Software Testing

Introduzione

In questo documento verranno trattati i seguenti argomenti: * Criterio scelto per la stesura dei requisiti di test * Fonte di informazioni per derivare i casi di test * Casi di test per il software **chatByPics** * Misura della copertura per i casi di test

Criterio per la stesura dei casi di test

I requisiti di test per l'applicazione **chatByPics** sono stati formulati seguendo il criterio della **copertura del prodotto** da testare. In particolare si è voluto andare a testare principalmente il funzionamento dei pulsanti e dei widget grafici per le schermate principali dell'applicazione.

Durante la scrittura dei casi di test sono state riscontrate delle criticità, in quanto le funzioni che vanno a effettuare chiamate a Firestore Database producono un risultato che fa fallire il test, dato che la classe che effettua i casi di test non è in grado di interfacciarsi direttamente con Firestore DB.

Questo ci ha portato alla decisione di concentrare i casi di test sulla parte grafica e funzionamento dei pulsanti (*test manuale basato sull'analisi dinamica*). Le funzioni che portavano a interfacciarsi con Firebase sono state testate in modo manuale attraverso il debug dell'applicazione mentre girava nel simulatore (IOS/AndroidStudio)

Fonte di informazioni

Per derivare i casi di test poichè non abbiamo potuto effettuare chiamate a Firebase, ci siamo concentrati sul derivare i casi di test partendo dalla struttura logica interna del software, seguendo il modello **white-box testing**.

Casi di test

Vengono ora divisi i casi di test per pagina dell'applicazione

HomePage.dart

Di seguito si trovano i casi di test relativi alla "*homePage*" dell'applicazione.

Test 1

Questo test va a verificare l'esistenza dei pulsanti di navigazione * Chats * Utenti CCN * Impostazioni

Posti nella barra di navigazione in basso nello schermo, per un **Utente Tutor**. I test vengono effettuati simulando la creazione della homePage con il ruolo di tutor, per testare che il tutor possa vedere il bottone "*Utenti CCN*" per accedere alla pagina di gestione. I test di cui viene riportato il codice sotto vengono svolti nel seguente ordine: 1. Verifica dell'esistenza del bottone **Chat** cercando il testo associato al bottone 2. Verifica dell'esistenza del bottone **Utenti CCN** cercando il testo associato al bottone 3. Verifica dell'esistenza del bottone **Impostazioni** cercando il testo associato al bottone 4. Verifica che i bottoni presenti siano 3 ``` testWidgets('Homepage per Tutor mostra 3 tab (Chats, Utenti CCN, Impostazioni)', (WidgetTester tester) async { // Impostiamo una dimensione schermo realistica per evitare errori grafici

tester.view.physicalSize = const Size(1080, 2400); tester.view.devicePixelRatio = 3.0;

```
//creazione della home page con il ruolo fittizio di tutor
await tester.pumpWidget(const MaterialApp(
  home: Homepage(testRole: "Tutor"),
));

expect(find.text('Chats'), findsOneWidget);
expect(find.text('Utenti CCN'), findsOneWidget);
expect(find.text('Impostazioni'), findsOneWidget);
expect(find.byType(NavigationDestination), findsNWidgets(3));

// Pulizia dimensione schermo
addTearDown(tester.view.resetPhysicalSize);
```

}); ```

Test 2

Questo test va a verificare l'esistenza dei pulsanti di navigazione * Chats * Impostazioni

Posti nella barra di navigazione in basso nello schermo, per un **Utente CCN** e **Utente Comunication partner**. ``` testWidgets('Homepage per Utente normale mostra solo 2 tab', (WidgetTester tester) async { tester.view.physicalSize = const Size(1080, 2400); tester.view.devicePixelRatio = 3.0;

```
//creazione della home page con il ruolo fittizio di utente
await tester.pumpWidget(const MaterialApp(
  home: Homepage(testRole: "Utente"),
));

expect(find.text('Chats'), findsOneWidget);
expect(find.text('Impostazioni'), findsOneWidget);
expect(find.text('Utenti CCN'), findsNothing);
expect(find.byType(NavigationDestination), findsNWidgets(2));

addTearDown(tester.view.resetPhysicalSize);
```

