# Overall description

The application is a comprehensive tool designed to manage, validate, and secure various operations. It primarily starts by initializing the application and the servlet. The application is fortified with a robust security setup, managing the security context, and authenticated user, along with handling exceptions and errors. It also validates and generates different types of tokens for security purposes.  
  
The application also takes care of email services, from validating the domain of the email address to defining the structure for sending emails and implementing those services. It's equipped with a system to manage roles within the application, creating new roles, and handling role-related operations.  
  
The system excels in managing sites, allowing creation, mapping, and request handling related to sites. It's designed to store and handle practice participation data and manage participation in keyspace services. The application also handles the journey a user takes within the system, creating new journeys, and managing the response data from a journey.  
  
The application handles details of a floor in a site, managing operations related to floors, and initializing the floor database. It defines the operations that can be performed on floors and manages floor services. It also manages exceptions in the application and handles the registration exceptions.  
  
The application is equipped with a feature to manage clues in a game, and it manages user authentication and registration, including handling of email verification and password reset functionality. Lastly, the application takes care of user-related operations, from defining the user entity in the system, mapping user data, to creating a new application user and handling all the services related to the user.

# General

**Nom du fichier:** GamificationApplication

**Langage:** Java

**Objectif du script:** This is the main application file for a gamification system. It starts up the application when run.

# Explanation

This is a Java application that uses the Spring Boot framework. The main function is the entry point of the application, which starts the application and sets it up for use.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| main | The main function that starts the application. | args (String[]) : Command line arguments that can be passed when starting the application. | SpringApplication.run |

# General

**Nom du fichier:** ServletInitializer.java

**Langage:** Java

**Objectif du script:** This script helps to initialize the web application with necessary settings for the Gamification Application.

# Explanation

This script is part of the startup procedure for a web-based application called 'Gamification Application'. It sets up the application with necessary configurations to ensure it runs smoothly.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| configure | This function configures the application with necessary settings for the Gamification Application. | application (SpringApplicationBuilder) : The application builder instance that needs to be configured. | SpringApplicationBuilder.sources |

# General

**Nom du fichier:** DomainEmailValidator.java

**Langage:** Java

**Objectif du script:** This program checks if the domain of an email is valid. It compares the domain of an email with a list of allowed domains. If the domain is in the allowed list or the email is null, it is considered valid.

# Explanation

The script is a custom validator used to verify if the domain of an email address is valid based on a list of allowed domains. If the email domain matches any of the allowed domains or the email address is null, the validation is successful.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| initialize | This function initializes the validator. It doesn't perform any action in this case. | constraintAnnotation (ValidEmailDomain) : The annotation instance. |  |
| isValid | This function checks if the email domain is valid. It returns true if the domain is in the allowed list or the email is null. | email (String) : The email address to be validated. context (ConstraintValidatorContext) : The validation context. |  |

# General

**Nom du fichier:** SpecialCharacterValidator

**Langage:** Java

**Objectif du script:** This code is a validation tool that checks if a given text (such as a password) contains at least one special character. This helps to enforce password complexity rules.

# Explanation

The SpecialCharacterValidator is a Java class that ensures a given string contains at least one special character. It is particularly used to validate password complexity. The validation process is performed by matching the string against a set of defined special characters.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| validate | Checks if the provided string contains any special character. | password (String) : The string to be validated. |  |
| initialize | Initializes the validation process, if needed. | constraintAnnotation (HasSpecialCharacter) : The annotation to be validated. |  |
| isValid | Validates if the provided password contains any special character. | password (String) : The password to be validated. context (ConstraintValidatorContext) : The validation context. | SpecialCharacterValidator.validate |

# General

**Nom du fichier:** HasSpecialCharacter.java

**Langage:** Java

**Objectif du script:** This script ensures that a password includes at least one special character.

# Explanation

This script is a security measure. It checks if a password meets a certain criteria, in this case, if it contains at least one special character. If the password does not meet this criteria, a default error message is displayed.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| message | Returns a default error message when the password doesn't contain a special character. |  |  |
| groups | Returns an array of class groups. In this case, it is empty. |  |  |
| payload | Returns an array of class payloads. In this case, it is empty. |  |  |

# General

**Nom du fichier:** ValidEmailDomain.java

**Langage:** Java

**Objectif du script:** This script is used to validate the domain of an email address. It checks if the domain of the email is valid or not.

# Explanation

The script contains a custom annotation named 'ValidEmailDomain' which is used to validate the domain of an email address. It ensures that the email addresses have a valid domain. If the domain is found to be invalid, it returns a default error message 'Invalid email domain.'

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| message | This function returns the default error message when the validation fails. |  |  |
| groups | This function returns the validation groups for this constraint. It controls when this validation should be applied. |  |  |
| payload | This function returns the payload types for this constraint. It allows for additional metadata to be associated with the constraint. |  |  |

# General

**Nom du fichier:** SecurityUtils

**Langage:** Java

**Objectif du script:** This script helps to manage user authentication in the application. It provides functions to get the authenticated user, set the authentication, clear the authentication, check if a user is authenticated, and retrieve user's claims.

# Explanation

This script is a utility tool for the application's security context. It provides functions that allow the system to handle user authentication. It can fetch the authenticated user, set or clear the authentication, check if a user is authenticated, and retrieve user's claims or specific claim.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| authentication | Fetches the current authentication from the security context. |  |  |
| getAuthenticatedUser | Fetches the currently authenticated user, if available. |  | SecurityUser |
| setAuthentication | Sets the authentication in the security context. | authentication (Authentication) : The authentication object to set. |  |
| clearAuthentication | Clears the authentication from the security context. |  |  |
| isAuthenticated | Checks if a user is authenticated. |  |  |
| getClaims | Fetches the claims from the current authentication. |  | Claims |
| getClaim | Fetches a specific claim from the current authentication's claims. | claim (String) : The name of the claim. type (Class<T>) : The type of the claim. | Claims |

# General

**Nom du fichier:** ThrowableUtil

**Langage:** Java

**Objectif du script:** This is a utility script that helps in handling errors. It has a function that converts the details of an error into a text format.

# Explanation

The script has a single function 'getStackTrace' which takes an error (Throwable) and returns its details as a string. This can be useful for logging or displaying error information in a readable format.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getStackTrace | It takes an error and returns its details in a text format. | throwable (Throwable) : The error object to be converted into string format. |  |

# General

**Nom du fichier:** TokenUtils

**Langage:** Java

**Objectif du script:** This script is a utility for generating and managing non-authentication tokens. It can also check if a token has expired.

# Explanation

The script provides a set of functions for managing tokens. These tokens are not used for user authentication, but for other purposes. It can generate new tokens, check if a token has expired, and hash a byte array using SHA-256 algorithm.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| isTokenExpired | Checks if a token has expired based on a specified expiration value and time unit. | date (Date) : The date when the token was generated. expirationValue (int) : The expiration value. timeUnit (int) : The time unit for expiration. |  |
| sha256HashBytes | Hashes a byte array using the SHA-256 algorithm and returns the hashed value as a Base64-encoded string. | inputBytes (byte[]) : The input bytes to be hashed. |  |
| generateToken | Generates a new token using a cryptographically secure random number generator and SHA-256 hashing. |  | TokenUtils.sha256HashBytes() |

# General

**Nom du fichier:** EmailService.java

**Langage:** Java

**Objectif du script:** This is a blueprint for a service that sends emails. It defines a method to send an email to a specified recipient with a specific subject and body.

