Mobile operating systems

IOS Application

RECIPE MASTER

Created by:

BAKKALI TAHIRI MOHAMED

Technology used: CODENAME ONE

Codename One is an open-source cross-platform framework aiming to provide write once, run anywhere code for various mobile and desktop operating systems (like Android, iOS, Windows, MacOS, and others). It was created by the co-founders of the Lightweight User Interface Toolkit (LWUIT) project, Chen Fishbein and Shai Almog, and was first announced on January 13, 2012. It was described at the time by the authors as "a cross-device platform that allows you to write your code once in Java and have it work on all devices specifically: iPhone/iPad, Android, Blackberry, Windows Phone 7 and 8, J2ME devices, Windows Desktop, Mac OS, and Web. The biggest goals for the project are ease of use/RAD (rapid application development), deep integration with the native platform and speed.

For more information: https://www.codenameone.com

Programming language: JAVA

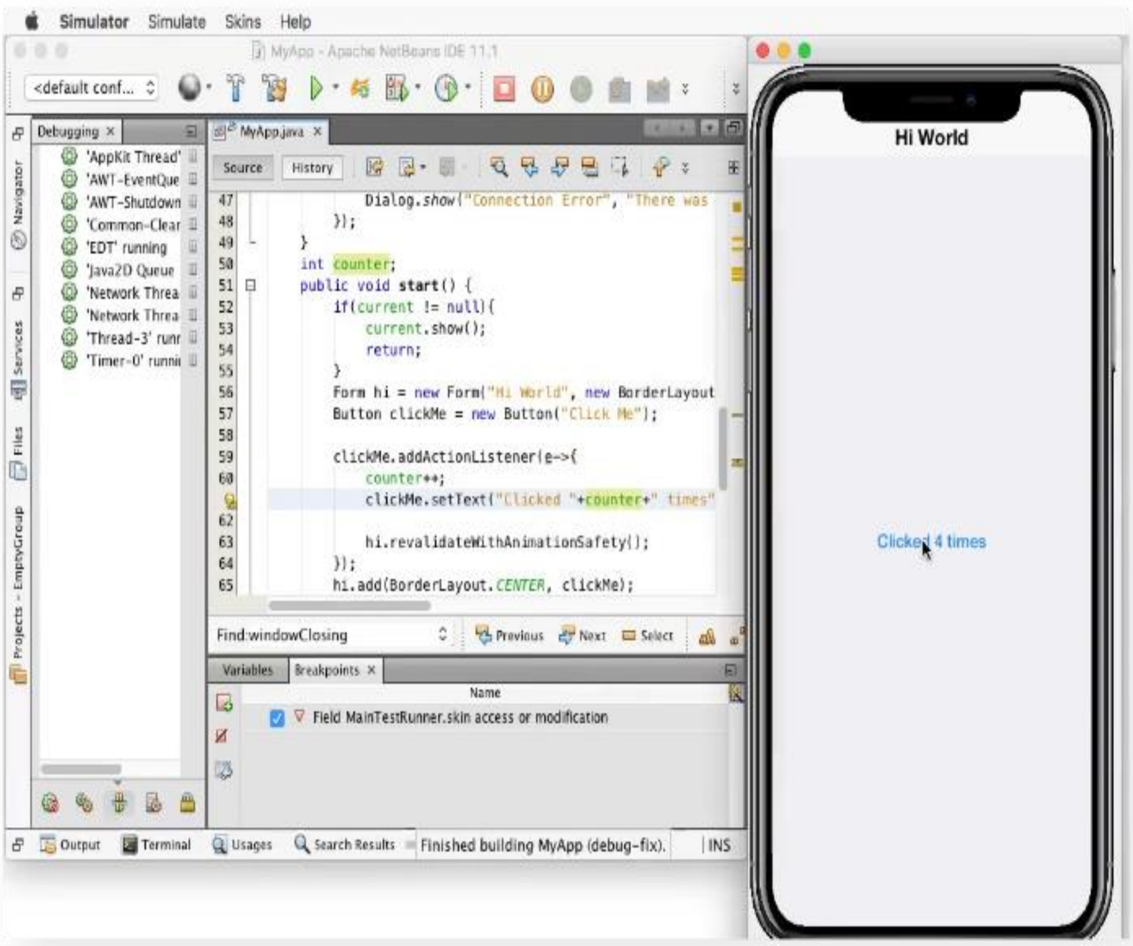
Java offers significant advantages in software development due to its platform independence, robust ecosystem, strong typing, memory management, security features, concurrency support, scalability, and vibrant community. Specifically in mobile development, Java's dominance in the Android ecosystem, along with specialized libraries, frameworks, and mature tooling, makes it a preferred choice for building high-quality mobile applications. With its "write once, run anywhere" capability and extensive support, Java continues to be a versatile and reliable option for developers across different domains, including mobile development.

