Dokumentation Software-Architektur

Die Architektur der Android Tetris App orientiert sich streng am MVC (Model View Controller) Pattern, es wurde ebenso viel wert auf Flexibilität und Abstaktion gelegt welshalb jede Komponente ein Interface hat und von anderen Komponenten auch nur die Interfaces kennt. Ebenso wurde eine Schichtarchitektur umgesetzt dh. Jedes Package ist ebenso ein Layer welcher nur mit sich selbst und dem Layern darunter kommunizieren darf wobei dies Leider nicht immer möglich ist zu realisieren wesegen zusätzlich das Observer Pattern implementiert wurde um die Abhängigkeit zu höheren Layern umzukehren, die trifft auf das Tetris spiel und die Android Gui zu.

Jedoch wurde für den Server welcher in PHP geschrieben wurde ein einfach MVC Pattern umgesetzt.

Server:

* Model
  + Für jede Datenbank Tabelle wir eine Bean benötigt welche die Informationen und Namen der Datenbank Tabelle enthält
  + Desweiten wird ein Datenbank Mapper benötigt hier wird die Datenbank erstellt und an das Datenbank Objekt weitergeleitet
  + zusätzlich wird ein Datenbank Table Objekt benötigt welches den tatsächlichen Namen der Tablelle enthält
* View
  + ist eine phtml Datei welche den selben Namen trägt wie die zuvor ausgeführte Aktion des Controllers sie muss in eine Subfolder der View Folder gelegt werden welche den Namen des Controllers haben muss. Sie enthält nur ein html fragment keine komplettes Dokument.
* Controller
  + Werden vom Dispatcher aufgerufen bekommen ein Request Objekt übergeben
* Router
  + Packt den Http Request in ein Request Objekt
* Dispatcher
  + Bekommt das vom zuvor vom Router generiertem Request objekt und ruft dann die vom User genwünschte Aktion auf dem Richtigen Controller per Reflection auf.

Client:

Da das Tetris Spiel vorher für ein anderes Projekt geschrieben wurde und für dieses Projekt nur eine neue Gui geschrieben wurde wird in diesem Abschnitt nur kurz auf die Architektur des Tetris Spiels eingegangen.

* View/Gui
  + Eula Activity zeigt dem nutzer die Nutzungsbedingungen an falls er sie noch nicht akzeptiert hat. Falls der Nutzer diese ablehnt wird die App wieder geschlossen. Akzeptiert er diese wird er zur nächsten Aktivity weitergeleitet ebenfalls wird in den sharedprefences gespeicheret das sie Akzeptiert wurde so das der nutzer diese nicht immer wieder akzeptieren muss.
  + Splash Activity erstellt eine initiale Instanz des hisghscoreController Singletons.
  + Menu Activity zeichnet das Menu und leitet zu den vom Nutzer gewünschten Activities weiter
  + Highscore Activity Zeichnet die Highscore Liste, sie berechnet für wieviele highscores platz auf dem Display ist und stößt dann das Laden der Highscores vom Server an.
  + Game Activity zeichnet den GameView, die Buttons und den Aktuellen Score des Nutzers. Sie Leitet die Nutzereingaben von den Buttons oder der Touchevents an die Controller weiter. Ausserdem Startet/Pausiert/Beended sie das Tetris Spiel, Speichert es Temporär falls der Nutzer eine Notification vom System bekommt oder falls er auf den home Screen geht. Nach dem Ende eines Spiels frägt sie beim Nutzer ab ob er sein Score Teilen möchte und stößt dies falls gewünsch dann an.
  + Game View Zeichnet das eigentliche Spielfeld und den Stein der als ncächsten dran kommt. Dies wurde alles mit einem Canvas Element realisiert.
  + MyApp Activity Kommunziert mit dem HighscoreController um highscores vom Server zu laden.
* Controller (nur HighscoreController ist neu hinzugekommen)
  + HighscoreController ist ein Singleton reicht die Daten an das/vom Model weiter.
  + GameController kontroliert das Tetris Spiel leitet die Nutzereingaben an die Hilfscontroller (Move Package) weiter. Stößt das generieren von neuen zufälligen Spielsteinen an.
  + GamemechanikController kontroliert die Gameloop
  + TetrisController kontroliert grundsätzlichere angelegenheiten des Spiels wie etwas den Highscore, kann das Spiel Starten/Resetten/Beenden.
* Move/HilfsController
  + Setzten die gewünschten Bewegungen des Spielsteins auf der Logischen Matrix des Spiels um.
* Observer
  + Bietet Klassen und Interfaces für Observer und Subject an um die Schichtarchitektur einhalten zu können dreht die Abhängigkeit der Layers um.
* Model
  + Enthält die Spielsteine(Element), die Logik Matrix(GameArray) und Kommunikation zum Server (Highscore).