# Explanation

This file is a blueprint for sending emails. It's like a recipe that other parts of the program can use to send emails. It details what information is needed (email address, subject, and body of the email) and how to use this information to send an email.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| sendEmail | A method to send an email to a specified recipient with a specific subject and body. | to (String) : The email address of the recipient. subject (String) : The subject of the email. body (String) : The body/content of the email. |  |

# General

**Nom du fichier:** EmailServiceImpl.java

**Langage:** Java

**Objectif du script:** This script is responsible for sending emails. It takes the recipient's address, the subject, and the body of the email as inputs, and then sends the email.

# Explanation

This script is part of an email service in a larger system. It's task is to send emails. It takes three pieces of information: the email address to send the email to, the subject of the email, and the content of the email. Then it attempts to send the email and logs whether the email was sent successfully or not.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| sendEmail | This function sends an email to a specified recipient with a given subject and body. | to (String) : The email address of the recipient. subject (String) : The subject of the email. body (String) : The body of the email. | JavaMailSender.send(), LoggerFactory.getLogger(), Logger.info(), Logger.error() |

# General

**Nom du fichier:** Site.java

**Langage:** Java

**Objectif du script:** This is a code for a Site object, which represents a physical location in the gamification system. It contains details such as the site name, city, region, description, image, and associated journeys, floors, and users.

# Explanation

The Site code defines a location in a gamification system. Each site has a unique ID, name, city, region, description, and image. The site is also linked to various journeys, floors, and users who are associated with this location. All these details are stored in a database.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** SiteController

**Langage:** Java

**Objectif du script:** This script is used for managing sites. It allows the users to perform operations such as creating, updating, deleting, and fetching sites.

# Explanation

This script is a part of the Site Management API. It provides various functionalities to manage sites, including creating a new site, updating an existing site, deleting a site, fetching all sites, and fetching a site by its ID. It also provides the ability to fetch sites based on the city or region, and search sites with a specific term.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getAllSites | Fetches all the sites. |  | SiteServiceImpl.getAll |
| getSiteById | Fetches a site by its ID. | id (Long) : The ID of the site. | SiteServiceImpl.getById |
| createSite | Creates a new site. | siteCreationDTO (SiteCreationDTO) : Data Transfer Object for site creation. | SiteServiceImpl.create |
| updateSiteById | Updates a site by its ID. | id (Long) : The ID of the site. siteCreationDTO (SiteCreationDTO) : Data Transfer Object for site creation. | SiteServiceImpl.updateById |
| deleteSiteById | Deletes a site by its ID. | id (Long) : The ID of the site. | SiteServiceImpl.delete |
| getAllSitesSortedByName | Fetches all the sites sorted by name. |  | SiteServiceImpl.getAllSortedByName |
| getSitesByCity | Fetches sites by city. | city (String) : The city of the sites. page (int) : The page number. size (int) : The number of sites per page. | SiteServiceImpl.findAllByCity |
| getSitesByRegion | Fetches sites by region. | region (String) : The region of the sites. page (int) : The page number. size (int) : The number of sites per page. | SiteServiceImpl.findAllByRegion |
| searchSites | Searches sites. | searchTerm (String) : The search term. page (int) : The page number. size (int) : The number of sites per page. | SiteServiceImpl.search |

# General

**Nom du fichier:** SiteRepository

**Langage:** Java

**Objectif du script:** This script helps to fetch the sites based on city, region, or a search term.

# Explanation

The script is a part of a larger system used to manage and fetch data related to different sites. It allows users to find sites based on the city or region. It also provides a search functionality to find sites using a specific term.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findAllByCity | Fetches all the sites based on the provided city. | city (String) : Name of the city. pageable (Pageable) : Pagination information. |  |
| findAllByRegion | Fetches all the sites based on the provided region. | region (String) : Name of the region. pageable (Pageable) : Pagination information. |  |
| search | Searches for sites based on a search term. | searchTerm (String) : The term to search for. pageable (Pageable) : Pagination information. |  |

# General

**Nom du fichier:** SiteService.java

**Langage:** Java

**Objectif du script:** This code is a service for managing sites in a system. It provides functionalities like getting all sites, getting a site by its ID, creating, updating, and deleting a site, and searching for sites by city, region or a search term.

# Explanation

The 'SiteService' interface provides methods for managing sites. These include retrieving all sites, retrieving a site by its ID, creating a new site, deleting a site, updating a site's details, retrieving all sites sorted by name, finding all sites by city or region, and searching for sites based on a search term.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getAll | Retrieve all sites. |  |  |
| getById | Retrieve a site by its ID. | id (Long) : The ID of the site. |  |
| create | Create a new site. | siteCreationDTO (SiteCreationDTO) : Data transfer object containing the details of the site to be created. |  |
| delete | Delete a site by its ID. | id (Long) : The ID of the site to delete. |  |
| updateById | Update a site's details by its ID. | id (Long) : The ID of the site to update. siteCreationDTO (SiteCreationDTO) : Data transfer object containing the new details of the site. |  |
| getAllSortedByName | Retrieve all sites sorted by name. |  |  |
| findAllByCity | Find all sites by city. | city (String) : The city to search for. pageable (Pageable) : Pagination information. |  |
| findAllByRegion | Find all sites by region. | region (String) : The region to search for. pageable (Pageable) : Pagination information. |  |
| search | Search for sites based on a search term. | searchTerm (String) : The term to search for. pageable (Pageable) : Pagination information. |  |

# General

**Nom du fichier:** SiteServiceImpl.java

**Langage:** Java

**Objectif du script:** This script manages the operations related to the 'Site' like retrieving all sites, getting a site by id, creating a new site, deleting a site, updating a site, getting sites sorted by name, getting sites by city or region, and searching for sites.

# Explanation

This script is responsible for managing various operations related to the 'Site'. It includes functions to fetch all sites, get a specific site by its ID, create a new site, delete a site, update the details of a site, get all sites sorted by their name, fetch sites by city or region, and search for sites.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getAll | Fetches all the sites. |  | siteRepository.findAll(), siteMapper.apply() |
| getById | Fetches a site by its ID. | id (Long) : ID of the site. | siteRepository.findById(), siteMapper.apply() |
| create | Creates a new site. | siteCreationDTO (SiteCreationDTO) : Data Transfer Object for site creation. | siteMapper.creationDtoToEntity(), siteRepository.save(), siteMapper.apply() |
| delete | Deletes a site by its ID. | id (Long) : ID of the site to be deleted. | siteRepository.existsById(), siteRepository.deleteById() |
| updateById | Updates a site by its ID. | id (Long) : ID of the site to be updated. siteCreationDTO (SiteCreationDTO) : Data Transfer Object for site update. | siteRepository.findById(), siteRepository.save(), siteMapper.apply() |
| getAllSortedByName | Fetches all the sites sorted by their name. |  | siteRepository.findAll(), siteMapper.apply() |
| findAllByCity | Fetches all the sites by city. | city (String) : City of the sites. pageable (Pageable) : Pagination information. | siteRepository.findAllByCity(), siteMapper.apply() |
| findAllByRegion | Fetches all the sites by region. | region (String) : Region of the sites. pageable (Pageable) : Pagination information. | siteRepository.findAllByRegion(), siteMapper.apply() |
| search | Searches for sites. | searchTerm (String) : Term to search for. pageable (Pageable) : Pagination information. | siteRepository.search(), siteMapper.apply() |

# General

**Nom du fichier:** SiteCreationDTO.java

**Langage:** Java

**Objectif du script:** This code is used to create a new site with details such as name, city, region, description, site image data, and image mime. It ensures that all the necessary details are provided.

# Explanation

This code represents a blueprint for creating a site. It ensures that the name, city, region, and description are provided. Additionally, it also has the ability to handle site image data and image mime. If any of these details are not provided, it will give an error message.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** SiteMapper

**Langage:** Java

**Objectif du script:** This script is responsible for converting data between two object types, Site and SiteRequestDTO. It also has a function to convert SiteCreationDTO to Site.

