Psyche Assistant

Bilag 7: Klasse oversigt

Indholdsfortegnelse

1. Introduktion

Dette bilags formål er primært at kunne give en kort oversigt over den interne struktur for hver klasser i front- og back-end-delene. Fokus er dermed ikke på hvordan klassens funktioner reelt udfører deres funktionalitet. Intentionen er blot at give et mere kortfattet og opsummeret overblik, over udformningen – og ikke nødvendigvis funktionaliteten.

Specifikt vedrørende front-end-delen henvises der til endelige rapport, afsnit 5.4, hvor selve visnings-funktionerne (Composables) gennemgåes.

Bemærk at evt. felter samles i én række, imens funktioner får hver sin.

Ligeledes samles funktioner fra interfaces i én række.

Endelig noteres kotlin platforms unikke expect/actual funktioner kun ved førnævnte, da tanken bag disse er, at der er én central implementering, der så implementeres unikt for hver platform. Af hensyn til overblikket, noteres de derfor kun den ene, centrale gang, og f.eks. ikke i den Android-specifikke kode, og hvor relevant noteres der, når en klasse implementerer eller nedarver hhv. et interface eller anden klasse.

Indekseringen følger filstrukturen, og ikke en semantisk relation, hvorfor en models repository kan komme efter dens egentlige model definition, imens dens controller kommer før; for eksempel hvis de er blevet opdelt i funktionelle (alle controllere, alle repositories) og ikke semantiske (alt bruger relateret) mapper.

2. Front-end

2.1. ActivityController

private val activityService = ActivityService()
suspend fun getHandledActivityByPeriod(groupId: Int, startDate: LocalDate, endDate: LocalDate): List<Activity>
suspend fun getActivityByPeriod(groupId: Int, startDate: LocalDate, endDate: LocalDate): List<Activity>
suspend fun getActivityForToday(groupId: Int, today: LocalDate): List<Activity>
suspend fun createActivity(authToken: String, deadline: String, description: String, energyCost: Int): Activity?
suspend fun completeActivity(authToken: String, activityId: Int): Activity?
suspend fun deleteActivity(authToken: String, activityId: Int): Boolean

2.2. GroupController

private val groupService = GroupService()
suspend fun getGroupDetails(id: Int): Group?

```
suspend fun createGroup(authToken: String): Group?
suspend fun joinGroup(authToken: String, code: String): Group?
suspend fun kickMember(authToken: String, groupId: Int, userId: Int): Group?
suspend fun leaveGroup(authToken: String, groupId: Int): Boolean
suspend fun disbandGroup(authToken: String, groupId: Int): Boolean
```

2.3. UserController

```
private val userService = UserService()
suspend fun getUserProfile(authToken: String): User?
suspend fun registerNewUser(email: String, password: String): String
suspend fun authenticateUser(email: String, password: String): String
suspend fun signOutUser()
suspend fun deleteUser(authToken: String): Boolean
suspend fun getUserById(id: Int): User?
```

2.4. Activity

```
val id: Int,
val description: String,
val energyCost: Int,
val groupId: Int? = null,
val ownerId: Int? = null,
val handledById: Int? = null,
val deadline: LocalDate,
val handledOn: LocalDate? = null,
val completed: Boolean = false
```

2.5. ActivityRepository

```
suspend fun getHandledActivityByPeriod(groupId: Int, startDate: LocalDate, endDate: LocalDate): List<Activity>
suspend fun getActivityByPeriod(groupId: Int, startDate: LocalDate, endDate: LocalDate): List<Activity>
suspend fun getActivityForToday(groupId: Int, today: LocalDate): List<Activity>
suspend fun getActivity(activityId: Int): Activity?
suspend fun createActivity(authToken: String, deadline: String, description: String, energyCost: Int): Activity?
suspend fun completeActivity(authToken: String, activityId: Int): Activity?
suspend fun deleteActivity(authToken: String, activityId: Int): Boolean
```

2.6. Group

```
val id: Int,
val userIds: List<Int>,
val activityIds: List<Int>,
val ownerId: Int?,
val code: String,
```

2.7. GroupRepository

```
suspend fun createGroup(authToken: String): Group?
suspend fun joinGroup(authToken: String, code: String): Group?
suspend fun leaveGroup(authToken: String, groupId: Int): Boolean
suspend fun disbandGroup(authToken: String, groupId: Int): Boolean
suspend fun getGroupDetails(id: Int): Group?
suspend fun removeUserFromGroup(authToken: String, groupId: Int, userId: Int): Group?
```

2.8. User

```
val id: Int,
val email: String,
val password: String,
val energyExpenditure: Int,
val groupId: Int? = null,
```

2.9. UserRepository

```
suspend fun getUserDetails(authToken: String): User?
suspend fun registerUser(email: String, password: String): String
suspend fun loginUser(email: String, password: String): String
suspend fun logoutUser()
suspend fun deleteUser(authToken: String): Boolean
suspend fun getUserById(id: Int): User?
```