Device Simulator:

The Codename One Device Simulator allows us to run/debug applications for multiple devices. It allows sending device events and device manipulations such as rotation, physical input etc. It supports multiple device types using an open pluggable architecture and is entirely open source.

Due to its unique architecture the device simulator allows developers to use all their existing development & debugging tools to check the quality and issues that arise while developing mobile applications.

The device simulator is very fast allowing developers to instantly check their applications without waiting for a slow emulator to "boot up", while it is not a replacement for checking applications on a physical device it still provides very good parity with device functionality



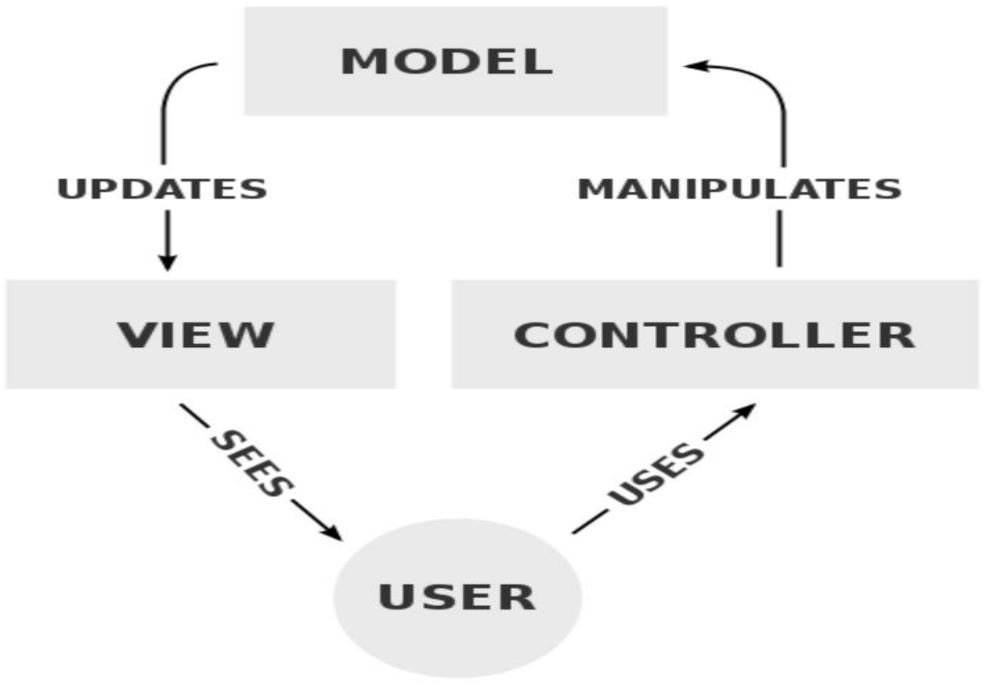
```
                Dialog.show("Connection Error", "There was
            });
        }
        int counter;
        public void start() {
            if(current != null){
                current.show();
                return;
            }
            Form hi = new Form("Hi World", new BorderLayout
            Button clickMe = new Button("Click Me");

            clickMe.addActionListener(e->{
                counter++;
                clickMe.setText("Clicked "+counter+" times"

                hi.revalidateWithAnimationSafety();
            });
            hi.add(BorderLayout.CENTER, clickMe);
```

Field MainTestRunner.skin access or modification

Finished building MyApp (debug-fix).