}); ``` I test di cui viene riportato il codice sotto vengono svolti nel seguente ordine: 1. Verifica dell'esistenza del bottone **Chat** cercando il testo associato al bottone 2. Verifica dell'esistenza del bottone **Utenti CCN** cercando il testo associato al bottone 3. Verifica dell'esistenza del bottone **Impostazioni** cercando il testo associato al bottone 4. Verifica che i bottoni presenti siano 2

Test 4

Questo test va a verificare se cambia la pagina cliccando sul bottone e se viene aggiornato il selettore della pagina evidenziandolo con la grafica. ``` testWidgets('Cliccando su una tab la selezione cambia', (WidgetTester tester) async { tester.view.physicalSize = const Size(1080, 2400); tester.view.devicePixelRatio = 3.0;

```
await tester.pumpWidget(const MaterialApp(
  home: Homepage(testRole: "Utente"),
));

//cerca il bottone impostazioni
final settingsTab = find.text('Impostazioni');
await tester.tap(settingsTab);
await tester.pump(); // Ridisegna l'interfaccia
```

```
//verifico che si sia aggiornato il selected index
final navBar = tester.widget<NavigationBar>(find.byType(NavigationBar));
expect(navBar.selectedIndex, 1);

addTearDown(tester.view.resetPhysicalSize);
```

}); ``` Nota: per poter effettuare il test della homepage, si è dovuto modificare il codice della homePage.dart per impedire di chiamare subito firebase, ma passandogli un ruolo fittizio viene simulato l'utilizzo dell'applicazione da quel tipo di utente bypassando completamente firebase, garantendo flessibilità per poter scrivere i casi di test.

AuthPage.dart

Test 1

Questo test va a verificare che alla prima apertura dell'apllicazione siano correttamente visualizzati i campi per la registrazione, che comprende: * campo per l'inserimento del nome * campo per l'inserimento del cognome * flag per la registrazione con il ruolo di tutor * testo cliccabile per passare alla schermata di accesso in caso di registrazione già avvenuta ``` testWidgets('AuthPage parte in modalità Registrazione', (WidgetTester tester) async { await tester.pumpWidget(const MaterialApp(home: AuthPage(),));

```
expect(find.text('Nome'), findsOneWidget);
expect(find.text('Cognome'), findsOneWidget);
expect(find.text('Registrati come Tutor'), findsOneWidget);


expect(find.text('Registrati'), findsOneWidget); // Deve dire "Registrati"

expect(find.text('Hai già un account? Accedi'), findsOneWidget);
```

}); ```

Test 2

Questo test verifica il corretto passaggio alla schermata di login, dove verranno non dovranno essere visualizzati i: * campo per l'inserimento del nome * campo per l'inserimento del cognome ``` // TEST 2: Verifica il passaggio alla modalità LOGIN testWidgets('Cliccando su accedi scompare il nome e resta solo email/password', (WidgetTester tester) async { await tester.pumpWidget(const MaterialApp(home: AuthPage(),));

```
final toggleButton = find.text('Hai già un account? Accedi');
await tester.tap(toggleButton);
await tester.pump(); // Ridisegna la pagina

expect(find.text('Accedi'), findsOneWidget); // Il bottone ora dice "Accedi"

expect(find.text('Nome'), findsNothing);
expect(find.text('Cognome'), findsNothing);
```

}); ```

Test 3

Questo test va a verificare che venga visualizzata l'icona per mostrare/nascondere la password e che venga cambiata dinamicamente quando si clicca per visualizzare la password ``` testWidgets('Cliccando l\'occhio della password cambia l\'icona', (WidgetTester tester) async { await tester.pumpWidget(const

MaterialApp(home: AuthPage()));

```
expect(find.byIcon(Icons.visibility), findsOneWidget);

await tester.tap(find.byIcon(Icons.visibility));
await tester.pump();

expect(find.byIcon(Icons.visibility_off), findsOneWidget);
```