2.10. ActivityService : ActivityRepository

```
private val client = HttpClient()

override suspend fun getHandledActivityByPeriod(groupId: Int, startDate: LocalDate, endDate: LocalDate):
List<Activity>

override suspend fun getActivityByPeriod(groupId: Int, startDate: LocalDate, endDate: LocalDate): List<Activity>

override suspend fun getActivityForToday(groupId: Int, today: LocalDate): List<Activity>

override suspend fun getActivity(activityId: Int): Activity?

override suspend fun createActivity(authToken: String, deadline: String, description: String, energyCost: Int):
```

Activity?
override suspend fun completeActivity(authToken: String, activityId: Int): Activity?
override suspend fun deleteActivity(authToken: String, activityId: Int): Boolean

2.11. ConfigProvider (expect/actual)

val baseUrl: String

2.12. GroupService : GroupRepository

private val client = HttpClient()
override suspend fun createGroup(authToken: String): Group?
override suspend fun joinGroup(authToken: String, code: String): Group?
override suspend fun leaveGroup(authToken: String, groupId: Int): Boolean
override suspend fun disbandGroup(authToken: String, groupId: Int): Boolean
override suspend fun getGroupDetails(id: Int): Group?
override suspend fun removeUserFromGroup(authToken: String, groupId: Int, userId: Int): Group?

2.13. ServiceBuilder

private val BASE_URL = ConfigProvider.baseUrl
val client = HttpClient(CIO)
fun url(path: String): String = "$BASE_URL/$path"

2.14. UserService : UserRepository

private val client = HttpClient()
override suspend fun getUserDetails(authToken: String): User?
override suspend fun registerUser(email: String, password: String): String
override suspend fun loginUser(email: String, password: String): String
override suspend fun logoutUser()
override suspend fun deleteUser(authToken: String): Boolean
override suspend fun getUserById(id: Int): User?
private suspend fun refreshToken(): String?
private fun validateEmail(email: String)
private fun validatePassword(password: String)

2.15. AuthStorage (expect/actual)

```
fun saveAuthToken(token: String)
fun getAuthToken(): String?
fun removeAuthToken()
fun saveRefreshToken(token: String)
fun getRefreshToken(): String?
fun removeRefreshToken()
```

3. Back-end

3.1. ActivityController

```
private ActivityService activityService;
private JwtUtil jwtUtil;
private AuthHelper authHelper;
```

```
public List<Activity> getAllActivities()
```

```
public Activity read(@PathVariable int id)
```

```
public ResponseEntity<Activity> createActivity(
      @RequestHeader("Authorization") String authHeader,
      @RequestBody Map<String, Object> payload)
```

```
public ResponseEntity<Activity> completeActivity(
      @RequestHeader("Authorization") String authHeader,
      @PathVariable int id)
```

```
public ResponseEntity<Boolean> deleteActivity(
      @RequestHeader("Authorization") String authHeader,
      @PathVariable int id)
```

```
public ResponseEntity<List<Activity>> getActivitiesForGroupWithDeadline(
      @PathVariable int groupId,
      @RequestParam("date") String date)
```

```
public ResponseEntity<List<Activity>> getPeriodActivitiesWithDeadline(
       @PathVariable int groupId,
       @RequestParam("startDate") String startDate,
       @RequestParam("endDate") String endDate)
```

```
public ResponseEntity<List<Activity>> getPeriodActivitiesWithHandled(
       @PathVariable int groupId,
       @RequestParam("startDate") String startDate,
       @RequestParam("endDate") String endDate)
```

3.2. GroupController

```
private GroupService groupService;
private UserService userService;
private JwtUtil jwtUtil;
private AuthHelper authHelper;

public List<Group> getAllGroups()

public Group read(@PathVariable int id)

public List<User> getUsersInGroup(@PathVariable int id)

public ResponseEntity<Group> createGroup(@RequestHeader("Authorization") String authHeader)

public ResponseEntity<Group> joinGroup(
    @RequestHeader("Authorization") String authHeader,
    @RequestBody Map<String, String> payload)

public ResponseEntity<Group> kickMember(
    @RequestHeader("Authorization") String authHeader,
    @PathVariable int id,
    @RequestBody Map<String, Integer> payload)

public ResponseEntity<Boolean> leaveGroup(
    @RequestHeader("Authorization") String authHeader,
    @PathVariable int id)

public ResponseEntity<Boolean> disbandGroup(
    @RequestHeader("Authorization") String authHeader,
    @PathVariable int id)
```