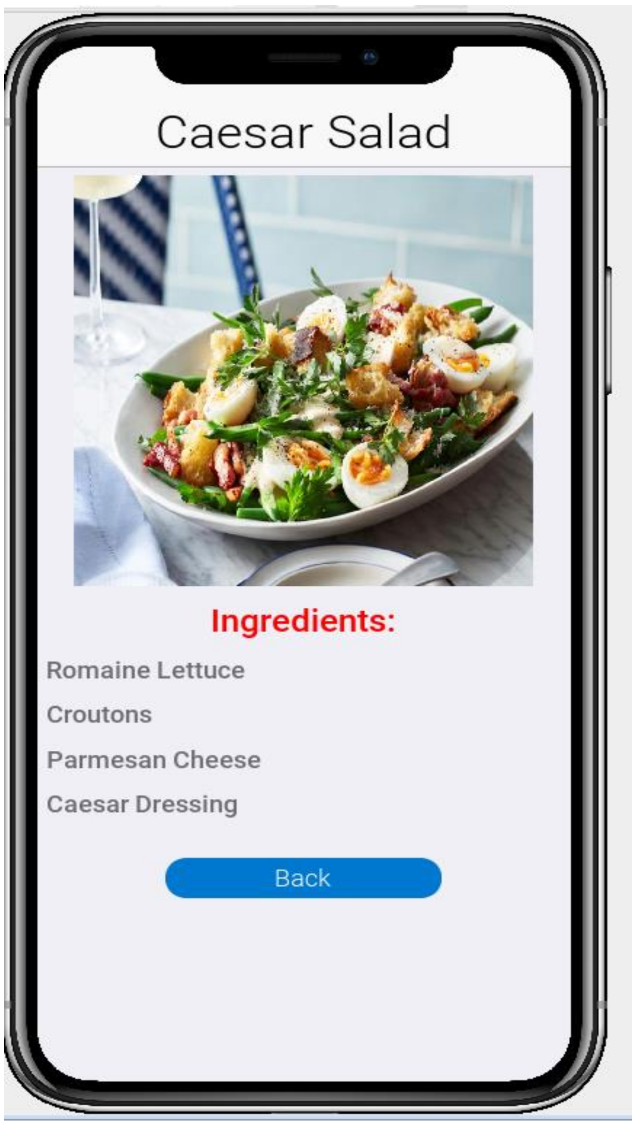
Application architecture: MVC

In mobile development, the Model-View-Controller (MVC) architecture is a design pattern that separates an application into three interconnected components: the Model, which represents the data and business logic, the View, which displays the user interface, and the Controller, which handles user input and updates the Model and View accordingly. This architecture promotes code organization, reusability, and maintainability. In the context of mobile development, MVC helps developers to create modular and easily maintainable applications by separating concerns and facilitating parallel development of different components. The Model encapsulates data retrieval, manipulation, and storage, the View handles the presentation layer, and the Controller manages user interactions and application flow. This separation of concerns enables better code organization, easier debugging, and scalability in mobile applications.



My Application : Recipe Master

The "Recipe Master" application functions as a catalog, presenting users with a diverse array of dishes along with their corresponding ingredients. It excels in providing users with an aesthetically pleasing interface that showcases each recipe alongside its ingredients, offering a delightful browsing experience. Additionally, "Recipe Master" assists users in effortlessly accessing detailed information about each recipe and its ingredients, enhancing their culinary journey.