# Explanation

The script 'SiteMapper' in Java is used to transform data between different formats. It has two main functions: 'apply' which converts a Site object to a SiteRequestDTO object, and 'creationDtoToEntity' which converts a SiteCreationDTO object to a Site object. These transformations are necessary for data exchange between different parts of the application.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| apply | Converts a Site object to a SiteRequestDTO object. | site (Site) : The Site object to be converted to SiteRequestDTO. |  |
| creationDtoToEntity | Converts a SiteCreationDTO object to a Site object. | siteCreationDto (SiteCreationDTO) : The SiteCreationDTO object to be converted to Site. |  |

# General

**Nom du fichier:** SiteRequestDTO.java

**Langage:** Java

**Objectif du script:** This script defines a data transfer object (DTO) for site requests. It extends the SiteCreationDTO, which means it inherits all its properties and methods, and adds an additional 'id' property. This object is used to transfer data between processes.

# Explanation

The script is a Java class named 'SiteRequestDTO'. It is a special kind of object used for transferring data across different parts of a software system. This particular DTO is used for site requests and it adds an 'id' field to the data inherited from another DTO, 'SiteCreationDTO'.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** JwtService.java

**Langage:** Java

**Objectif du script:** This is a security service used for managing user identification tokens. It can extract a username from a token, and generate a new token for a user.

# Explanation

The JwtService is a part of the security system of the application. It handles user identification tokens, known as JWTs. It can both create new tokens and extract information from existing ones.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| extractUsername | Extracts the username from the provided token. | token (String) : The token from which the username is to be extracted. |  |
| generateToken | Generates a new token for the provided username. | userName (String) : The username for which a new token is to be generated. |  |

# General

**Nom du fichier:** JwtServiceImpl.java

**Langage:** Java

**Objectif du script:** This code is responsible for managing security tokens for users. It can create new tokens and extract user information from existing tokens.

# Explanation

This script is a part of the system's security mechanism. It creates secure tokens for users when they log in, and can also read these tokens to get user information. This is done in a secure way to ensure user information is safe.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| extractUsername | This function reads a security token, and returns the username associated with that token. | token (String) : The security token to be read. | io.jsonwebtoken.Jwts.parserBuilder(), io.jsonwebtoken.io.Decoders.BASE64.decode() |
| generateToken | This function creates a new security token for a given username. | userName (String) : The username for which a new security token will be created. | io.jsonwebtoken.Jwts.builder(), java.util.Date(), io.jsonwebtoken.security.Keys.hmacShaKeyFor() |
| getSigningKey | This function generates a key to be used in creating or reading security tokens. |  | io.jsonwebtoken.io.Decoders.BASE64.decode(), io.jsonwebtoken.security.Keys.hmacShaKeyFor() |

# General

**Nom du fichier:** JwtAuthenticationFilter.java

**Langage:** Java

**Objectif du script:** This script is used to authenticate users of the application by checking their provided credentials against stored user information.

# Explanation

The script uses a method called 'JSON Web Token' (JWT) to authenticate users. When a user logs in, they are given a token. Every time they make a request, this token is checked to ensure it's valid and belongs to them. This way, we can be sure that the user is who they say they are.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| doFilterInternal | This function checks the user's token every time they make a request. | request (HttpServletRequest) : The request made by the user. response (HttpServletResponse) : The response that will be sent back to the user. filterChain (FilterChain) : The chain of filters that the request goes through. | com.capgemini.gamification.security.service.JwtService.extractUsername(), com.capgemini.gamification.utils.SecurityUtils.isAuthenticated(), com.capgemini.gamification.utils.SecurityUtils.setAuthentication() |
| buildAuth | This function builds the authentication token for the user. | request (HttpServletRequest) : The request made by the user. | com.capgemini.gamification.security.service.JwtService.extractUsername(), com.capgemini.gamification.utils.SecurityUtils.isAuthenticated() |
| getToken | This function extracts the token from the user's request. | request (HttpServletRequest) : The request made by the user. |  |
| buildAuthenticationToken | This function builds the authentication token for the user. | user (UserDetails) : The details of the user. request (HttpServletRequest) : The request made by the user. |  |

# General

**Nom du fichier:** SecurityConfig

**Langage:** Java

**Objectif du script:** This script sets up security measures for a web application, including user authentication and access permissions.

# Explanation

The script ensures that only authenticated users can access certain parts of the application. It also sets up Cross-Origin Resource Sharing (CORS), which allows the application to safely exchange data with different domains.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| securityFilterChain | Sets up the security rules for the application. | http (HttpSecurity) : Object that allows configuration of web-based security for specific http requests. accessDeniedHandler (AccessDeniedHandler) : Handles cases where access is denied. authenticationEntryPoint (AuthenticationEntryPoint) : Starts the authentication process. |  |
| corsConfigurationSource | Sets up Cross-Origin Resource Sharing (CORS) for the application. |  |  |

# General

**Nom du fichier:** SecurityUser.java

**Langage:** Java

**Objectif du script:** This code is used to manage the security settings of a user in an application.

# Explanation

The code helps in setting up the security for the user in an application. It defines the user's attributes, their roles and if they have the necessary permissions to access certain aspects of the application.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| SecurityUser | Constructor function that sets the user's details. | userId, firstName, lastName, avatar, email, password, createdAt, updatedAt, enabled, role, site, practice, participates (Long, String, String, Integer, String, String, Date, Date, boolean, Role, Site, Practice, List<Participate>) : These arguments represent the various attributes of a user. |  |
| map | Function that maps the attributes of an AppUser object to a SecurityUser object. | appUser (AppUser) : An instance of the AppUser class. | AppUser.getId(), AppUser.getFirstName(), AppUser.getLastName(), AppUser.getAvatar(), AppUser.getEmail(), AppUser.getPassword(), AppUser.getCreatedAt(), AppUser.getUpdatedAt(), AppUser.isEnabled(), AppUser.getRole(), AppUser.getSite(), AppUser.getPractice(), AppUser.getParticipates() |
| getAuthorities | Function that returns the user's role as a list of granted authorities. |  | Role.getName() |
| getUsername | Function that returns the user's email as the username. |  | AppUser.getEmail() |
| isAccountNonExpired | Function that returns true, indicating that the user's account is not expired. |  |  |
| isAccountNonLocked | Function that returns true, indicating that the user's account is not locked. |  |  |
| isCredentialsNonExpired | Function that returns true, indicating that the user's credentials are not expired. |  |  |

# General

**Nom du fichier:** Role.java

**Langage:** Java

**Objectif du script:** This script represents different roles that users can have within the application.

# Explanation

The script defines a 'Role' within a system, such as 'admin' or 'user'. Each role has an ID and a name, and is associated with a list of users who have that role. This information is stored in a database table called 'role'.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** RoleController.java

**Langage:** Java

**Objectif du script:** This code is responsible for managing user roles in an application. It allows for creating, retrieving, updating and deleting roles.

# Explanation

The code is a part of a larger system that handles user roles. It allows administrators to create, update, and delete roles, and both users and administrators to view roles. All these operations are performed through a service called 'RoleService'.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getRoleById | Fetches a role based on its ID. | id (Long) : The unique identifier of the role. | RoleService.findRoleById() |
| getAll | Fetches all roles. |  | RoleService.findAll() |
| createRole | Creates a new role. | newRoleDto (RoleCreationDTO) : Data Transfer Object containing information for creating a new role. | RoleService.saveRole() |
| updateRole | Updates an existing role. | updatedRoleDto (RoleCreationDTO) : Data Transfer Object containing updated information for a role. id (Long) : The unique identifier of the role to be updated. | RoleService.updateRole() |
| deleteRole | Deletes a role based on its ID. | id (Long) : The unique identifier of the role to be deleted. | RoleService.deleteRoleById() |

# General

**Nom du fichier:** RoleMapper.java

**Langage:** Java

**Objectif du script:** This script is used to convert data between different formats that are RoleCreationDTO and RoleResponseDTO.

