Ubiquitous Cooking

Daniel Perez, Guadalupe Torres, Manuel Serna-Aguilera, Jean Roca, Alejandro Urquieta, Jack Scholes

Problem & Objective

- Cooking new and unfamiliar recipes can be a challenge to create quickly and accurately
- We wished to lower the barrier of entry of cooking brand new recipes for everyone
- We accomplished this by creating a Smartwatch app that makes it easy to transition between cooking and completing the necessary steps in the cooking process.

Key Concepts

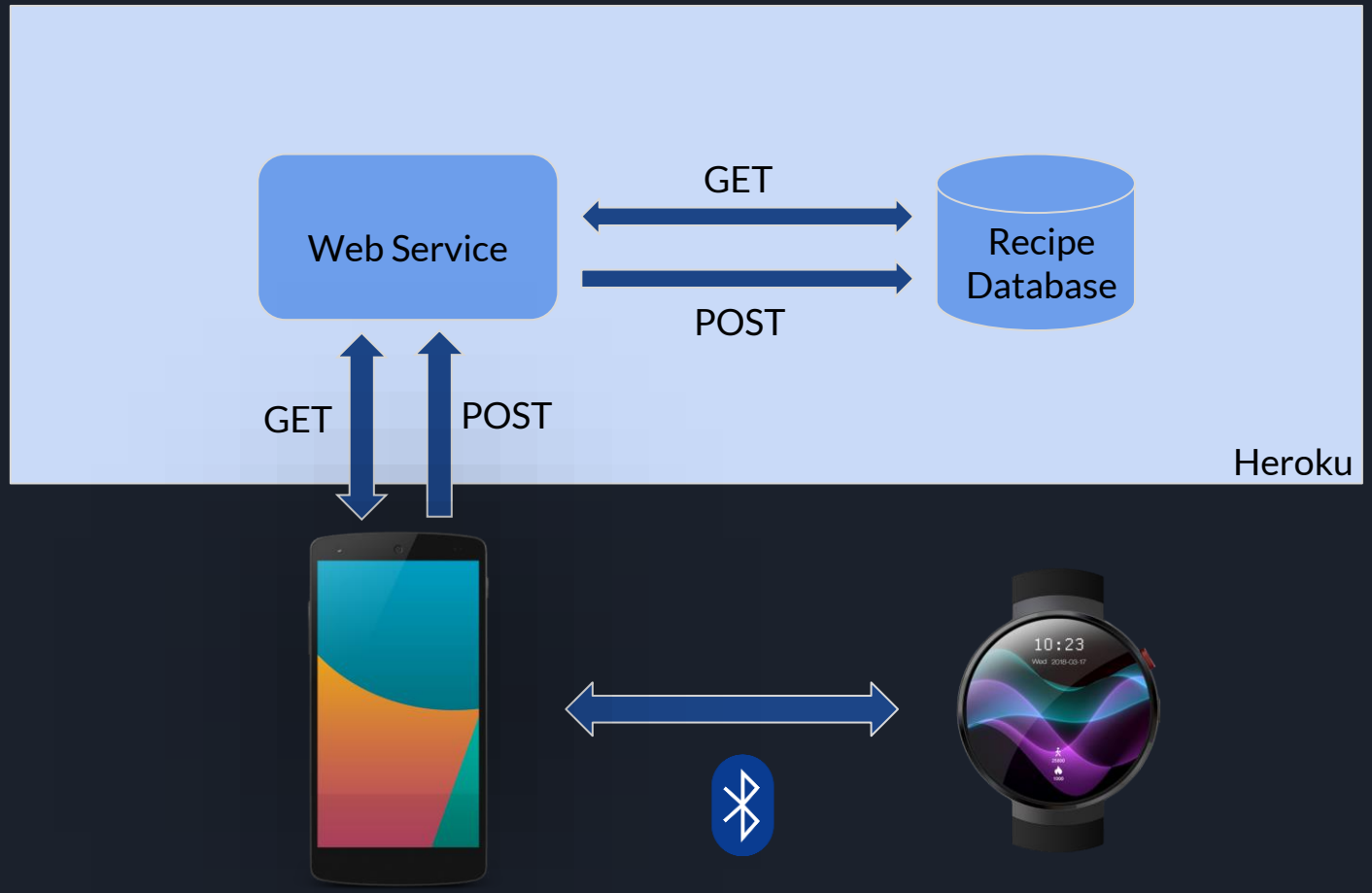
- Spring
  - Java framework for creating REST services. Will host a database for sharing recipes
  - Hosted on Heroku, an online service that can host a server that we can upload.
- JPA
  - Java Persistence API
  - Standard API for accessing databases within Java applications
- Dependency Injection
  - Concept used to pull objects from across an application on an as-needed basis. For example, we would “inject” a singleton Http client object wherever we need it in a class.
  - Greatly improves code quality.