MODEL:

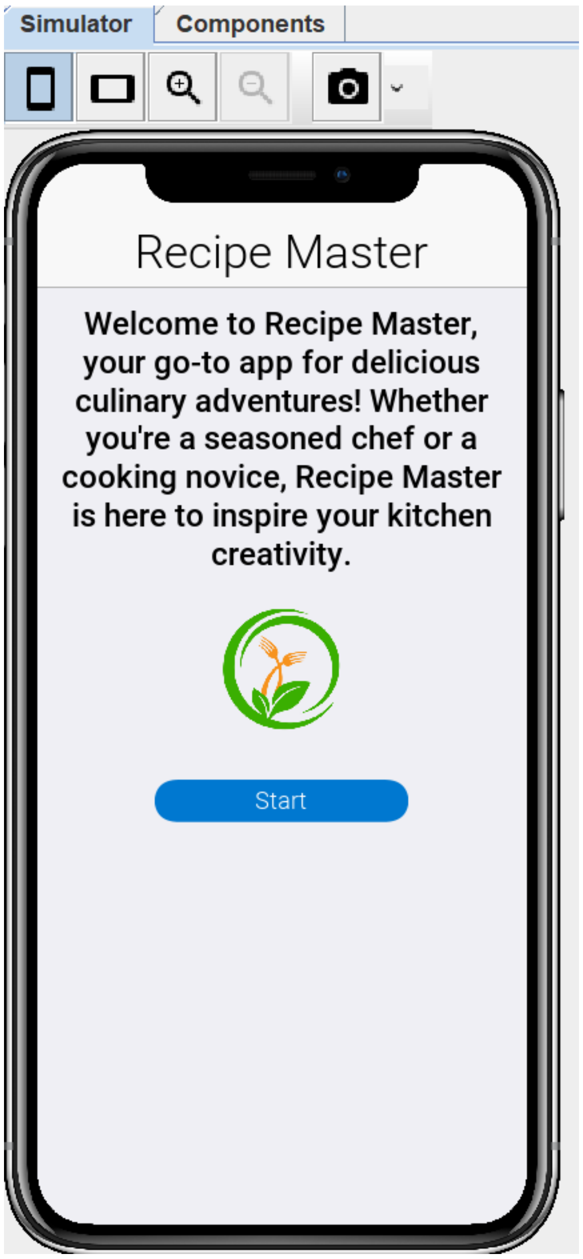
```java
public class RecipeModel {
    private String name;
    private String ingredients;
    private String imagePath;
    private static Database db;
    private static boolean dbInitialized = false;
    public RecipeModel(String name, String ingredients, String imagePath) {
        this.name = name;
        this.ingredients = ingredients;
        this.imagePath = imagePath;
    }
    @Override
    public String toString() {
        return name;
    }
    public String getName() { return name; }
    public String getIngredients() { return ingredients; }
    public String getImagePath() { return imagePath; }
    // database creation
    public static void initDB() {
        if (dbInitialized) return;
        try {
            db = Database.openOrCreate("RecipeDB.db");
            db.execute("CREATE TABLE IF NOT EXISTS recipes (id INTEGER PRIMARY KEY, name
TEXT, ingredients TEXT, imagePath TEXT);");
            dbInitialized = true;
        } catch (IOException e) {
            Log.e(e);
        }
    }

    // inseting data into the database
    public static void insertExampleRecipes() {
        if (!dbInitialized) initDB();
        try {
            db.execute("DELETE FROM recipes;");
            String[][] examples = {
                    {"Spaghetti Carbonara", "Pasta, Eggs, meet", "spagetti.png"},
                    {"Margherita Pizza", "Dough, Tomatoes, Mozzarella, Basil",
"pizza.png"},
                    {"Chicken Alfredo", "Fettuccine, Chicken, Cream, Parmesan Cheese",
"chicken_alfredo.png"},
                    {"Caesar Salad", "Romaine Lettuce, Croutons, Parmesan Cheese, Caesar
Dressing", "caesar_salad.png"},
                    {"Beef Burger", "Beef Patty, Lettuce, Tomato, Onion, Pickles, Cheese,
Burger Bun", "beef_burger.png"},
                    {"Chocolate Cake", "Flour, Sugar, Cocoa Powder, Eggs, Butter, Milk,
Vanilla Extract", "chocolate_cake.png"},
                    {"Fish and Chips", "Fish Fillets, Potatoes, Flour, Baking Powder",
"fish_and_chips.png"}
            };
            for (String[] recipe : examples) {
                db.execute("INSERT INTO recipes (name, ingredients, imagePath) VALUES
(?, ?, ?);", new Object[]{recipe[0], recipe[1], recipe[2]});
            }
        } catch (IOException e) {
            Log.e(e);
        }
    }

    // Read data from the database
    public static List<RecipeModel> readRecipes() {
        List<RecipeModel> recipes = new ArrayList<>();
        if (!dbInitialized) initDB();
        try {
            Cursor cur = db.executeQuery("SELECT * FROM recipes;");
            while (cur.next()) {
                Row row = cur.getRow();
                String name = row.getString(1);
                String ingredients = row.getString(2);
                String imagePath = row.getString(3);
                recipes.add(new RecipeModel(name, ingredients, imagePath));
            }
        } catch (IOException e) {
            Log.e(e);
        }
        return recipes;
    }

}
```

