UNIKASSEL
VERSITÄT

SOFTWARE ENGINEERING
UNIVERSITÄT KASSEL

Softwaretechnik-Praktikum
Vorlesung 03-07

Übersicht der Themen und Technologien

Sommersemester 2024
Adrian Kunz

VL 03: GitHub und Login

- Anforderungen
  - Erklärungen
  - Server-Dokumentation (Swagger, Rate Limit, Auth)
- GitHub Projects
  - Aufgaben/Stories/Subtasks
  - Wireframes
  - Story Point Schätzung (Scrum Poker)
- App Init (Main, App)
- Login
  - FXML, Controller, DTOs
  - fulibFx: @Controller, Routes
  - Dagger: @Inject, @Provides
  - Retrofit: AuthApiService
  - RxJava: Observable

VL 04: Pull Requests und Signup

- Login
  - LoginService
  - Retrofit: OkHttpClient, Interceptors
  - TokenStorage
  - fulibFx: @Title
- Tests
  - ControllerTest
  - Mockito: @Spy/@Mock/@InjectMocks, when/doReturn, verify
  - Dagger: TestModule @Provides mit mock
- GitHub Pull Requests
  - CI
  - Coverage
  - Review
- Signup
  - Retrofit: @GET, @Query, @Path
  - fulibFx: @Param, @OnInit, @OnRender
  - JavaFx: Binding/Validierung

VL 05: WebSocket, Listen, Remember Me

- Lobby
  - JavaFX: ListView, ObservableList
  - fulibFx: Subscriber, @OnDestroy
- User Liste
  - fulibFx: ComponentListCell, @Component, ReusableItemComponent
  - FXML: fx:root
- WebSocket
  - Event, ClientEndpoint, EventListener
- ImageCache
  - Data URIs
- Remember Me
  - Preferences
  - Dagger: @Component MainComponent
  - Logout
- WebSocket Testing
  - RxJava: Subject

VL 06: CSS und Localization

- CSS
  - JavaFX: Selectors, Styling, styleClass
  - CSSFX
  - fulibFx: refresh (F5, FXML reload)
  - CSS: Border Image Slice
- Übersetzung (Localization)
  - JavaFX: ResourceBundle
  - fulibFx: @Resource
  - Dagger: MainModule @Provides ResourceBundle (abh. PrefService), @Named, wann @Singleton
  - ControllerTest: @Spy ResourceBundle

VL 07: Restekiste

- Response Handling
  - Retrofit: HttpException
  - ErrorService, ErrorResponse, ErrorResponseException
  - RxJava: subscribe Error Handling
  - RxJava: Observable.error
- Subcomponents
  - fulibFx: @SubComponent
  - Test mit @InjectMocks
  - @Spy Provider
  - Dagger: Konstruktor Injection
- Testing
  - Dagger: TestComponent, TestModule
  - AppTest: Kritischer Pfad
  - Headless Testing: Debug Screenshots
  - Service Tests
  - Mockito: ArgumentCaptor

Glossar: fulibFx

- @Controller – Controller-Klasse, verwaltet View, automatisch mit FXML, nach MVC Pattern
- @Component – Wiederverwendbare Komponente, ist gleichzeitig View, FXML optional, nach MVVM Pattern
- @SubComponent – Einbettung von Component in größerem Controller/Component, inkl. automatischer Lifecycle
- @Route – Definiert Route zu einem Controller/Component
- @Title – Setzt Fenstertitel für Controller/Component
- @Param – Eingabeparameter für Controller/Component
- @OnInit, @OnRender, @OnDestroy – Lifecycle-Methoden für Controller/Component
- Subscriber – Hilfsklasse für Observables/Disposables, wiederverwendbar
- ComponentListCell / ReusableItemComponent – wiederverwendbare ("recycle") Elemente von ListViews

Glossar: Dagger

- @Inject – Deklariert die Abhängigkeit von einer Dependency (z.B. Service)
- @Provides – Methode, die eine Dependency erzeugt, welche Dagger nicht unterstützt (z.B. Retrofit ApiService, ResourceBundle, …)
- @Component – Einstiegspunkt für Dependency Injection, "Wurzel des Dependency Baums", via DaggerXComponent
- @Named – Benannte Dependency, falls unterschiedliche Instanzen vom selben Typ (z.B. ResourceBundle)
- @Singleton – Einzigartige Instanz, globaler Zustand (böse – nur für zustandslose Services, Tests, und einzige Ausnahme TokenStorage)

Glossar: Retrofit

- @GET – Lesende HTTP-Anfrage
- @POST, @PUT, @PATCH, @DELETE – Schreibende HTTP-Anfrage
- @Query – Query-Parameter, z.B. GET /foo?bar=baz
- @Path – Pfad-Parameter, z.B. GET /users/{user} mit user=1 => /users/1
- @Body – Rumpf der HTTP-Anfrage (bei POST, PUT, PATCH, etc.)
- OkHttpClient – Zugrundeliegender http Client (OkHttp Library)
- Interceptors – Zwischenschritte, die die Anfrage bearbeiten können (z.B. Header hinzufügen)

Glossar: RxJava

- Observable – beobachtbarer Datenstrom
- doOnNext, map, zip, switchMap, … – Operationen auf Observables
- Subject – Datenstrom mit eigener Eingabe / Anstoßen von Events

Glossar: Mockito

- @Spy – erzeuge echtes Objekt mit realem Verhalten, erlaube Überprüfung mit verify (z.B. für zustandsvolle Services)
- @Mock – erzeuge Mock-Objekt (Fake) ohne Verhalten, erlaube Definieren von Verhalten mit mock und Überprüfung mit verify (z.B. für zustandslose Services)
- @InjectMocks – erzeuge echtes Objekt, injecte @Spy- und @Mock-Objekte in @Inject-Attribute oder Konstruktor (für das Testsubjekt)
- when(obj.someMethod(any())).thenReturn(result) – Definiere Verhalten von Mock
- doNothing().when(obj).someMethod(any()) – Definiere Verhalten von Mock bei void-Methoden
- verifyThat(obj, times(1)).someMethod(expectedArg) – Prüfe Verhalten von Mock/Spy
- ArgumentCaptor – "Einfangen" von Parametern bei verify, um sie später mit Assertions zu prüfen