# Explanation

The script 'RoleMapper.java' is a component of a larger system, responsible for converting data between two different formats. It takes in RoleCreationDTO (Data Transfer Object) and converts it into an entity Role. It also converts a Role entity to a RoleResponseDTO.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| creationDtoToEntity | Transforms a RoleCreationDTO object into a Role entity. | roleCreationDTO (RoleCreationDTO) : An object that carries data between processes. |  |
| entityToRequestDto | Transforms a Role entity into a RoleResponseDTO object. | role (Role) : An entity that represents a Role in the system. |  |

# General

**Nom du fichier:** RoleRepository.java

**Langage:** Java

**Objectif du script:** This is a script that helps us interact with the roles data in our database. It includes a function to find a role by its name.

# Explanation

This script is a part of our system that helps us manage and access role information. It provides a way to find a specific role by its name. This could be used, for example, to check what permissions a certain role has.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findByName | A function that searches for a role by its name. | name (String) : The name of the role we are searching for. | JpaRepository.findAll() |

# General

**Nom du fichier:** RoleService.java

**Langage:** Java

**Objectif du script:** This script manages the different actions that can be performed on roles, such as finding, creating, updating, and deleting roles.

# Explanation

The RoleService script is a key component of our application. It allows us to perform various operations on roles, ensuring that we can add, update, or remove roles as needed. It also allows us to fetch information about a specific role or all roles.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findRoleById | Finds and returns a role based on the provided ID. | id (Long) : The unique identifier of the role. |  |
| findAll | Returns a list of all roles. |  |  |
| saveRole | Creates a new role. | role (RoleCreationDTO) : The details of the role to be created. |  |
| deleteRoleById | Deletes a role based on the provided ID. | id (Long) : The unique identifier of the role. |  |
| updateRole | Updates the details of a role. | updatedRoleDto (RoleCreationDTO) : The updated details of the role. id (Long) : The unique identifier of the role. |  |

# General

**Nom du fichier:** RoleServiceImpl.java

**Langage:** Java

**Objectif du script:** This code is responsible for managing roles in a system. It allows for creating, updating, deleting, and fetching roles.

# Explanation

The script manages user roles in an application. It provides functionalities such as finding a role by its ID, retrieving all roles, saving a new role, updating an existing role, and deleting a role by its ID.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findRoleById | Fetches a role by its ID. | id (Long) : The unique identifier of the role. | RoleRepository.findById(), RoleMapper.entityToRequestDto() |
| findAll | Fetches all roles. |  | RoleRepository.findAll(), RoleMapper.entityToRequestDto() |
| saveRole | Saves a new role. | roleDto (RoleCreationDTO) : Data transfer object containing the details of the role to be saved. | RoleRepository.findByName(), RoleMapper.creationDtoToEntity(), RoleRepository.save() |
| updateRole | Updates an existing role. | updatedRoleDto (RoleCreationDTO) : Data transfer object containing the updated details of the role. id (Long) : The unique identifier of the role to be updated. | RoleRepository.findById(), RoleRepository.save(), RoleMapper.entityToRequestDto() |
| deleteRoleById | Deletes a role by its ID. | id (Long) : The unique identifier of the role to be deleted. | RoleRepository.deleteById() |

# General

**Nom du fichier:** RoleCreationDTO.java

**Langage:** Java

**Objectif du script:** This code is used to create a new role in our system. It ensures that a role name is provided when a new role is created.

# Explanation

The RoleCreationDTO is a simple data holder for creating a new role. It has a single field 'name' which is mandatory. If no name is provided, an error message 'role name is required' will be displayed.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** RoleResponseDTO.java

**Langage:** Java

**Objectif du script:** This file defines a structure to hold and manipulate data related to roles in a system.

# Explanation

The RoleResponseDTO class is a simple data holder. It has two properties: id and name, which represent the unique identifier and name of a role respectively. This class is used to transfer role data within the system.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** OpenApiProperties

**Langage:** Java

**Objectif du script:** This script manages the properties of the Open API Documentation, such as title, version, contact information, license information, and server details.

# Explanation

The script is a configuration file for the Open API Documentation of an application. It sets and retrieves information like the title, version, contact details, license info, and server details. The information is stored in a structured manner for easy retrieval and modification.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getTitle | Retrieves the title of the Open API Documentation. |  |  |
| getVersion | Retrieves the version of the Open API Documentation. |  |  |
| getDescription | Retrieves the description of the Open API Documentation. |  |  |
| getContact | Retrieves the contact information for the Open API Documentation. |  |  |
| getLicence | Retrieves the license information for the Open API Documentation. |  |  |
| getServers | Retrieves the server information for the Open API Documentation. |  |  |

# General

**Nom du fichier:** Practice.java

**Langage:** Java

**Objectif du script:** This code is about the 'Practice' entity which represents a practice in a gamification system. It has properties like ID, name, and lists of users and journeys associated with it.

# Explanation

The 'Practice' code defines a component of a gamification system. It is like a category or group where users and journeys are associated. Each practice has a unique ID and a name. The users and journeys related to a practice are stored in lists.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** PracticeController

**Langage:** Java

**Objectif du script:** This script is responsible for managing 'practices'. It allows users to create, update, delete, and retrieve practices.

# Explanation

The script is a part of a larger system that manages 'practices'. It provides functionalities such as creating a new practice, updating an existing practice, deleting a practice, and retrieving either a specific practice or all practices. These functionalities are only accessible to certain users based on their roles.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getPracticeById | Retrieves a specific practice by its ID. | id (Long) : The ID of the practice to retrieve. | practiceService.findPracticeById() |
| getAllPractices | Retrieves all practices. |  | practiceService.findAll() |
| createPractice | Creates a new practice. | newPracticeDto (PracticeCreationDTO) : The details of the practice to create. | practiceService.createPractice() |
| updatePractice | Updates an existing practice. | updatedPracticeDto (PracticeCreationDTO) : The updated details of the practice. id (Long) : The ID of the practice to update. | practiceService.updatePractice() |
| deletePractice | Deletes a practice. | id (Long) : The ID of the practice to delete. | practiceService.deletePracticeById() |

# General

**Nom du fichier:** PracticeMapper.java

**Langage:** Java

**Objectif du script:** This script is responsible for converting data between different formats that are used in the application. It converts data from the format used for creating practices to the format used in the application's database and vice versa.

# Explanation

The PracticeMapper script is a key component in the data handling of the application. It ensures that data related to 'practices' is correctly converted between the format used when creating new practices and the format used in the application's database. This conversion is necessary for the smooth functioning of the application.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| creationDtoToEntity | Converts the data from the creation format to the database format. | practiceCreationDTO (PracticeCreationDTO) : The data of the practice in the creation format. |  |
| entityToResponseDto | Converts the data from the database format to the response format. | practice (Practice) : The data of the practice in the database format. |  |

# General

**Nom du fichier:** PracticeRepository.java

**Langage:** Java

**Objectif du script:** This file is used to interact with the 'Practice' data in our database.

# Explanation

The 'PracticeRepository' file is a part of our system that communicates with the database. It specifically deals with the 'Practice' data, allowing us to retrieve, update, and delete records as necessary.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** PracticeService

**Langage:** Java

**Objectif du script:** This is a blueprint for a service that manages 'Practice' records. It allows for creating, reading, updating, and deleting these records.