Key Concepts

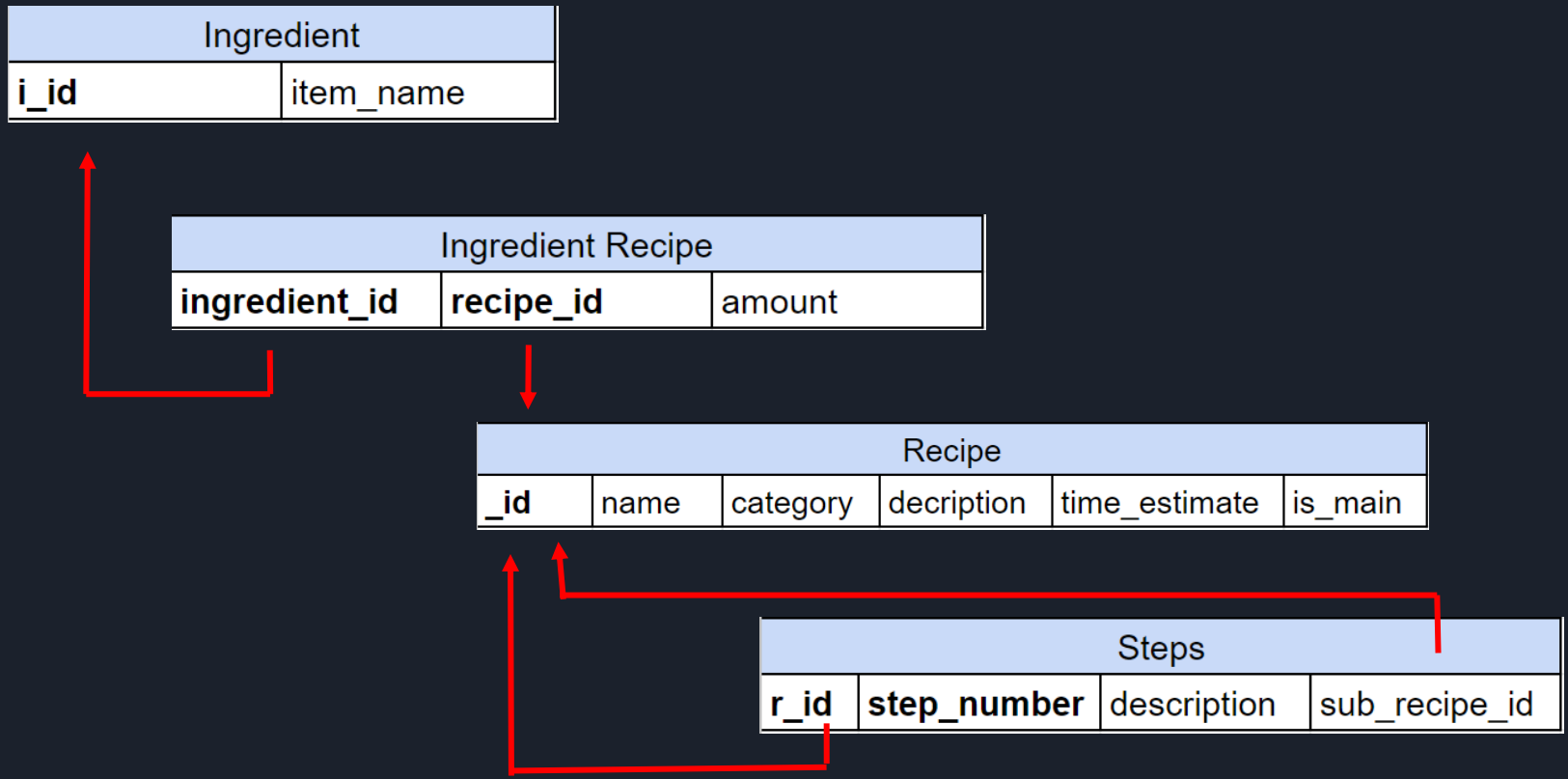
- Android:
  - Google’s Mobile Operating System. Its SDK framework contains a set of development tools which include a debugger, and libraries. Both the Wear OS and Android Application will be developed using Android SDK.
- Wear OS:
  - Google's Android operating system marketed for smartwatches and other wearables. The smart watch client will be developed as a Wear OS app.
- SQL Implementation:
  - Systematic collection of data which supports storage and manipulation of data. We will implement three separate SQL databases for the server, Android client, and Wear OS client.

