



TESTY AUTOMATYCZNE APLIKACJI MOBILNYCH

Dagmara Surma, Jakub Kuc, Piotr Szczęsny

Witamy!



Dagmara Surma

Software Quality
Assurance Engineer

Future Processing



Jakub Kuc

Android Software
Engineer

Future Processing



Piotr Szczęsny

Software Quality
Assurance Engineer

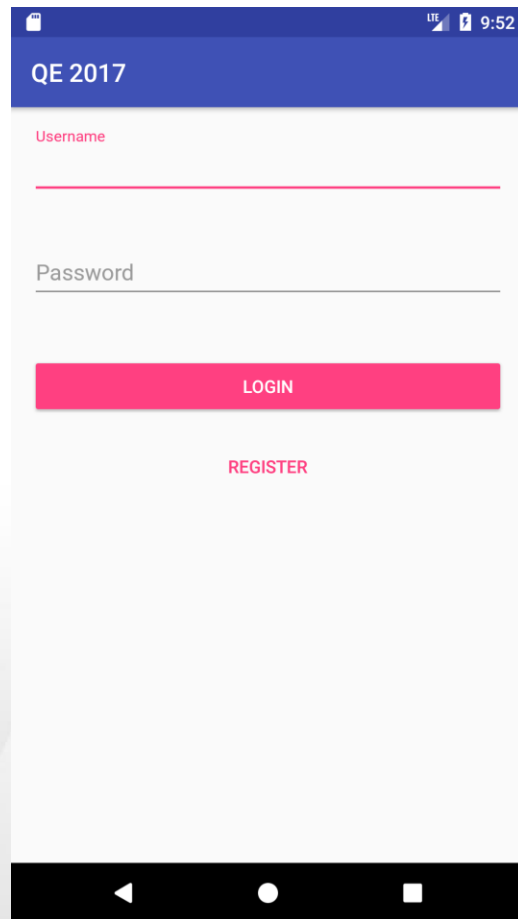
Future Processing

Cel warsztatu

Zapoznanie z frameworkiem Espresso oraz popularnymi wzorcami projektowymi ułatwiającymi pracę z testami automatycznymi.

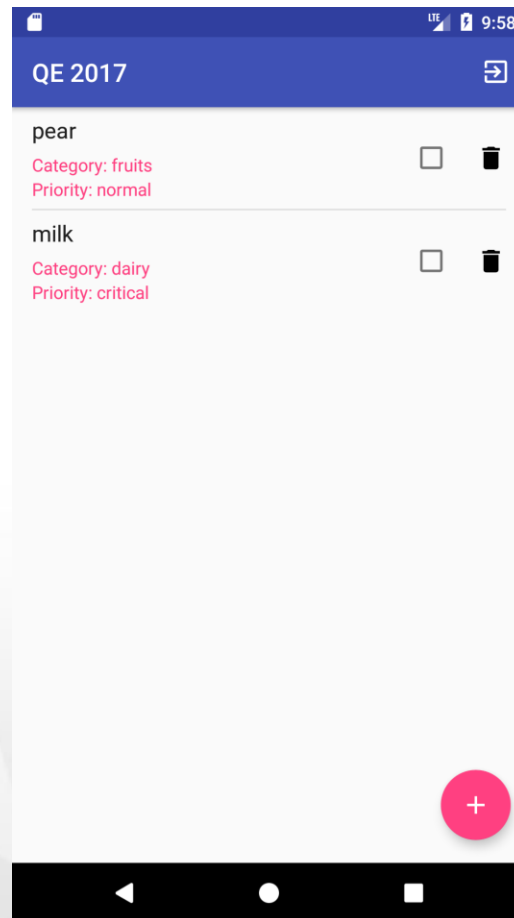
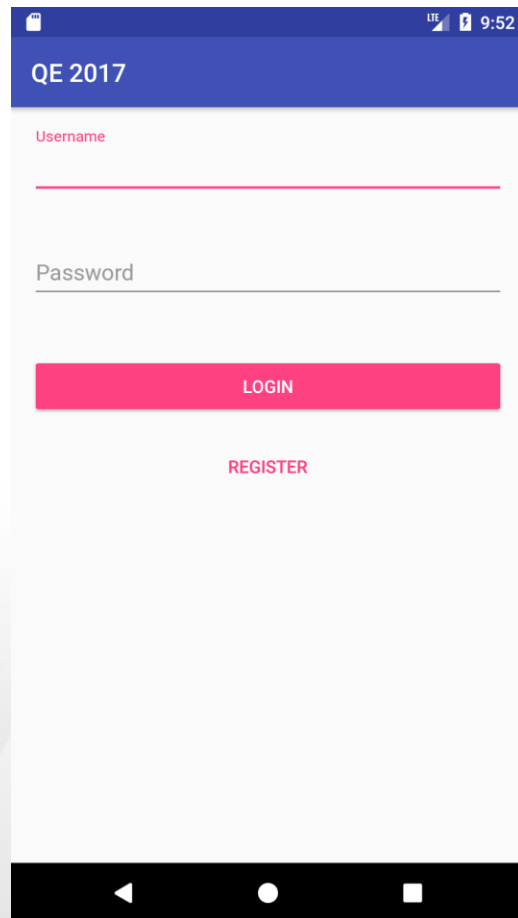
Aplikacja

