Cooking Companion
Software Architecture Documentation

Group 2

Cooking Code-panions

March, 5, 2023

The architecture that the Cooking Companion app will be implementing is the Flux architecture model. Flux is essentially a data flow pattern that aims to solve state management complexity by using Views, Stores, Actions, and a Dispatcher. Flux handles this flow of data in a one way direction so that there isn't much of a change for increased stores or views.

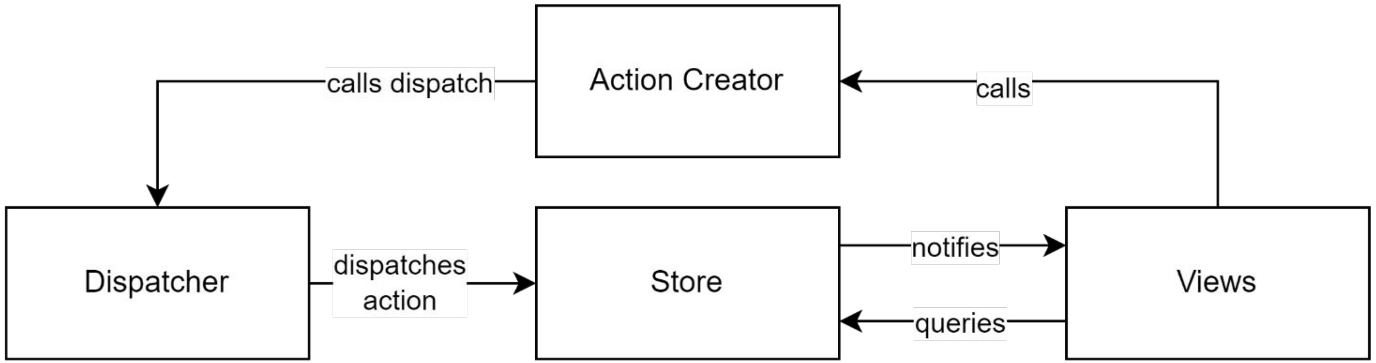
The main reason for our group choosing Flux is that we will be using React in developing our application and these two systems pair nicely together. Implementing Flux will help with the scalability of the application as our user base grows, as well as, deals with any state management issues that might arise.

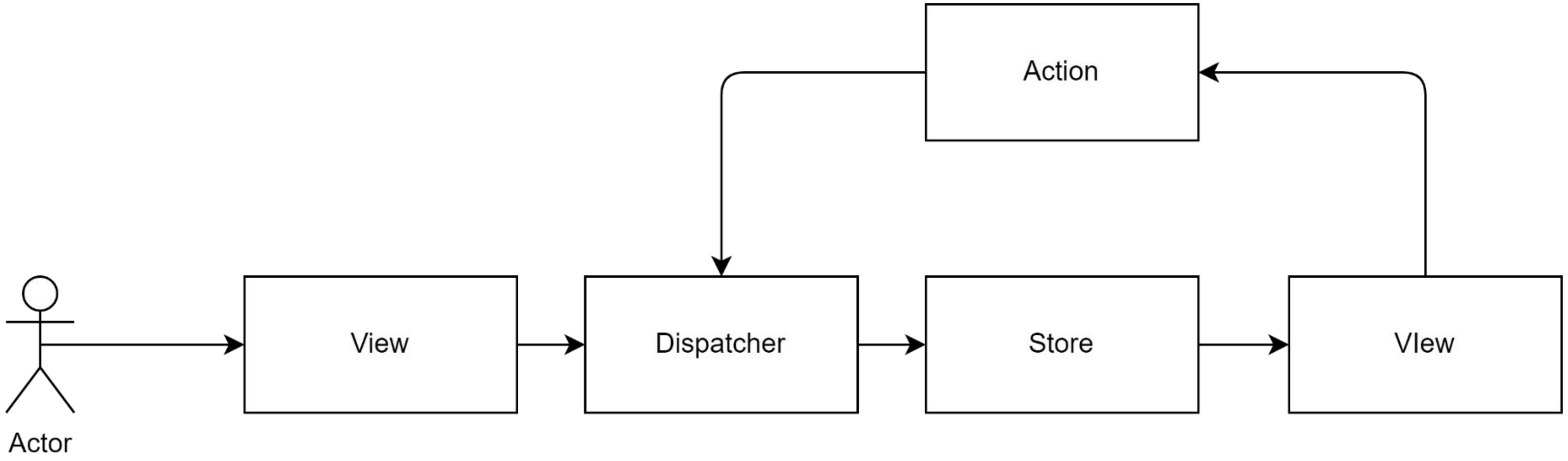
In Flux all of the UI will be handled in the Views section of the data flow. The user actions in the *views* will call an *action creator*, which will cause an *action* event that will be sent to the *dispatcher.* The dispatcher is the single point of emission for ALL actions in a flux application. Stores update themselves in response to an action sent from the dispatcher. When there is an influx of actions sent from the dispatcher, they are sent to all the stores in the application. The actions from the dispatcher have no capability of updating the stores; the stores update themselves using internal logic.