Spring

- Spring server was set up in order for the phone to connect to the SQL database.
- Java framework for creating REST services.
- Broken down into 4 components:
  - Controller
    - In charge of handling all HTTP calls when they hit the server.
  - Model
    - Object models that are shared with the database.
  - Repository
    - Handles CRUD operations for the database.
  - Service
    - Connects with the repository to call its methods.
- Each component also includes their appropriate unit testing file to test for proper functionality.
- Tests were written using the JUnit testing framework.
- Hosted on Heroku, an online service that can host a server that we can upload.

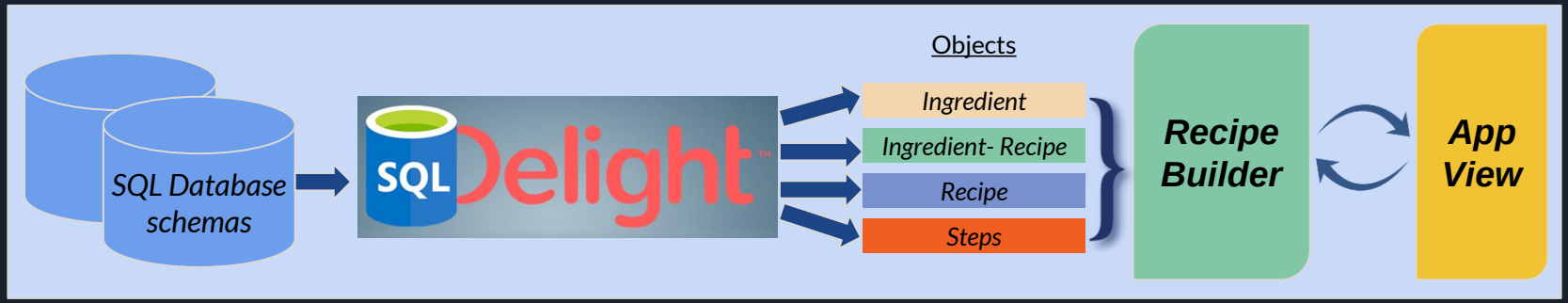

Web Service
GET
POST
Recipe Database
GET
POST
Heroku
10:23

SQL Implementation



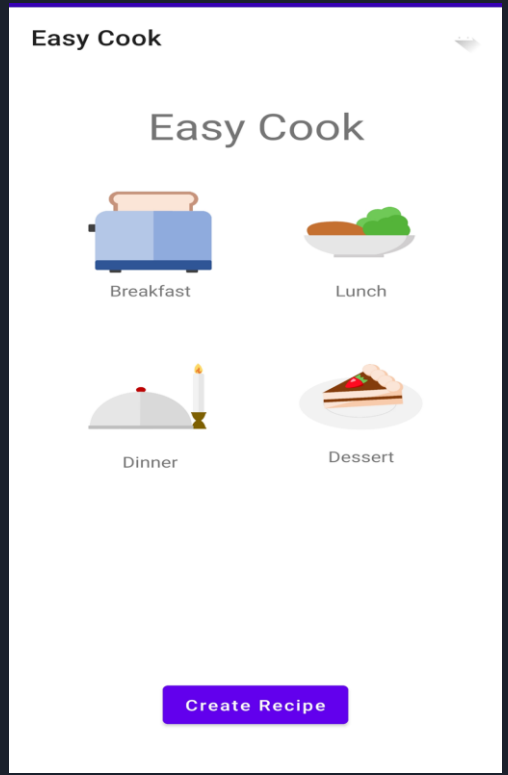
SQLDelight

- SQLDelight was an Android library utilized to implement the database module.
- It generated typesafe APIs from SQL statements which verified schema, statements, and migrations and provides IDE features.
- By providing SQL schemas, SQLDelight generated a Database Kotlin class with an associated Schema object that was used to create the RecipeBuilder class.



Recipe Category View

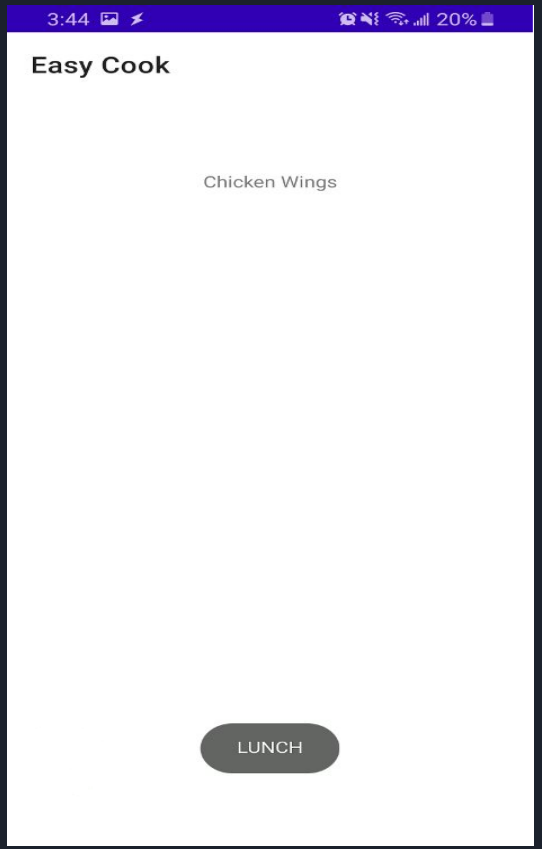
- The view the user first sees upon entering the application
- There are four food categories each recipe will fall under
  - Breakfast
  - Lunch
  - Dinner
  - Dessert
- Tap on the corresponding icon to cook a certain recipe from a certain category