Aplikacja



Mobile app login screen titled "QE 2017". The screen features a blue header bar with the title. Below the header, there are two input fields: "Username" and "Password". The "Username" field is highlighted with a red underline. Below the input fields, there is a red "LOGIN" button and a red "REGISTER" link. The screen is displayed on a mobile device interface, showing status bar icons (LTE, signal, battery, 9:52) at the top and Android navigation bar icons at the bottom.

Aplikacja



Warsztat

Czyli z czym będziemy dziś pracować

- Android
- Espresso
- JUnit
- Java
- Android Studio

JUnit

@RunWith - pozwala ustawić runner dla klasy testowej

@Before - oznaczenie metody wywoływanej przed każdym testem

@After - oznaczenie metody wywoływanej po każdym teście

@Rule - w Espresso jest odpowiedzialne za obsługę cyklu życia Activity

@Test - oznaczenie metody testowej

Espresso

Stworzone po to by pisać zwięzłe, piękne i niezawodne testy interfejsu użytkownika

```
@Test
public void greeterSaysHello() {
    onView(withId(R.id.name_field))
        .perform(typeText("Steve"));
    onView(withId(R.id.greet_button))
        .perform(click());
    onView(withText("Hello Steve!"))
        .check(matches(isDisplayed()));
}
```

```
onView(withId(R.id.my_view))      //(withId(R.id.my_view) is a ViewMatcher)
    .perform(click())              //click() is a ViewAction
    .check(matches(isDisplayed())); //matches(isDisplayed()) is a ViewAssertion
```

```
onView(withId(R.id.my_view))      //(withId(R.id.my_view) is a ViewMatcher)
    .perform(click())              //click() is a ViewAction
    .check(matches(isDisplayed())); //matches(isDisplayed()) is a ViewAssertion
```

Espresso - podstawowy punkt wejścia do frameworku

```
onView(withId(R.id.my_view))      //(withId(R.id.my_view) is a ViewMatcher)
    .perform(click())              //click() is a ViewAction
    .check(matches(isDisplayed())); //matches(isDisplayed()) is a ViewAssertion
```

Espresso - podstawowy punkt wejścia do frameworku

```
onView(withId(R.id.my_view))      //(withId(R.id.my_view) is a ViewMatcher)
    .perform(click())              //click() is a ViewAction
    .check(matches(isDisplayed())); //matches(isDisplayed()) is a ViewAssertion
```

Espresso - podstawowy punkt wejścia do frameworku

ViewMatchers - pozwalają wyszukiwać widoki po podanym warunku

ViewActions - pozwalają wykonywać akcje na widokach

```
onView(withId(R.id.my_view))      //(withId(R.id.my_view) is a ViewMatcher)
    .perform(click())              //click() is a ViewAction
    .check(matches(isDisplayed())); //matches(isDisplayed()) is a ViewAssertion
```

Espresso - podstawowy punkt wejścia do frameworku

ViewMatchers - pozwalają wyszukiwać widoki po podanym warunku

ViewActions - pozwalają wykonywać akcje na widokach

ViewAssertions - pozwalają zapisywać “wymagania” względem widoków

```
onView(withId(R.id.my_view))      //(withId(R.id.my_view) is a ViewMatcher)
    .perform(click())              //click() is a ViewAction
    .check(matches(isDisplayed())); //matches(isDisplayed()) is a ViewAssertion
```

Espresso - podstawowy punkt wejścia do frameworku

ViewMatchers - pozwalają wyszukiwać widoki po podanym warunku

ViewActions - pozwalają wykonywać akcje na widokach

ViewAssertions - pozwalają zapisywać “wymagania” względem widoków

ViewInteraction - obiekt pośredni interakcji

```
onView(  
    allOf(  
        withId(R.id.checkbox),  
        hasSibling(viewMatcher),  
        withChild(viewMatcher)  
    )  
);
```

*// or anyOf()
// find view with given id
// find sibling view with given matcher
// find view with child with given matcher*

Piszemy testy!!