# Explanation

This piece of code outlines a service for 'Practice' records. It provides functionalities to view a 'Practice' record by its ID, view all 'Practice' records, create a new 'Practice' record, delete a 'Practice' record by its ID, and update an existing 'Practice' record.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findPracticeById | Fetches a 'Practice' record using its ID | id (Long) : ID of the 'Practice' record |  |
| findAll | Fetches all 'Practice' records |  |  |
| createPractice | Creates a new 'Practice' record | practiceCreationDTO (PracticeCreationDTO) : Data needed to create a new 'Practice' record |  |
| deletePracticeById | Deletes a 'Practice' record using its ID | id (Long) : ID of the 'Practice' record |  |
| updatePractice | Updates an existing 'Practice' record | updatedPracticeDto (PracticeCreationDTO) : Data needed to update the 'Practice' record id (Long) : ID of the 'Practice' record |  |

# General

**Nom du fichier:** PracticeServiceImpl.java

**Langage:** Java

**Objectif du script:** This script is used to manage 'Practices'. It allows to create, retrieve, update and delete Practices.

# Explanation

This script is part of the service layer of an application. It provides the logic to operate on the data sent to and from the Practice Repository and acts as an intermediary between the repository and the user interface.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findPracticeById | Finds a specific Practice using its ID. | id (Long) : The ID of the Practice. | practiceRepository.findById(), practiceMapper.entityToResponseDto() |
| findAll | Finds all Practices. |  | practiceRepository.findAll(), practiceMapper.entityToResponseDto() |
| createPractice | Creates a new Practice. | practiceCreationDTO (PracticeCreationDTO) : Data Transfer Object containing the information needed to create a new Practice. | practiceRepository.save(), practiceMapper.creationDtoToEntity(), practiceMapper.entityToResponseDto() |
| deletePracticeById | Deletes a specific Practice using its ID. | id (Long) : The ID of the Practice. | practiceRepository.existsById(), practiceRepository.deleteById() |
| updatePractice | Updates a specific Practice using its ID and the updated data. | updatedPracticeDto (PracticeCreationDTO) : Data Transfer Object containing the updated information of the Practice. id (Long) : The ID of the Practice. | practiceRepository.findById(), practiceRepository.save(), practiceMapper.entityToResponseDto() |

# General

**Nom du fichier:** PracticeCreationDTO.java

**Langage:** Java

**Objectif du script:** This script is used to create new 'Practice' objects. It ensures that the 'name' field is always provided when creating a new 'Practice'.

# Explanation

This script is a part of the 'gamification' package and it's used to create a new Practice object. It has a 'name' field which cannot be blank. This ensures that every Practice object created has a name.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** PracticeResponseDTO.java

**Langage:** Java

**Objectif du script:** This is a data transfer object that holds information about a 'Practice' object. It contains two pieces of information - an id and a name.

# Explanation

The script defines a structure to hold specific data related to a 'Practice'. It has two fields: 'id' and 'name'. The 'id' is a unique identifier for each 'Practice' and 'name' is the name of the 'Practice'.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** Participate.java

**Langage:** Java

**Objectif du script:** This script manages the participation of users in different journeys. It also handles the relationship between the user, the journey they are participating in, and the keyspace related to that participation.

# Explanation

The script 'Participate.java' is part of a gamification system. It creates a connection between users (AppUser), journeys (Journey), and keyspaces (ParticipateKeyspace) they are participating in. It doesn't perform any actions or calculations, but provides a structure for storing and retrieving participation data.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** ParticipateController.java

**Langage:** Java

**Objectif du script:** This script manages the participation of users in a game. It allows to create, update, delete, and retrieve information about the participants and their associated game keys.

# Explanation

This is a Java script for a web application. It manages user participation in a game. It includes functions to create, update, delete, and retrieve participant information. It also manages game keys associated with each participant.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getParticipateById | Retrieves a participant's details using their ID. | id (Long) : The ID of the participant. | participateService.findParticipateById() |
| getAllParticipates | Retrieves details of all participants. |  | participateService.findAll() |
| createParticipate | Creates a new participant. | participateCreationDTO (ParticipateCreationDTO) : The details of the participant to be created. | participateService.saveParticipate() |
| updateParticipate | Updates the details of a participant. | updatedParticipate (ParticipateCreationDTO) : The updated details of the participant. id (Long) : The ID of the participant to be updated. | participateService.updateParticipate() |
| deleteParticipateById | Deletes a participant using their ID. | id (Long) : The ID of the participant to be deleted. | participateService.delete() |
| findAllByJourney | Retrieves all participants associated with a specific journey ID. | id (Long) : The ID of the journey. | participateService.findAllByJourney() |
| findAllByUser | Retrieves all participants associated with a specific user ID. | id (Long) : The ID of the user. | participateService.findAllByUser() |
| getParticipateKeyspaceById | Retrieves the game key associated with a participant using the participant's ID. | id (Long) : The ID of the participant. | participateKeyspaceService.findById() |
| updateParticipateKeyspaceById | Updates the game key associated with a participant. | id (Long) : The ID of the participant. body (ParticipateKeyspaceDTO) : The updated game key details. | participateKeyspaceService.updateClueConsumption() |

# General

**Nom du fichier:** ParticipateKeyspace.java

**Langage:** Java

**Objectif du script:** This script is responsible for creating the 'participate\_keyspace' table in the database. It contains information about participation in a certain activity, and includes details such as the status of participation, the participant, the keyspace, and the number of clues consumed.

# Explanation

This script defines a data model for the 'participate\_keyspace' table. It includes the unique ID of each entry, the status of participation (true/false), the participant's details, the keyspace details, and the number of clues consumed during participation.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** ParticipateKeyspaceMapper

**Langage:** Java

**Objectif du script:** This file is a component in the system that helps convert a particular type of data (ParticipateKeyspace) into a format (ParticipateKeyspaceDTO) that can be easily used or displayed in other parts of the system.

# Explanation

The ParticipateKeyspaceMapper is a tool that takes in data related to user participation, and converts it into a more friendly format. This includes information like the ID, any clues they've used, and their current status. This makes it easier to manage and display this data elsewhere in the system.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| entityToResponseDto | This function takes a ParticipateKeyspace object and transforms it into a ParticipateKeyspaceDTO object. | participateKeyspace (ParticipateKeyspace) : An object that contains data about the user's participation. |  |

# General

**Nom du fichier:** ParticipateKeyspaceRepository.java

**Langage:** Java

**Objectif du script:** This code is used to interact with the database where the information about participation in keyspaces is stored. It allows to find a specific participation record by its ID and the keyspace ID.

# Explanation

This script is a part of our system that manages interactions with the database. It specifically handles the data related to participation in keyspaces. It allows us to find a specific participation by using the participant's ID and the keyspace ID.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findByParticipateIdAndKeyspaceId() | This function is used to find a specific participation record in the database by using the participant's ID and the keyspace ID. | participateId (Long) : The ID of the participant. keyspaceId (Long) : The ID of the keyspace. |  |

# General

**Nom du fichier:** ParticipateMapper.java

**Langage:** Java

**Objectif du script:** This script is used for converting data between different formats or structures. It converts data related to user participation in a journey from one format to another.

# Explanation

The script provides two main functionalities. First, it converts a Participate entity into a ParticipateResponseDTO, which is a data transfer object used to send data over the network. Second, it converts a ParticipateCreationDTO into a Participate entity, which is used to store data in the database.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| entityToResponseDto | Converts a Participate entity into a ParticipateResponseDTO. | participate (Participate) : The Participate entity to be converted. |  |
| creationDtoToEntity | Converts a ParticipateCreationDTO into a Participate entity. | participateCreationDTO (ParticipateCreationDTO) : The ParticipateCreationDTO to be converted. | com.capgemini.gamification.appuser.AppUser, com.capgemini.gamification.journey.Journey |

# General

**Nom du fichier:** ParticipateRepository

**Langage:** Java

**Objectif du script:** This code is part of a game application. It manages the interactions of the participants with the game, such as joining a journey or checking their progress.