Recipe List View

- Under each category will be a list of recipes the user can choose from

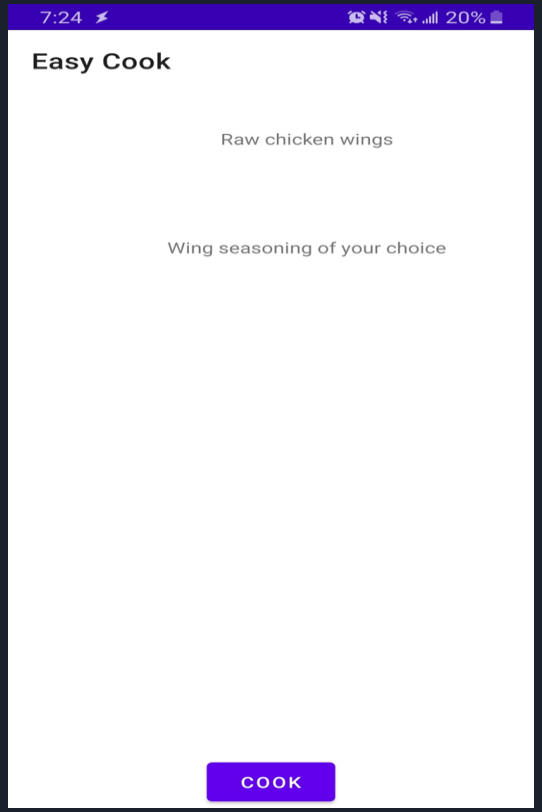
- Tap on the recipe name to begin the cooking process



Ingredients View

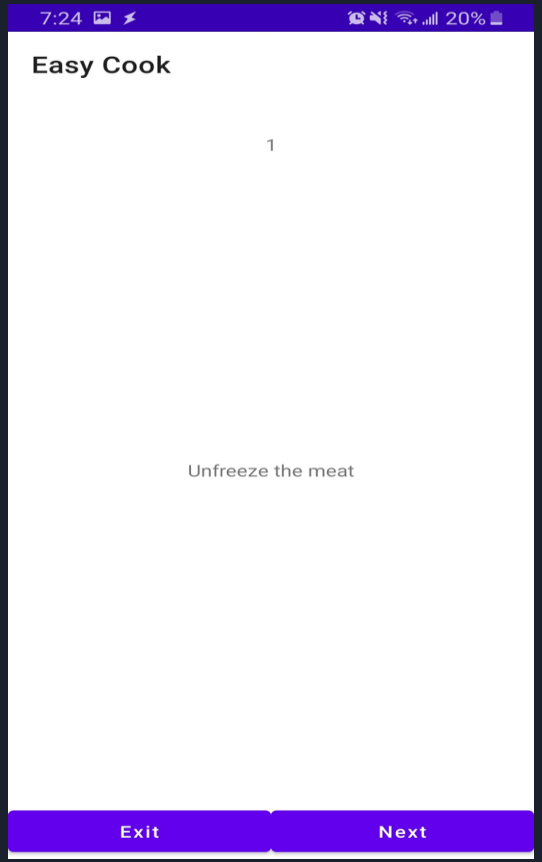
- Before the user starts cooking, show all the ingredients in a scroll view

- The user can then tap “COOK” to begin cooking.



Steps View

- Here, the user goes through the steps for the recipe, one at a time
- Tap “Next” to move to the next step
- Tap “Exit” to quit the process



Syncing Data between the Watch and Phone

- Needed something that would allow the Watch to query the phone, even if the phone screen is off.
    - Android background Services were the perfect solution to this and they even provide API's for handling this exact scenario with a watch.
- Very similar to a client-server, or request-response relationship.
- The phone provides "endpoints" just like a server would for the features the watch needs, the watch just needs to query against the phone like most devices would a server
    - '/recipes' to serve up the list of recipes
    - '/steps' to serve the list of steps associated with the recipes
- The phone serves up its' data from its SQLite database, whose work is largely already handled for this task and needed little modification
- On response, the Watch just needs to display the information it received.

DEMO


Pou offee
Pour Over Coffee