Controller :

```
public class RecipeController {
    private Form currentForm;
    public RecipeController() {
        RecipeModel.initDB();
        RecipeModel.insertExampleRecipes();
    }
    public void showWelcomeScreen() {
        Form welcomeForm = new WelcomeView(this).createForm();
        switchForm(welcomeForm);
    }
    public void showRecipesList() {
        java.util.List<RecipeModel> recipes = RecipeModel.readRecipes();
        Form recipesListForm = new RecipesListView(this, recipes).createForm();
        switchForm(recipesListForm);
    }
    public void showRecipeDetails(RecipeModel recipe) {
        Form recipeDetailsForm = new RecipeDetailsView(this, recipe).createForm();
        switchForm(recipeDetailsForm);
    }
    private void switchForm(Form newForm) {
        if (currentForm != null) {
            currentForm.showBack();
        }
        currentForm = newForm;
        currentForm.show();
    }


}
```

First View:

```
public class WelcomeView {
    private RecipeController controller;

    public WelcomeView(RecipeController controller) {
        this.controller = controller;
    }

    public Form createForm() {
        Form hi = new Form("Recipe Master", BoxLayout.y());
        SpanLabel welcomeLabel = new SpanLabel("Welcome to Recipe Master, " +
                "your go-to app for delicious culinary adventures! Whether you're a
seasoned chef or a cooking novice," +
                " Recipe Master is here to inspire your kitchen creativity.");
        welcomeLabel.setTextUIID("MultiLineLabel");
       welcomeLabel.setUIID("WelcomeLabel");
       welcomeLabel.getAllStyles().setAlignment(Component.CENTER);
        welcomeLabel.setWidth(Display.getInstance().getDisplayWidth() - 100);
        welcomeLabel.setHeight(300);

        try {
            Resources res = Resources.openLayered("/NativeTheme");
            String imageName = "logopng.png";
            Image logo = res.getImage(imageName);
           logo.scaled(1500,1500);
            ImageViewer imageView = new ImageViewer(logo);
            hi.add(imageView);
        } catch (IOException e) {
            e.printStackTrace();
        }

        Button startButton = new Button("Start");
        startButton.addActionListener(e -> controller.showRecipesList());
        hi.add(startButton);
        return hi;
    }
}
```