# Explanation

The ParticipateRepository is a part of the backend of a gamification application. It manages the data related to game participants, such as which journey they are on, and their progress. It also prepares a user's journey when they start a new one.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findAllByJourney | Finds all participants in a specific journey. | journeyId (Long) : The unique identifier of the journey. |  |
| findAllByUser | Finds all journeys that a specific user is participating in. | userId (Long) : The unique identifier of the user. |  |
| findByUserIdAndJourneyId | Finds a specific participation instance by user and journey identifiers. | userId (Long) : The unique identifier of the user. journeyId (Long) : The unique identifier of the journey. |  |
| prepareUserJourney | Prepares a new journey for a user, ensuring that they are not already participating in it. | userId (Long) : The unique identifier of the user. journeyId (Long) : The unique identifier of the journey. |  |

# General

**Nom du fichier:** ParticipateKeyspaceService

**Langage:** Java

**Objectif du script:** This code is part of a service that manages the interactions between participants and keyspaces (areas of interest). It provides functionalities such as finding a keyspace by its ID and a participant's ID, finding a keyspace by its ID, and updating the number of clues consumed in a keyspace.

# Explanation

This service is used to handle the operations related to the participation of users in keyspaces. It allows to find a keyspace by its ID and a participant's ID, to find a keyspace by its ID, and to update the number of clues consumed in a keyspace.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findByKeyspaceIdAndParticipateId | Finds a keyspace by its ID and a participant's ID. | keyspaceId (Long) : The ID of the keyspace. participateId (Long) : The ID of the participant. |  |
| findById | Finds a keyspace by its ID. | participateKeyspaceId (Long) : The ID of the keyspace. |  |
| updateClueConsumption | Updates the number of clues consumed in a keyspace. | participateKeyspaceId (Long) : The ID of the keyspace. clueConsumption (Integer) : The number of clues consumed. |  |

# General

**Nom du fichier:** ParticipateKeyspaceServiceImpl.java

**Langage:** Java

**Objectif du script:** This code manages the interactions related to ParticipateKeyspace, such as finding a specific ParticipateKeyspace and updating clue consumption.

# Explanation

This script is part of a gamification system. It provides services to find and update information related to a participant's interaction with a specific area or 'keyspace' in the game. It can find a ParticipateKeyspace by its ID or by both keyspace and participant IDs. It can also update the number of clues consumed in a ParticipateKeyspace.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findByKeyspaceIdAndParticipateId | Finds a specific ParticipateKeyspace using keyspace and participate IDs. | keyspaceId (Long) : The ID of the keyspace. participateId (Long) : The ID of the participant. | ParticipateKeyspaceMapper.entityToResponseDto, ParticipateKeyspaceRepository.findByParticipateIdAndKeyspaceId |
| findById | Finds a specific ParticipateKeyspace using its ID. | participateKeyspaceId (Long) : The ID of the ParticipateKeyspace. | ParticipateKeyspaceMapper.entityToResponseDto, ParticipateKeyspaceRepository.findById |
| updateClueConsumption | Updates the number of clues consumed in a specific ParticipateKeyspace. | participateKeyspaceId (Long) : The ID of the ParticipateKeyspace. clueConsumption (Integer) : The number of clues to be consumed. | ParticipateKeyspaceMapper.entityToResponseDto, ParticipateKeyspaceRepository.findById, ParticipateKeyspaceRepository.save |

# General

**Nom du fichier:** ParticipateService.java

**Langage:** Java

**Objectif du script:** This code is a service interface for managing participation records. It includes functionalities such as finding, saving, updating, and deleting participations, as well as fetching participation lists based on journey or user ID.

# Explanation

This Java interface provides a blueprint for a service dealing with 'Participations'. It outlines key functionalities like retrieving a single participation record or all records, saving new records, updating existing ones, deleting records, and fetching participation lists based on either a journey or user ID.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findParticipateById | Fetches a participation record by its ID. | id (Long) : The ID of the participation. |  |
| findAll | Fetches all participation records. |  |  |
| saveParticipate | Saves a new participation record. | participateCreationDTO (ParticipateCreationDTO) : The new participation record to be saved. |  |
| updateParticipate | Updates an existing participation record. | updatedParticipateDto (ParticipateCreationDTO) : The updated participation record. id (Long) : The ID of the participation to be updated. |  |
| delete | Deletes a participation record. | id (Long) : The ID of the participation to be deleted. |  |
| findAllByJourney | Fetches all participation records for a specific journey. | id (Long) : The ID of the journey. |  |
| findAllByUser | Fetches all participation records for a specific user. | id (Long) : The ID of the user. |  |

# General

**Nom du fichier:** ParticipateServiceImpl.java

**Langage:** Java

**Objectif du script:** This code is responsible for managing participation in a game or event. It can find, create, update, or delete a participation record. It can also list all participations or filter them by user or journey.

# Explanation

This code handles the participation-related tasks in a game or event. It can find a specific participation record, list all records, or filter them based on the user or journey. It also allows the creation, update, or deletion of a participation record.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findParticipateById | Finds a specific participation record by its ID. | id (Long) : The ID of the participation record. | participateRepository.findById(), participateMapper.entityToResponseDto() |
| findAll | Lists all participation records. |  | participateRepository.findAll(), participateMapper.entityToResponseDto() |
| saveParticipate | Creates a new participation record. | participateCreationDTO (ParticipateCreationDTO) : The data for the new participation record. | participateMapper.creationDtoToEntity(), participateRepository.save(), participateMapper.entityToResponseDto() |
| updateParticipate | Updates a specific participation record. | updatedParticipateDto (ParticipateCreationDTO) : The updated data for the participation record. id (Long) : The ID of the participation record to be updated. | participateRepository.findById(), journeyRepository.findById(), participateRepository.save(), participateMapper.entityToResponseDto() |
| delete | Deletes a specific participation record. | id (Long) : The ID of the participation record to be deleted. | participateRepository.deleteById() |
| findAllByJourney | Lists all participation records for a specific journey. | id (Long) : The ID of the journey. | participateRepository.findAllByJourney(), participateMapper.entityToResponseDto() |
| findAllByUser | Lists all participation records for a specific user. | id (Long) : The ID of the user. | participateRepository.findAllByUser(), participateMapper.entityToResponseDto() |

# General

**Nom du fichier:** ParticipateCreationDTO.java

**Langage:** Java

**Objectif du script:** This script is used to create a participation record. It requires a user ID and a journey ID.

# Explanation

This Java script is a Data Transfer Object (DTO) used for creating a participation record. It has two essential fields: User ID and Journey ID. Both fields are mandatory for the creation of a new participation record.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** ParticipateKeyspaceDTO.java

**language:** Java

**Objectif du script:** This script is used to manage and store the key details of the participation in the game. It keeps track of the participant's ID, their current status in the game, and the number of clues they've used.

# Explanation

The script is a data transfer object (DTO) that carries data between processes. In this case, it carries the data related to participation in the game, such as the participant's ID, status and the number of clues consumed.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** ParticipateResponseDTO.java

**Langage:** Java

**Objectif du script:** This file is used to create a data structure that holds information about a user's participation in a journey. It includes details such as the ID of the participation, the user involved, and the journey they are participating in.

# Explanation

The 'ParticipateResponseDTO' is a simple data transfer object that is used to encapsulate the information about a user's participation in a journey. It is used to easily transfer data between different parts of the application or between the application and its users. It includes details such as the ID of the participation, the user involved, and the journey they are participating in.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** Keyspace.java

**Langage:** Java

**Objectif du script:** This script defines a 'Keyspace' object, which is a key part of a gamification system. It includes properties like the space name, help text, a code, coordinates, and associated journeys, clues, and participants.