3.3. UserController

```
private UserService userService;
private JwtUtil jwtUtil;
private ActivityService activityService;

public List<User> getAllUsers()

public User read(@PathVariable int id)

public ResponseEntity<String> registerUser(@RequestParam String email, @RequestParam String password)

public ResponseEntity<User> getCurrentUser(@RequestHeader("Authorization") String authToken)

public ResponseEntity<Map<String, String>> loginUser(
    @RequestParam String email,
    @RequestParam String password)

public ResponseEntity<String> refreshToken(@RequestHeader("Authorization") String authHeader)

public ResponseEntity<Boolean> deleteUser(@RequestHeader("Authorization") String authHeader)
```

3.4. AuthHelper

```
private final JwtUtil jwtUtil;
private final UserService userService;

public AuthHelper(JwtUtil jwtUtil, UserService userService) {
   this.jwtUtil = jwtUtil;
   this.userService = userService;
```

}
public String extractToken(String authHeader) throws IllegalArgumentException
public String getEmailFromToken(String authHeader) throws IllegalArgumentException
public User getUserFromToken(String authHeader) throws IllegalArgumentException
public ResponseEntity<User> validateAndGetUser(String authHeader)

3.5. Activity

```java
private int id;
private String description;
private int energyCost = 0;
private Group group = null;
private User owner;
private User handledBy = null;
private LocalDate deadline;
private LocalDate handledOn = null;
private boolean completed = false;
```

```java
public Activity(User owner, String description, int energyCost, LocalDate deadline) {
   this.owner = owner;
   this.description = description;
   this.energyCost = energyCost;
   this.deadline = deadline;

   if (owner.getGroup() != null) {
      this.group = owner.getGroup();
   }
}
```

3.6. Group

```java
private int id;
private List<User> users = new ArrayList<>();
private List<Activity> activities = new ArrayList<>();
private User owner;
private String code;
```

```java
public void addActivity(Activity activity)
```

```java
public void removeActivity(Activity activity)
```

```java
public void addUser(User user)
```

```java
public void removeUser(User user)
```

3.7. User

```java
private int id;
private String email;
private String password;
private int energyExpenditure = 0;
private Group group = null;
```

3.8. ActivityRepository (extends JpaRepository)

List<Activity> findActivitiesForGroupWithDeadline
List<Activity> findActivitiesForPeriod
List<Activity> findHandledActivitiesForPeriod
void updateActivitiesHandledByUser(@Param("user") User user)
void updateActivitiesOwnedByUser(@Param("user") User user)

3.9. GroupRepository (extends JpaRepository)

Group findByCode(String code)

3.10. UserRepostiory (extends JpaRepository)

User findByEmail(String email)

3.11. CustomUserDetailsService (implements UserDetailsService)

private UserRepository userRepository
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException

3.12. JwtFilter (extends GenericFilterBean)

private final JwtUtil jwtUtil
public JwtFilter(JwtUtil jwtUtil) { this.jwtUtil = jwtUtil; }
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException

3.13. JwtUtil

private String SECRET_KEY; private long JWT_EXPIRATION; private long REFRESH_TOKEN_EXPIRATION; private UserDetailsService userDetailsService;
public String generateToken(String email)

public String extractEmail(String token)
public boolean validateToken(String token, String email)
private boolean isTokenExpired(String token)
public String generateRefreshToken(String email)
public boolean validateRefreshToken(String refreshToken, String email)
public void setAuthentication(String username, HttpServletRequest request)

3.14. SecurityConfig

private JwtUtil jwtUtil;
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception

public JwtFilter jwtFilter() {
   return new JwtFilter(jwtUtil);
}

3.15. ActivityService

private ActivityRepository activityRepository;
private UserRepository userRepository;
private GroupRepository groupRepository;

public List<Activity> findAll()
public Activity findById(int id)
public Activity createActivity(User user, String deadline, String description, int energyCost)
public Activity completeActivity(User user, int activityId)
public boolean deleteActivity(int activityId)
public List<Activity> getActivitiesForGroupWithDeadline(int groupId, String date)
public List<Activity> getByPeriod(int groupId, String startDate, String endDate)
public List<Activity> getHandledByPeriod(int groupId, String startDate, String endDate)

3.16. GroupService

private GroupRepository groupRepository;
private UserRepository userRepository;
private ActivityRepository activityRepository;

public List<Group> findAll()
public Group findById(int id)
public Group findByCode(String code)
public Group createGroup(int userId)

public Group joinGroup(String code, User user)
public Group kickMember(int groupId, int userId)
public void deleteGroup(int groupId)
private String generateUniqueCode()

3.17. UserService

private UserRepository userRepository;
private ActivityRepository activityRepository;
private GroupRepository groupRepository;
private final BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();

public User findById(int id)
public List<User> findAll()
public void registerUser(String email, String password)
public User findByEmail(String email)
public boolean deleteUser(User user)
private boolean isValidEmail(String email)
private boolean isValidPassword(String password)

3.18. PsycheAssistantApiApplication

```
public static void main(String[] args) {
        SpringApplication.run(PsycheAssistantApiApplication.class, args);
}
```