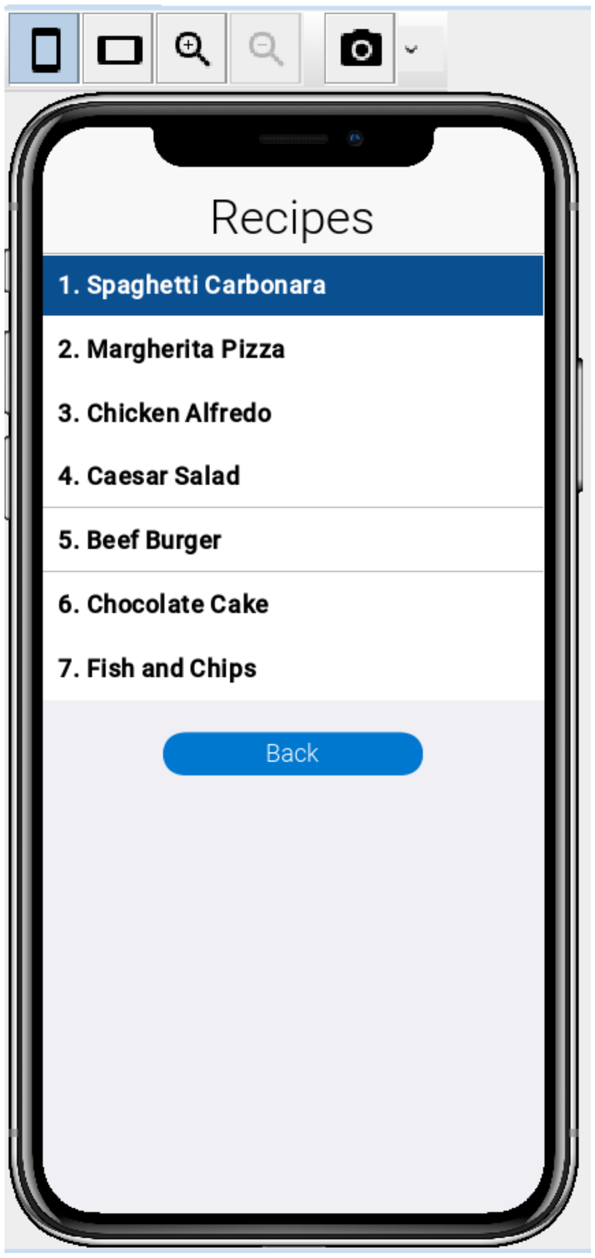


Recipe Master

Welcome to Recipe Master, your go-to app for delicious culinary adventures! Whether you're a seasoned chef or a cooking novice, Recipe Master is here to inspire your kitchen creativity.

Recipes List View:

```
public class RecipesListView {
    private RecipeController controller;
    private java.util.List<RecipeModel> recipes;

    public RecipesListView(RecipeController controller, java.util.List<RecipeModel>
recipes) {
        this.controller = controller;
        this.recipes = recipes;
    }

    public Form createForm() {
        Form recipesListForm = new Form("Recipes", BoxLayout.y());
        DefaultListModel<RecipeModel> model = new DefaultListModel<>();
        for (RecipeModel recipe : recipes) {
            model.addItem(recipe);
        }

        List<RecipeModel> recipesList = new List<>(model);
        recipesList.addActionListener(e -> {
            RecipeModel selectedRecipe = recipesList.getSelectedItem();
            recipesList.setUIID("subtitle");
            controller.showRecipeDetails(selectedRecipe);
        });

        recipesListForm.add(recipesList);
        Button backButton = new Button("Back");
        backButton.addActionListener(e -> controller.showWelcomeScreen());
        recipesListForm.add(backButton);
        return recipesListForm;
    }
}
```



