection = url.openConnection() as HttpURLConnection
        httpURLConnection.setRequestProperty(
            "Accept",
            "application/json"
        ) // The format of response we want to get from the server
        httpURLConnection.requestMethod = "GET"
        httpURLConnection.doInput = true
        httpURLConnection.doOutput = false
        // Check if the connection is successful
        val responseCode = httpURLConnection.responseCode
        Log.d(tag: "reponse code", msg: "reader: " + responseCode.toString())
        if (responseCode == HttpURLConnection.HTTP_OK) {
            withContext(Dispatchers.Main) { thisCoroutineScope
                //on récupère la réponse du serveur web concernant la récupération
                val `in` : InputStream = BufferedInputStream(httpURLConnection.inputStream)
                val reader = BufferedReader(InputStreamReader(`in`))
                Log.d(tag: "JSON Parser", msg: "reader: " + reader.toString())
                var line: String?
                while (reader.readLine().also { line = it } != null) {
                    response.append(line)
                }
                Log.d(tag: "JSON Parser", msg: "reponse: " + response.toString()) ^withContext
            }
        } else {
            Log.e(tag: "HTTPURLCONNECTION_ERROR", responseCode.toString())
        }
    }
}
```

- Pour ajouter un cube :

```
//Fonction pour insérer des données
suspend fun addCube(lgcote: String): JSONObject {
    //La variable JSON qui contient la réponse de l'ajout reçue du serveur WEB
    var obj1 = JSONObject()

    //les paramètres à envoyer en post au serveur web pour insérer dans la BDD
    val uriBuilder = Uri.Builder() UriBuilder
        .appendQueryParameter("lgcote", lgcote) UriBuilder!
        .build()

    //Formatage des paramètres
    val params = uriBuilder.toString().replace(
        oldValue: "?",
        newValue: ""
    ) // Remove the "?" from the beginning of the parameters ?name=Jack&salary=8054&age=45
    val postData = params.toByteArrayList(StandardCharsets.UTF_8)

    //on lance le job indépendant qui insère les données
    val insertionJob = GlobalScope.launch(Dispatchers.IO) { thisCoroutineScope
        Log.d(tag: "dans le scope", msg: "GlobalScope")
        //url contient l'adresse du site à exécuter
        val url = URL(spec: "http://"+ ProjectConstance.URL_SERVEUR +"/geo/add_cube.php")

        //le traitement de la connection au site
        val httpURLConnection = url.openConnection() as HttpURLConnection
        httpURLConnection.requestMethod = "POST"
        httpURLConnection.setRequestProperty(
            "Content-Type",
            "application/x-www-form-urlencoded"
        ) // The format of the content we're sending to the server
        httpURLConnection.setRequestProperty(
            "Accept",
            "application/json"
        ) // The format of response we want to get from the server
        httpURLConnection.doInput = true
        httpURLConnection.doOutput = true
        val dataOutputStream = DataOutputStream(httpURLConnection.getOutputStream)
```

Pour triangle :

- **Dans la classe Triangle :**

```

private var coteA: Double
private var coteB: Double
var hauteur : Double = 0.0
    private set

constructor(
    nom: String,
    perimetre: Double,
    surface: Double,
    coteA: Double,
    coteB: Double,
    coteC: Double
) : super(nom, perimetre, surface)

constructor(coteA: Double, coteB: Double, coteC: Double) : super(nom = "triangle", perimetre = 0.0, surface = 0.0) {
    this.coteA = coteA
    this.coteB = coteB
    this.coteC = coteC
}

fun calculHauteur(): Double {
    val a2 = coteA * coteA
    val b2 = coteB * coteB
    val c2 = coteC * coteC
    return Math.sqrt(c2 - (a2 + b2 + c2) / (2 * coteA)) * ((a2 - b2 + c2) / (2 * coteA))
}

override fun calculPerimetre(): Double {
    return coteA + coteB + coteC
}

override fun calculSurface(): Double {
    val base = coteA
    hauteur = calculHauteur()
    val s = base * hauteur / 2
    return coteA * hauteur / 2
}

```

- Faire la même chose que les autres formes pour ajouter, supprimer un, supprimer plusieurs, ajouter plusieurs et récupérer les données toujours dans la classe triangle :

```

//Fonction pour insérer des données
suspend fun addTriangle(coteA: String, coteB: String, coteC: String): JSONObject {
    //La variable JSON qui contient la réponse de l'ajout reçue du serveur WEB
    var jObj = JSONObject()

    //les paramètres à envoyer en post au serveur web pour inserer dans la BDD
    val uriBuilder = Uri.Builder() Uri.Builder!
        .appendQueryParameter("coteA", coteA) Uri.Builder!
        .appendQueryParameter("coteB", coteB)
        .appendQueryParameter("coteC", coteC)
        .build()

    //Formatage des paramètres
    val params = uriBuilder.toString().replace(
        oldValue: "?",
        newValue: ""
    ) // Remove the "?" from the beginning of the parameters ?name=Jack&salary=8054&age=45
    val postData = params.toByteArray(StandardCharsets.UTF_8)

    //on lance je job indépendant qui insere les données
    val insertionJob = GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope
        Log.d(tag: "dans le scope", msg: "globalScope")
        //url contient l'adresse du site à exécuter
        val url = URL(spec: "http://" + ProjectConstance.URL_SERVEUR + "/geo/add_triangle.php")

        //le traitement de la connection au site
        val httpURLConnection = url.openConnection() as HttpURLConnection
        httpURLConnection.requestMethod = "POST"
        httpURLConnection.setRequestProperty(
            "Content-Type",
            "application/x-www-form-urlencoded"
        )
    }
}

```

- **Dans AfficheTriangle :**

- Pour afficher le tableau :

```
333 item { this: LazyItemScope
334
335     Row(
336         Modifier.fillMaxWidth(),
337         horizontalArrangement = Arrangement.Center,
338         verticalAlignment = Alignment.CenterVertically
339     ) {
340         this.RowScope
341         TableCell(
342             text = "Id",
343             weight = column1Weight,
344             alignment = TextAlign.Left,
345             title = true
346         )
347         TableCell(
348             text = "coteA",
349             weight = column2Weight,
350             alignment = TextAlign.Right,
351             title = true
352         )
353         TableCell(
354             text = "coteB",
355             weight = column2Weight,
356             alignment = TextAlign.Right,
357             title = true
358         )
359         TableCell(
360             text = "coteC",
361             weight = column2Weight,
362             alignment = TextAlign.Right,
363             title = true
364         )
365         TableCell(
366             text = "Périmètre",
367             weight = column3Weight,
368             alignment = TextAlign.Center,
369             title = true
370         )
371         TableCell(
372             text = "Surface",
373             weight = column4Weight,
374             alignment = TextAlign.Center,
```

- Faire même méthode pour les autres formes.