Dodajemy dobrych praktyk

Page object

- Metody publiczne odpowiadają akcjom które są dostępne na ekranie,
- Nie powinno się wystawiać „na zewnątrz” struktury ekranu(kontrolerek etc.)
- Nie powinno się wykonywać w nich asercji
- Metody mogą zwracać inne PageObject’y
- Nie muszą reprezentować całej strony

Object factory

- Dostarcza obiekty
- Zajmuje się ich odpowiednią inicjalizacją
- Upraszcza kod testu

Transporter

- Izoluje kod nawigacji wewnątrz aplikacji
- Minimalizuje wpływ zmiany flow aplikacji na nasze testy
- Poprawia czytelność testów

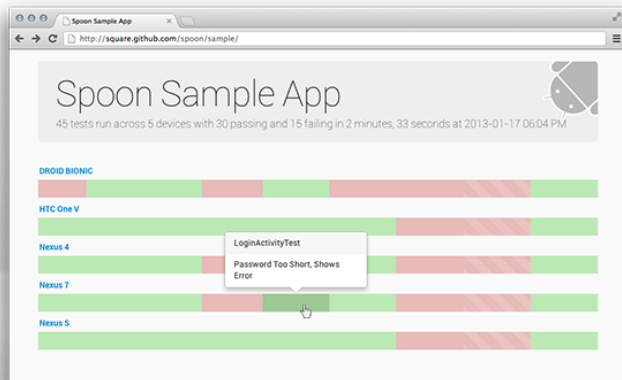
!! Czas na refaktor !!

Spoon

- Pozwala wykonywać testy równoległe na wielu urządzeniach/emulatorach

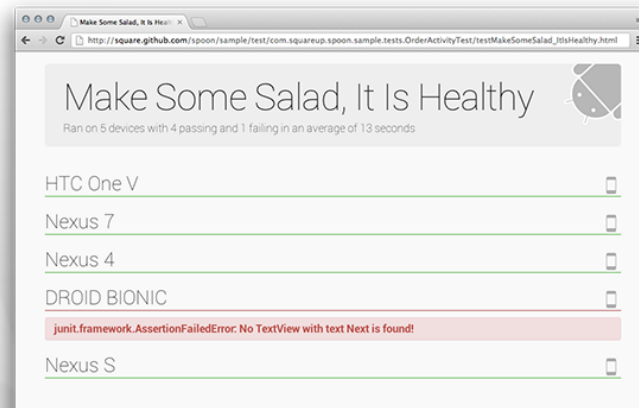
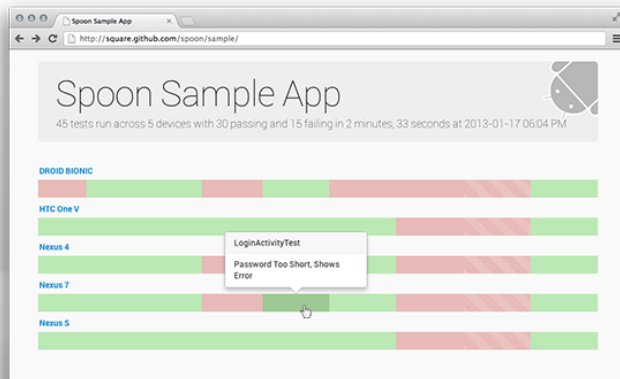
Spoon

- Pozwala wykonywać testy równoległe na wielu urządzeniach/emulatorach.
- Udostępnia wyniki w przystępnej formie



Spoon

- Pozwala wykonywać testy równoległe na wielu urządzeniach/emulatorach.
- Udostępnia wyniki w przystępnej formie.
- W wygodny sposób umożliwia pobieranie screenshotów z wykonywanych testów.



Napiszcie do nas!

Janka janinam.mazur@gmail.com

Kuba jkuc@future-processing.com

Piotr pszczeny2@future-processing.com

Kod aplikacji, testów oraz materiały:

<https://bit.ly/2Xf63aB>



QUALITY
EXCITES

DZIĘKUJĘ ZA UWAGĘ