Dokumentation

GeocacheHunt-App

Aufgabenverteilung

Frontend: Felix Unthan

Backend: Alexander Perschke

Frontend Backend Kommunikation: Alexander Perschke(übernommen)

Frontend

Unterstützte Betriebssysteme

Die Applikation läuft nun auf Smartphones mit einem Betriebssystem ab API Level 21. Die Activitys wurden dementsprechend darauf erweitert. Getestet wurde weitestgehend auf API Level 23.

Applikation

Die Applikation besteht aus vier Activitys. Die erste Activity ist die StartActivity, die der Nutzer sieht, wenn er die App öffnet. Hier öffnet sich ein Fenster, in welchem er sich zuerst einen Namen gibt. Falls er dies überspringt, wird er beim nächsten Öffnen der App wiederholt zur Eingabe eines Namens gebeten. Der Nutzer bekommt eine ID über den Server zugewiesen.

Die StartActivity zeigt oben eine Actionbar mit dem Namen der App „GeocacheHunt“. Links ist ein Gesicht zu sehen, auf dass der User klicken kann und somit zur zweiten Activity der CharacterActivity kommt. Neben dem User-Icon steht der Nutzername sowie seine Punktzahl. Es gibt noch die zwei Button „Map“ und „Shop“. Durch das Klicken auf Map kommt man zur dritten Activity (MapsActivity) und von dieser zur vierten und letzten, der CameraActivity. Im unterem Bereich wird eine Liste von Markern angezeigt, sofern vorhanden.

In der CharacterActivity kann der Nutzer sich sein User-Icon zusammenstellen. Dafür stehen ihm eine Auswahl an Hautfarben, Augenfarben, Nasen und Mündern zur Verfügung. Er kann zwischen den verschiedenen Mini-Icons per Scroll wählen. Mithilfe einer PlayerClass werden die gewählten Icons dem User zugeordnet und gespeichert. Nach dem Anklicken wird ihm sofort sein aktuelles Ergebnis angezeigt. Zur StartActivity kommt über den Back-Knopf des Smartphones.

Bevor Sie auf den Maps-Button klicken, BITTE die Berechtigung für den Standort freigeben in den Smartphone-Einstellungen. Durch Klicken auf den Maps-Button öffnet sich dann Google Maps. Hier kann der User seinen aktuellen Standort sehen und die schon gesetzten Marker. Wenn er in Reichweite von 25 Metern ist, sollte die App am unteren Rand einen ActionButton mit einer Kamera anzeigen. Durch Betätigen dieses Buttons gelangt der Nutzer zur CameraActivity. Der User kann selbst Marker setzen in dem er etwas länger auf einen Punkt auf der Map klickt. Anschließend sollte ein Alert aufgehen, wo der Name eingetragen werden kann.

Drückt der Nutzer nun auf den Kamera-Button wird er zuerst nach der Erlaubnis gefragt auf diese zugreifen zu dürfen. Nach dem die Berechtigung erteilt wurde, erscheint ein weiterer Button in der Mitte des Bildschirms. Durch Klicken auf den Button, welcher ein kleiner Geldhaufen ist, bekommt der Spieler 100 Coins auf sein Spielerkonto gutgeschrieben. Danach gelangt er wieder zur StartActivitiy.

Activitys

StartActivity:

- protected void OnCreate(Bundle savedInstanceState)

Die Funktion startet die StartActivity, die Startseite der App für den Nutzer..

- protected void OnResume()

Die Spielerdaten werden erneuert, sodass die aktuellen Punkte auf dem Bildschirm angezeigt werden.

CharacterActivity:

- protected void OnCreate(Bundle savedInstanceState)

Die Funktion startet die CharacterActivity und zeigt alle auswählbaren Icons.

- protected void OnResume()

Hier wird der Character geupdatet.

CharacterActivity:

- protected void OnCreate(Bundle savedInstanceState)

Die Funktion startet die CharacterActivity und zeigt alle auswählbaren Icons.

- protected void OnResume()

Hier wird der Character geupdatet.

CameraActivity:

- protected void OnCreate(Bundle savedInstanceState)

Erstellt eine View und zeigt den Cache an, auf den man drauf klicken kann.

- protected void createCameraPreview()

Hier wird eine Vorschau der Kamera gegeben, die eine neue Session startet.