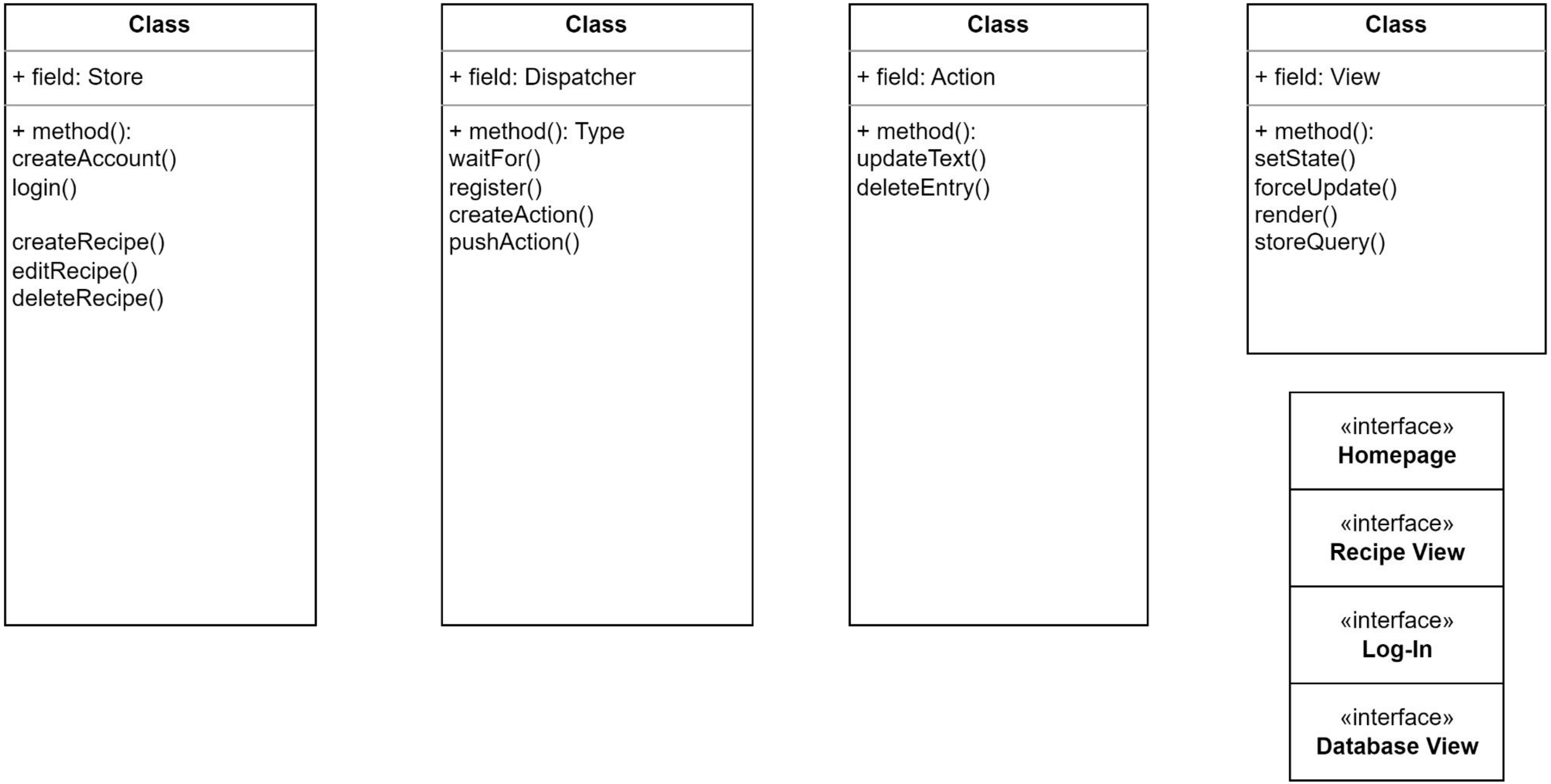
This will allow for a level of low coupling and high cohesion between the functions of our application.