# Explanation

The Keyspace object is a crucial part of our gamification system. It represents a specific area or level in the game. It has various properties like a unique ID, a name, a help text, a code, and coordinates. It also has relationships with other objects like Floor, Journey, Clue, and ParticipateKeyspace, which are used to define the game's structure.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** KeyspaceController

**Langage:** Java

**Objectif du script:** This code provides a way for administrators to manage 'keyspaces' - these are essentially digital 'spaces' or 'areas' which can contain various clues. It allows administrators to create, update, delete, and validate these keyspaces, as well as retrieve information about them.

# Explanation

This code is a controller for managing keyspaces in a game. It provides endpoints for administrators to create, update, delete, and validate keyspaces. It also allows retrieving information about keyspaces, such as finding all clues in a keyspace, or finding all keyspaces in a particular site or floor.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findKeyspaceById | Finds a keyspace by its ID. | id (Long) : The ID of the keyspace to find. | keyspaceService.findKeyspaceById() |
| findAllClueByKeyspace | Finds all clues in a keyspace. | keyspaceId (Long) : The ID of the keyspace to find clues in. | keyspaceService.getAllCluesByKeyspace() |
| createKeyspace | Creates a new keyspace. | dto (KeyspaceCreationDTO) : Data Transfer Object containing information for creating a keyspace. | keyspaceService.createKeyspace() |
| updateKeyspace | Updates an existing keyspace. | id (Long) : The ID of the keyspace to update. dto (KeyspaceCreationDTO) : Data Transfer Object containing information for updating a keyspace. | keyspaceService.updateKeyspace() |
| removeKeyspace | Deletes a keyspace. | id (Long) : The ID of the keyspace to delete. | keyspaceService.deleteKeyspace() |
| validateKeySpace | Validates a user's keyspace journey. | validateKeyspaceDTO (ValidateKeyspaceDTO) : Data Transfer Object containing information for validating a keyspace journey. | keyspaceService.validateKeySpace() |

# General

**Nom du fichier:** KeyspaceMapper

**Langage:** Java

**Objectif du script:** This file provides functions to convert Keyspace data to different formats and vice versa. It also has functions to hide certain information from the Keyspace data.

# Explanation

This script is part of a larger system that manages 'Keyspace' data. It primarily converts Keyspace data from one format to another, and also has functions to hide certain information from the Keyspace data for security purposes.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| entityToResponseDTO | Converts a Keyspace entity into a response data transfer object (DTO). | keyspace (Keyspace) : A Keyspace entity to be converted. |  |
| entityListToResponseDTO | Converts a list of Keyspace entities into a list of response DTOs. | list (List<Keyspace>) : A list of Keyspace entities to be converted. |  |
| strike | Hides the code and coordinates from a Keyspace response DTO. | dto (KeyspaceResponseDTO) : A Keyspace response DTO to be modified. |  |
| strike | Hides the code and coordinates from a list of Keyspace response DTOs. | list (List<KeyspaceResponseDTO>) : A list of Keyspace response DTOs to be modified. |  |
| creationDTOToEntity | Creates a Keyspace entity from a creation DTO. | dto (KeyspaceCreationDTO) : A Keyspace creation DTO to be converted into a Keyspace entity. floor (Floor) : A Floor entity to be associated with the Keyspace entity. keyspaceId (Long) : The ID to be assigned to the new Keyspace entity. |  |

# General

**Nom du fichier:** KeyspaceRepository

**Langage:** Java

**Objectif du script:** This script manages operations related to 'Keyspace' data. It provides ways to retrieve 'Keyspace' data based on site, floor, journey, and id.

# Explanation

This script is a repository for the 'Keyspace' data in our database. It enables us to fetch 'Keyspace' information based on specific criteria like site, floor, journey, and id. It's essentially a tool for retrieving and managing 'Keyspace' data.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findAllBySite | Fetches all 'Keyspace' data associated with a particular site. | siteId (Long) : The id of the site. |  |
| findAllByFloor | Fetches all 'Keyspace' data associated with a particular floor. | floorId (Long) : The id of the floor. |  |
| findAllByJourney | Fetches all 'Keyspace' data associated with a particular journey. | journeyId (Long) : The id of the journey. |  |
| findById | Fetches 'Keyspace' data by its id. | keyspaceId (Long) : The id of the keyspace. |  |

# General

**Nom du fichier:** KeyspaceService

**Langage:** Java

**Objectif du script:** This script is a service interface for managing and interacting with 'Keyspaces'. It allows for creating, updating, finding, and deleting Keyspaces. Additionally, it provides functionality for working with Clues associated with a Keyspace.

# Explanation

The 'KeyspaceService' script provides various functions to manage 'Keyspaces' and their associated 'Clues'. It allows users to find a Keyspace by its ID, by its site, floor, or journey. It also provides functions to create, update, and delete a Keyspace. Furthermore, it provides functions to retrieve all Clues by a Keyspace and to validate a Keyspace.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findKeyspaceById | Finds a Keyspace by its ID | id (Long) : The ID of the Keyspace |  |
| findAllBySite | Finds all Keyspaces by Site ID | siteId (Long) : The ID of the Site |  |
| createKeyspace | Creates a new Keyspace | dto (KeyspaceCreationDTO) : Data Transfer Object containing Keyspace creation information |  |
| deleteKeyspace | Deletes a Keyspace by its ID | id (Long) : The ID of the Keyspace to be deleted |  |
| getAllCluesByKeyspace | Retrieves all Clues associated with a Keyspace | keyspaceId (Long) : The ID of the Keyspace |  |
| validateKeySpace | Validates a Keyspace | validateKeyspaceDTO (ValidateKeyspaceDTO) : Data Transfer Object containing Keyspace validation information |  |

# General

**Nom du fichier:** KeyspaceServiceImpl.java

**Langage:** Java

**Objectif du script:** This script is used to manage 'keyspaces' in an application. It can find, create, update, delete, and validate keyspaces, as well as retrieve clues associated with a keyspace. It interacts with several other parts of the application, such as sites, floors, journeys, and users.

# Explanation

This script provides functions to manage 'keyspaces'. It can find keyspaces based on different criteria, create new keyspaces, update existing ones, delete keyspaces, and validate a keyspace. It can also retrieve clues associated with a keyspace. It interacts with other parts of the application like sites, floors, journeys, and users.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findKeyspaceById, findAllBySite, findAllByFloor, findAllByJourney, createKeyspace, updateKeyspace, deleteKeyspace, getAllCluesByKeyspace, getClueByKeyspaceAndOrder, validateKeySpace, findKeyspaceByIdOrThrow, findFloorByIdOrThrow | Functions to manage keyspaces and their related entities. | id, siteId, floorId, journeyId, dto, keyspaceId, order, validateKeyspaceDTO (Long, Long, Long, Long, KeyspaceCreationDTO, Long, int, ValidateKeyspaceDTO) : These are identifiers or data transfer objects used to perform operations on keyspaces. | keyspaceRepository.findById, siteRepository.existsById, keyspaceRepository.findAllBySite, floorRepository.findById, keyspaceRepository.save, journeyRepository.existsById, keyspaceRepository.findAllByJourney, keyspaceRepository.deleteById, appUserRepository.findByEmail, journeyRepository.findJourneyByPracticeIdAndSiteId, participateRepository.findByUserIdAndJourneyId, participateKeyspaceRepository.findByParticipateIdAndKeyspaceId |

# General

**Nom du fichier:** KeyspaceCreationDTO.java

**Langage:** Java

**Objectif du script:** This script is used to create a new 'Keyspace', which is a specific area in a virtual environment. It includes details like name, help text, code, floor ID, coordinates, and a list of clues.