- private void openCamera()

Die Einstellungen werden überprüft und die Berechtigungen erstellt, wodurch abschließend die Kamera geöffnet werden darf.

- private void closeCamera()

Kontrolle, ob die Kamera geschlossen ist, falls nicht, wird sie hier geschlossen.

- protected void onResume()

Updating des Surface.

- protected void onPause()

Kamera wird geschlossen.

Installation/Aufbau des Servers

Um das Projekt starten zu können, muss zunächst NetBeans gestartet werden.

Desweitern muss ein lokaler Hotspot eingerichtet werden, sodass sich Server und Client im gleichen Netzwerk befinden.

Starten des Servers durch das Ausführen von Server.java mittels Shift+F6

Die private String Variable **IP** in der **StartActivity**(Android Studio) muss möglicherweise geändert werden.

IP kann über cmd->ipconfig ermittelt werden.

Dokumentation Server

Auf dem Server liegen 3 Datenbanken.

Users: *Spalten*; ID, Username, Credits, Admin(darf neue Caches verteilen)

Die Tabelle User speichert alle User und ihre wichtigsten Werte.

Caches: *Spalten*; ID, Cachename, PositionX, PositionY

Die Tabelle Caches speichert den Namen und den Standort der Caches.

FoundCaches: *Spalten*; UserID, CacheID

FoundCaches verknüpft jeweils eine UserID und eine CacheID, so können alle vom User gefundenen Caches herausgesucht werden.

Die Kommunikation zwischen Server und Client wird mittels Socket und JSON-Objekten realisiert.

Funktionen des Servers (Alexander Perschke)

createUser(String username):

Der Server bietet die Möglichkeit einen User anzulegen und in die Datenbank zu schreiben.

Doppelte oder leere Nutzernamen werden abgefangen.

selectUser(int userID):

Der Server gibt ein JSON-Objekt es Users mit der übergeben ID zurück.

syncUserCredits(int userID, int credits):

Der Server schreibt den neuen Wert der Credits in die Datenbank.

createCache(String name, int posX, int posY):

Der Server speichert einen Cache mit Namen und den zugehörigen Dezimalkoordinaten in der Datenbank Caches.

collectCache(int userID, int cacheID):

Der Server speichert einen Eintrag mit der ID des User der den Cache gefunden hat und der ID des Caches in der Datenbank FoundCaches.

syncCaches(int posX, int posY):

Der Server gibt ein Array von JSON-Objekten(Caches) zurück, die sich im näheren Umkreis des Spielers befinden.

showFoundCaches(int userID):

Der Server gibt ein Array von JSON-Objekten(Caches) zurück, die der User bereits gesammelt hat.

deleteCache(int cacheID):

Der Server löscht den Cache mit der übergebenen ID aus der Datenbank Caches

Funktionen Frontend (Alexander Perschke)

Diese Aufgabe wurde kurz vor Ende der Abgabe von Alexander Perschke übernommen, da Felix Unthan noch nicht daran gearbeitet hatte. Mangels Zeit sind leider nur diese Funktionen zu Stande gekommen. Das Grundgerüst für eine createCaches Methode ist in der StartActivity zu finden. Diese konnte allerdings nicht getestet werden, da die MapsActivity noch nicht fertig programmiert ist.

createUser(String username)

Stellt ein JSON-Objekt zusammen und übergibt es an den Server, bei vergebenem Username oder unzulässigem Namen wird an den errorHandler verwiesen.

syncUserCredits(int userID, int credits)

Wird in der onResume() Methode aufgerufen. Trägt man eine feste ID ein so funktioniert die Methode. Die Funktion Playerclass.getPlayerID() scheint nicht den richtigen Wert zu liefern.

waitForResponse()

Zwingt den Code zu warten, bis der Sever eine Antwort liefert. Sehr unschön über eine globale String Variable gelöst.

getResponse()

Liest die Nachricht vom Server und schreibt sie in die globale Variable globalResponseString

errorHandler(String error)

Anhand des Strings wird entschieden welcher Fehler aufgetreten ist und wie darauf reagiert werden soll.

handleInvalidUsername()

Erzeugt einen AlertDialog und informiert den User über eine falsche Eingabe bei seinem Username