Relzione sul progetto

Nome del progetto di laboratorio:

Beer Masters Shop

SOMMARIO

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

1 DESVIAZIONI RISPETTO A QUANTO SPECIFICATO

Con lo sviluppo del progetto finito, l'applicazione soddisfa tutti i requisiti funzionali specificati nello schema del progetto.

Il tempo di sviluppo ha richiesto più tempo del previsto, la date per la prima release funzionale e con tutte le funzionalità implementate specificata nello schema era prevista per 25/05/2020 ma a causa di problemi nell'implementazione del backoffice di Nexi e problemi nella gestione dei prodotti, la data finale era il 04/06/2020.

Nello schema è specificato che l'applicazione solo richiede un'autorizzazione ma come l'app permette aggiungere/modificare foto ai prodotti, è stato necessario aggiungere l'autorizzazione android.permission.READ_EXTERNAL_STORAGE per ottenere foto del dispositivo.

È stato inoltre necessario utilizzare Cloud Storage di Firebase per archiviare le foto dei prodotti online.

L'app, alla fine, ha 6 activity invece 3 che era lo specificato nello schema.

Era anche previsto implementare un sistema di ricerca di prodotti, ma per mancanza di tempo e complessità, è stato finalmente implementato un sistema di filtraggio del prodotto, che è più facile.

2 METOLOGIA DI SVILUPPO

Con l'obbiettivo di avere un Per lo sviluppo dell'applicazione si ha utilizzato l'approccio Test Driven Development (TDD), utilizzando questo metodo permette realizzare un codice più pulito e puoi evitare di aggiungere codice o elementi dell'interfaccia utente che non sono necessari per la soluzione, allo stesso tempo che fai i test.
Quindi, prima di tutto, per ogni funzionalità dell'app sono stati specificati i requisiti che devono soddisfare e poi per ogni requisito sono stati specificati i test.

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

3 FUNZIONALITÀ DELL'APP

- Gestire l'account: l'app permette creare un account con e-mail e password, fare login/logout, modificari i dati personali(nome, cognome), cambiare e recuperare la password e infine eliminare l'account.
- Carrello: permette gestire un carrello di prodotti(aggiungere/rimuovere birre)
- Visualizzare catalogo prodotti: il catalogo ha due sezione, uno dei prodotti più venduti e e un altro per vedere tutti. Permette anche di filtrare i prodotti per paese di origine e per bran, e ordinare i prodotti per nome, prezzo e rating.
- Fare ordini: permette acquistare prodotti.
- XPay: uso del SDK XPay per il pagamenti, tramite il Pagamento Semplice
- Storico ordini: permette visualizzare lo storico degli ordini dell'utente, dove puoi vedere il totale pagato, la data di acquisto, l'indirizzo di spedizione, i prodotti acquistati e esito della operazione.
- Gestire prodotti: aggiungere/rimuovere/modificare prodotti del catalogo per gli amministratori.
- Storico pagamenti: Per amministratori, visualizzare lo storico di tutti pagamenti ricevuti e di stornare/rimborsare un pagamento tramite API Nexi.
- Reviews: scrivere recensioni di prodotti e vedere recensioni di altri utenti.

4 ARCHITETTURA DELL'APP

L'applicazione è strutturata seguendo il pattern architetturale Model-View-Controller (MVC). Sebbene in realtà, il modello Model-View-Presenter (MVP), è stato utilizzato nell'implementazione, che è una derivazione del MVC. Questa decisione è dovuta al fatto che MVP consente una maggiore separazione tra la vista e la logica dei dati, e facilita l'esecuzione dei test unitari.

Una possibile alternativa è Model-View-Viewmodel (MVVM), è più recente. In MVP è facile da imparare, modificare, aggiungere funzionalità ma in MVVM è più complesso per gli sviluppatori con poca esperienza. Il debug è più facile in MVP che in MVVM, questo perché in MVVM alcune parti del codice finiscono in XML e anche devi lavorare sulla View in due modi (DataBinding e/o funzioni della View). MVVM consente test unitari più potenti, ma la scrittura di tali test è un'attività più complessa. Però, la quantità di codice, classi e interfacce richieste è maggiore in MVP, e in MVVM la stessa Viewmodel può essere utilizzata in per diversi View.

Poiché i dati devono essere ottenuti da un server, l'applicazione funge da client e Firebase viene utilizzato da server.

Firebase fornisce più servizi cloud che possono essere utilizzati gratuitamente, ma ha dei limiti, ad esempio, nel numero di scritture al secondo nel database. Si usa Firebase Authentication per registrare e autenticare gli utenti tramite e-mail e password. Firebase Cloud Firestore viene utilizzato come database, è NoSQL. Firebase Cloud Storage viene utilizzato per archiviare immagini dei prodotti nel cloud. Infine, Firebase Functions è usato per scrivere funzioni che si attivano automaticamente quando si verifica un evento nel Cloud Firestore e/o nel Cloud Storage.

Una possibile alternativa a Firebase è creare un server da zero usando ad esempio java o php, e utilizzare MySQL come database. Questa opzione richiede più tempo di sviluppo e test approfonditi rispetto a Firebase, ma l'uso di un database SQL consente di realizzare query più complesse e più veloci di quelle di Cloud Firestore (NoSQL).