Recipes

1. Spaghetti Carbonara

2. Margherita Pizza

3. Chicken Alfredo

4. Caesar Salad

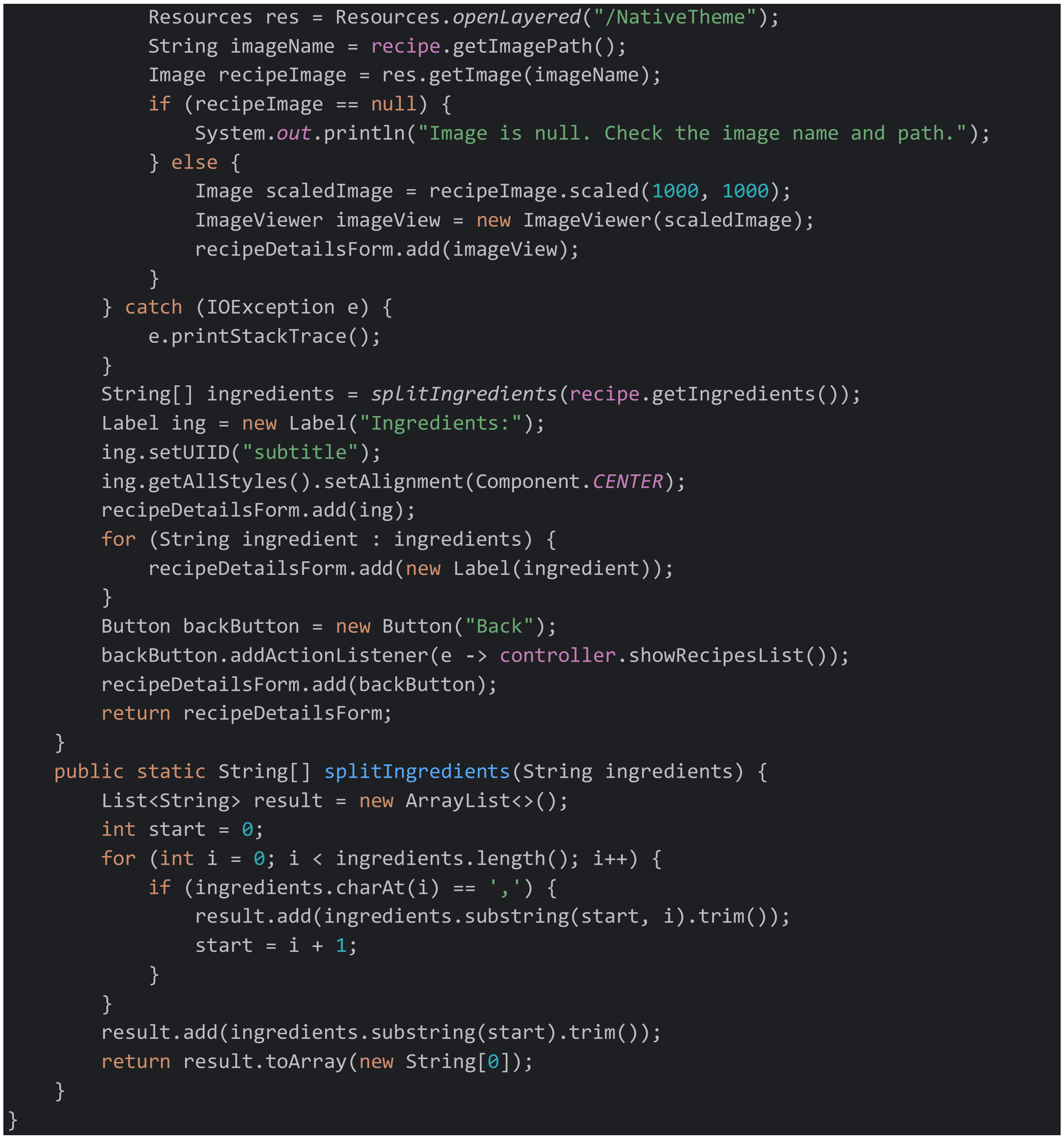
5. Beef Burger

6. Chocolate Cake

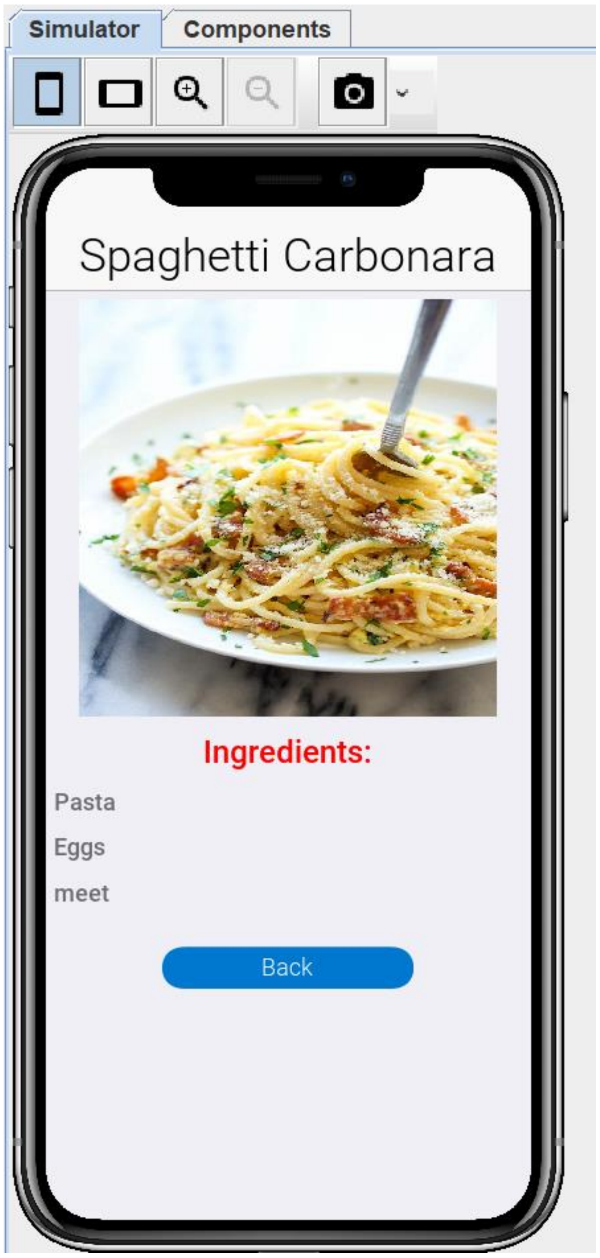
7. Fish and Chips

Recipe details View:

```java
public class RecipeDetailsView {
    private RecipeController controller;
    private RecipeModel recipe;
    public RecipeDetailsView(RecipeController controller, RecipeModel recipe) {
        this.controller = controller;
        this.recipe = recipe;
    }
    public Form createForm() {
        Form recipeDetailsForm = new Form(recipe.getName(), BoxLayout.y());
        try {
```



```java
            Resources res = Resources.openLayered("/NativeTheme");
            String imageName = recipe.getImagePath();
            Image recipeImage = res.getImage(imageName);
            if (recipeImage == null) {
                System.out.println("Image is null. Check the image name and path.");
            } else {
                Image scaledImage = recipeImage.scaled(1000, 1000);
                ImageViewer imageView = new ImageViewer(scaledImage);
                recipeDetailsForm.add(imageView);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        String[] ingredients = splitIngredients(recipe.getIngredients());
        Label ing = new Label("Ingredients:");
        ing.setUIID("subtitle");
        ing.getAllStyles().setAlignment(Component.CENTER);
        recipeDetailsForm.add(ing);
        for (String ingredient : ingredients) {
            recipeDetailsForm.add(new Label(ingredient));
        }
        Button backButton = new Button("Back");
        backButton.addActionListener(e -> controller.showRecipesList());
        recipeDetailsForm.add(backButton);
        return recipeDetailsForm;
    }
    public static String[] splitIngredients(String ingredients) {
        List<String> result = new ArrayList<>();
        int start = 0;
        for (int i = 0; i < ingredients.length(); i++) {
            if (ingredients.charAt(i) == ',') {
                result.add(ingredients.substring(start, i).trim());
                start = i + 1;
            }
        }
        result.add(ingredients.substring(start).trim());
        return result.toArray(new String[0]);
    }
}
```



Spaghetti Carbonara

Ingredients:

Pasta

Eggs

meet