}); ```

Test 4

Questo test va a verificare che venga registrata la spunta sulla modalità di registrazione come tutor. ``` testWidgets('La checkbox Tutor si attiva e disattiva correttamente', (WidgetTester tester) async { await tester.pumpWidget(const MaterialApp(home: AuthPage(),));

```
final checkboxFinder = find.widgetWithText(CheckboxListTile, "Registrati come Tutor");

expect(checkboxFinder, findsOneWidget);

CheckboxListTile checkboxWidget = tester.widget(checkboxFinder);
expect(checkboxWidget.value, false); // Di default isTutor = false

await tester.tap(checkboxFinder);
await tester.pump(); // aggiorna la schermata per vedere la spunta cliccata

checkboxWidget = tester.widget(checkboxFinder);
expect(checkboxWidget.value, true);

await tester.tap(checkboxFinder);
await tester.pump();

checkboxWidget = tester.widget(checkboxFinder);
expect(checkboxWidget.value, false);
```

}); ```

Test 5

Successivamente viene verificato l'inserimento non corretto di un email da parte dell'utente ``` group('Test Validazione Email', () { test('Deve ritornare true se la mail è corretta', () { String email = "test@example.com"; bool result = isEmailValid(email); expect(result, true); //expect è l'equivalente della funzione assertEquals per i test junit });

```
test('Deve ritornare false se manca la chiocciola', () {
  expect(isEmailValid("testexample.com"), false);
});

test('Deve ritornare false se vuota', () {
  expect(isEmailValid(""), false);
});
```

}); ```

AddCcnPage

Test 1

Questo test va a verificare che siano correttamente presenti i campi per la registrazione del CCN attraverso: * campo per l'inserimento del nome * campo per l'inserimento del cognome * campo per l'inserimento dell'email * campo per l'inserimento dalla password * bottone di registrazione ``` testWidgets('AddCcnPage mostra i 4 campi di testo e il bottone', (WidgetTester tester) async { await tester.pumpWidget(const MaterialApp(home: AddCcnPage(),));

```
expect(find.text('Nome'), findsOneWidget);
expect(find.text('Cognome'), findsOneWidget);
expect(find.text('Email'), findsOneWidget);
expect(find.text('Password provvisoria'), findsOneWidget);
expect(find.text('Registra Utente CCN'), findsOneWidget);
```

}); ```

Test 2

Questo test va a verificare che se si preme il tasto di registrazione del ccn con i campi vuoti vengono visualizzati correttamente gli errori a schermo ``` testWidgets('Premendo registra con campi vuoti appaiono gli errori', (WidgetTester tester) async { await tester.pumpWidget(const MaterialApp(home: AddCcnPage()));

```
await tester.tap(find.text('Registra Utente CCN'));
await tester.pump();

expect(find.text('Inserisci il nome'), findsOneWidget);
expect(find.text('Inserisci il cognome'), findsOneWidget);
expect(find.text('Email non valida'), findsOneWidget);
expect(find.text('Minimo 6 caratteri'), findsOneWidget);
```

}); ```

Test 3

Questo test va a verificare che vengano correttamente visualizzati gli errori inserendo: * email non valida (senza @) * password troppo corta (6 caratteri lunghezza minima) ``` testWidgets('Mostra errori specifici per Email non valida e Password corta', (WidgetTester tester) async { await tester.pumpWidget(const MaterialApp(home: AddCcnPage()));

```
await tester.enterText(find.widgetWithText(TextFormField, 'Nome'), 'Mario');
await tester.enterText(find.widgetWithText(TextFormField, 'Cognome'), 'Rossi');

await tester.enterText(find.widgetWithText(TextFormField, 'Email'), 'mariorossi.it');

await tester.enterText(find.widgetWithText(TextFormField, 'Password provvisoria'), '123');

await tester.tap(find.text('Registra Utente CCN'));

await tester.pump();

expect(find.text('Inserisci il nome'), findsNothing);
expect(find.text('Email non valida'), findsOneWidget);
expect(find.text('Minimo 6 caratteri'), findsOneWidget);
```

}); ```

Copertura

I risultati del test di copertura sono riportati nella foto sottostante.