5 LIBRERIE UTILIZZATE

- Picasso: libreria utilizzata per scaricare le immagini dei prodotti. È molto semplice e facile di utilizzare, il più delle volte è sufficiente una singola riga di codice per scaricare un'immagine. Si occupa dell'intero processo dal download all'inclusione in un ImageView, gestendo i possibile errori. Fa anche la memorizzazione nella cache. MVP

  Una possibile alternativa è scaricare le immagini tramite la libreria di Cloud Storage, utilizzata per caricare le immagini sul cloud, e una libreria verrebbe utilizzata per gestire le immagini invece di due e avrebbe un maggiore controllo sul processo di download, ma questo richiede implementare molte operazioni diversi e fare molti test, e anche il download tramite Picasso è più veloce.
- Gson: è una libreria Java per serializzare e deserializzare oggetti Java su (e da) JSON. Una alterniva è Jackson, entrambe sono praticamente uguali, se hai bisogno di lavorare in un modo più complesso forse Jackson è meglio perché offre più operazioni.
- Firebase: libreria utilizzata per connettersi ed eseguire operazioni sui servizi Firebase, è l'unica libreria disponibile se vuoi utilizzare Firebase.

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

6 STRUTTURA DEL CODICE

Classi più importante:

- FBModel: contiene operazioni da eseguire sul database, sono operazioni per gestire l'account utente, come login, logout, cambiare password, modificare dati personalli…
- DBProducts: contiene operazioni sul database, per gestire i prodotti (ottenere prodotti disponibili, aggiungere / rimuovere / modificare prodotti e gestire i brand)
- DBOrders: operazioni sul database per gestire ordini.
- DBCart: operazioni sul database per gestire il carrello.
- XPayModel: classe utilizzata per gestire il SDK XPay e permette realizzare pagamenti tramite questo SDK.
- XPayBackOffice: classe per collegarsi al backoffice di Nexi, permette ottenere lo storico di tutti i pagamenti ricevuti e di stornare/rimborsare un pagamento.
- BaseActivity: classe astratta che funge da base per tutte le Activity dell'app
- BaseFragment: classe astratta che funge da base per tutte le Fragment dell'app

Interfacce più importante:

- IFBConnector: interfaccia che le classi devono implementare per collegarsi al databse tramite le classi che contengono operazioni sul DB (FBModel, DBProducts…), ha due funzioni una per quando la risposta è corretta e un'altra per quando è sbagliata.
- IXPayContoller.Payment: interfaccia con un'operazione per ricevere la risposta dal XPayModel quando un pagamento è stato completato.
- IXPayContoller.BackOffice: interfaccia con due operazione per ricevere la risposta dal XPayBackOffice quando un'operazione viene eseguita sul backoffice.

Sommario:

	Numero	Linee di codice
Classi	102	7841
Interfacce	3	19
Risorse xml	73	
Immagini png	6	

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603

7 TEST EFFETTUATI

Come si ha utilizzato l'approccio TDD, quindi numerosi test sono stati effettuati durante lo sviluppo per ogni funzionalità.

L'applicazione è stata testata su più dispositivi, sia fisici che virtuali, con diverse versioni di Android e dimensioni e risoluzioni dello schermo diverse. Per questo, è stato utilizzato AVD (Android Virtual Device) per dispositivi virtuali e dispositivi personali e il Test Lab di Firebase per i fisici.

Alcuni dispositivi virtuali:

- Nexus One: Android Marshmallow (6), Risoluzione 480 x 800 252 PPI 3,7''
- Pixel 3 XL: Android Q(10), Risoluzione 1440 x 2960 QHD+ 523 PPI 6,3''
- Nexus 5X: Android Pie(9), Risoluzione 1080 x 1920 FHD 424 PPI 5,2''
- Pixel 3a: Android Nougat(7.1), Risoluzione 1080 x 2220 440 PPI 5,6''

Dispositivi fisici personali:

- Motorola Moto G: Android Lollipop (5.1), Risoluzione 720 x 1280 HD 329 PPI 4,5''
- Huawei P10 lite: Android Oreo (8), Risoluzione 1080 x 1920 FHD 424 PPI 5,2''
- Xiaomi Mi 9T: Android Q (10), Risoluzione 1080 x 2340 FHD+ 403 PPI 6,39''

Robo è stato utilizzato in Test Lab, che consente di eseguire test. Da un lato, sono stati eseguiti test con comportamento casuale e, dall'altro, i test sono stati registrati in Android Studio e successivamente caricati in Test Lab.

Alcuni test registrati ed eseguiti su Robo sono:

- Effettuare un ordine, con esito positivo e negativo
- Scrivere una recensione
- Rimuovere/Modificare un prodotto
- Stornare/Rimborsare un pagamento
- Resgistrare/Eliminare un account

Per testare il pagamento tramite XPay, Nexi fornisce numerose carte di credito di prova (VISA e MasterCard). Ogni carta dà un risultato diverso al momento del pagamento, per esempio pagamento corretto, authentication denied, not sufficient founds…, con questo si verifica che al termine del pagamento l'applicazione risponda correttamente ai possibili risultati.

È stato anche testato a perdere la connessione in diverse parti dell'applicazione, per verificare che la situazione è controllata. Inserisci dati errati, ad esempio un e-mail formattata in modo errato, campi vuoti… Transizioni veloci tra schermate, più clic sullo stesso pulsante.

Dipartimento di Informatica, C.so Svizzera 185. Tel. 0116706711 Fax. 011751603