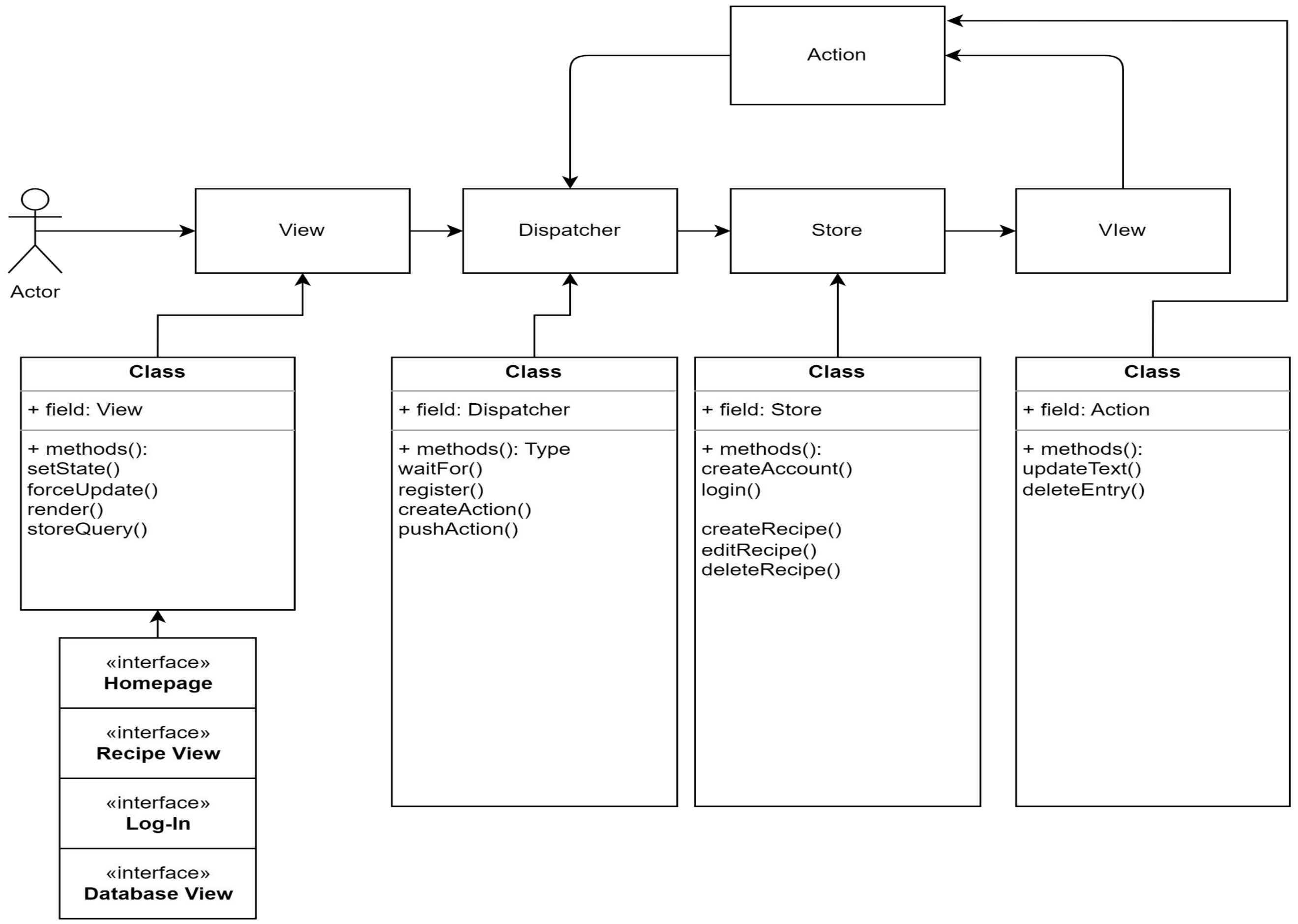
Base
FLUX Models

simple flux flow






Class
+ field: Store
+ method():
createAccount()
login()
createRecipe()
editRecipe()
deleteRecipe()
Class
+ field: Dispatcher
+ method(): Type
waitFor()
register()
createAction()
pushAction()
Class
+ field: Action
+ method():
updateText()
deleteEntry()
Class
+ field: View
+ method():
setState()
forceUpdate()
render()
storeQuery()
«interface»
Homepage
«interface»
Recipe View
«interface»
Log-In
«interface»
Database View


Action
Actor
View
Dispatcher
Store
VIew
Class
+ field: View
+ methods():
setState()
forceUpdate()
render()
storeQuery()
Class
+ field: Dispatcher
+ methods(): Type
waitFor()
register()
createAction()
pushAction()
Class
+ field: Store
+ methods():
createAccount()
login()
createRecipe()
editRecipe()
deleteRecipe()
Class
+ field: Action
+ methods():
updateText()
deleteEntry()
«interface»
Homepage
«interface»
Recipe View
«interface»
Log-In
«interface»
Database View