# Explanation

The script is essentially a blueprint for creating a 'Keyspace' in a virtual game. It requires certain information like the name of the keyspace, help text, a unique code, the ID of the floor it's located on, its coordinates (X and Y), and a list of clues related to it.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| KeyspaceCreationDTO | This is a constructor function used to create an instance of the KeyspaceCreationDTO class. | name (String) : The name of the keyspace help (String) : The help text associated with the keyspace code (String) : A unique code for the keyspace floorId (Long) : The ID of the floor the keyspace is located on coordX (Double) : The X coordinate of the keyspace coordY (Double) : The Y coordinate of the keyspace clues (List<Clue>) : A list of clues associated with the keyspace |  |

# General

**Nom du fichier:** KeyspaceResponseDTO.java

**Langage:** Java

**Objectif du script:** This is a data transfer object that holds information about a 'keyspace', including its name, help text, associated floor, coordinates and a code. Some of this information may be null for users who haven't validated it.

# Explanation

The 'KeyspaceResponseDTO' file is used to transfer specific data related to a 'keyspace'. The data includes basic information like name and help text, but also more specific details like coordinates and a code. However, not all users will have access to all information, depending on their validation status.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** KeyspaceValidityResponseDTO.java

**Langage:** Java

**Objectif du script:** This script is essentially a blueprint for creating responses to check if a keyspace (a certain type of data storage) is valid or not.

# Explanation

This script is a data transfer object (DTO). It is a simple object that holds information about the validity of a keyspace. It is used to transfer data between different parts of the application, or between different applications.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** ValidateKeyspaceDTO.java

**Langage:** Java

**Objectif du script:** This code is used to validate the information related to a 'Keyspace', which includes a code and an ID. It ensures that these pieces of information are not missing when provided.

# Explanation

This script is a data transfer object (DTO) in Java, specifically for validating a 'Keyspace'. It checks if the 'Keyspace' code and ID are present. If not, it will show an error message.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** Journey.java

**Langage:** Java

**Objectif du script:** This script represents a 'Journey' in a gamification system, which includes its name, associated site, practice, participants, and related keyspaces.

# Explanation

The 'Journey' is a part of a gamification system. Each journey has a unique identity, a name, and is associated with a site and a practice. It also keeps track of participants and keyspaces, which are the steps in the journey.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** JourneyController

**Langage:** Java

**Objectif du script:** This code manages the journeys in the application. It enables creating, updating, deleting, and retrieving journeys.

# Explanation

This script is a part of a web application that manages 'journeys'. It provides functionalities like getting a journey by its ID, getting all journeys, creating a new journey, updating an existing journey, deleting a journey, and getting a journey by practice ID. It is meant for both administrators and users, with certain functionalities reserved for administrators only.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getJourneyById | Fetches a journey by its ID | id (Long) : The unique identifier for the journey | JourneyService.findJourneyById() |
| getAllJourneys | Fetches all journeys |  | JourneyService.findAll() |
| createJourney | Creates a new journey | journeyCreationDto (JourneyCreationDTO) : The details of the journey to be created | JourneyService.saveJourney() |
| updateJourney | Updates an existing journey | updatedJourney (JourneyCreationDTO) : The updated details of the journey id (Long) : The unique identifier for the journey | JourneyService.updateJourney() |
| deleteJourneyById | Deletes a journey by its ID | id (Long) : The unique identifier for the journey | JourneyService.delete() |
| getJourneyByPracticeId | Fetches journeys by practice ID | id (Long) : The unique identifier for the practice | JourneyService.findJourneysByPracticeId() |

# General

**Nom du fichier:** JourneyMapper.java

**Langage:** Java

**Objectif du script:** This script is used to convert between different data formats related to 'Journey'. It has two main functions: one for converting a 'Journey' object to a 'JourneyResponseDTO' object, and another for converting a 'JourneyCreationDTO' object to a 'Journey' object.

# Explanation

The script is part of a system managing 'Journeys'. It helps in transforming data from the format used in the system (Journey) to a format that can be sent to users (JourneyResponseDTO), and vice versa. It's like a translator between the system and the users.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| entityToRequestDto | Converts a 'Journey' object to a 'JourneyResponseDTO' object. | journey (Journey) : The 'Journey' object to be converted. |  |
| creationDtoToEntity | Converts a 'JourneyCreationDTO' object to a 'Journey' object. | journeyCreationDTO (JourneyCreationDTO) : The 'JourneyCreationDTO' object to be converted. |  |

# General

**Nom du fichier:** JourneyRepository.java

**Langage:** Java

**Objectif du script:** This script is used to fetch information about 'journeys' from a database. 'Journey' here refers to a specific entity in our system, and this script allows us to retrieve these journeys based on certain criteria like their associated practice ID or site ID.

# Explanation

The script is a part of our system that interacts with our database. It helps us get data about 'journeys'. It can fetch all journeys related to a specific practice or a specific journey related to both a practice and a site.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findJourneysByPracticeId | Fetches a list of journeys that are associated with a particular practice ID. | practiceId (Long) : ID of the practice to fetch journeys for. |  |
| findJourneyByPracticeIdAndSiteId | Fetches a specific journey that is associated with a particular practice ID and site ID. | practiceId (Long) : ID of the practice to fetch journey for. siteId (Long) : ID of the site to fetch journey for. |  |

# General

**Nom du fichier:** JourneyService.java

**Langage:** Java

**Objectif du script:** This script is a service layer in a web application that handles the business logic related to 'Journey'. It provides functionalities like finding, saving, updating, deleting a 'Journey' and finding 'Journeys' by practice ID or by practice and site ID.

# Explanation

This script is essentially a blueprint for operations that can be performed on 'Journey'. It includes operations such as retrieving a 'Journey' by its ID, retrieving all 'Journeys', saving or updating a 'Journey', deleting a 'Journey', and finding 'Journeys' based on practice ID or both practice and site ID.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findJourneyById | Finds a 'Journey' by its ID. | id (Long) : The ID of the 'Journey'. |  |
| findAll | Finds all 'Journeys'. |  |  |
| saveJourney | Saves a new 'Journey'. | journeyCreationDto (JourneyCreationDTO) : The 'Journey' to be saved. |  |
| updateJourney | Updates an existing 'Journey'. | updatedJourney (JourneyCreationDTO) : The updated 'Journey'. id (Long) : The ID of the 'Journey' to be updated. |  |
| delete | Deletes a 'Journey'. | id (Long) : The ID of the 'Journey' to be deleted. |  |
| findJourneysByPracticeId | Finds 'Journeys' by practice ID. | id (Long) : The practice ID. |  |
| findJourneyByPracticeIdAndSiteId | Finds a 'Journey' by practice ID and site ID. | practiceId (Long) : The practice ID. siteId (Long) : The site ID. |  |

# General

**Nom du fichier:** JourneyServiceImpl

**Langage:** Java

**Objectif du script:** This script is a service in our system that manages 'journeys'. It can find, create, update, and delete journeys. It also allows finding journeys by their associated 'practice id', and by both 'practice id' and 'site id'.

# Explanation

The script provides functionalities for handling 'journeys' in our system. It can find a journey by its ID, get all journeys, create a new journey, update an existing journey, and remove a journey. It also has the ability to find journeys based on their associated 'practice id', and by 'practice id' and 'site id' together.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findJourneyById | Finds a journey by its ID. | id (Long) : The ID of the journey to find. | journeyRepository.findById(), journeyMapper.entityToRequestDto(), ResourceNotFoundException.ResourceNotFoundException() |
| findAll | Returns all journeys. |  | journeyRepository.findAll(), journeyMapper.entityToRequestDto() |
| saveJourney | Creates a new journey. | journeyCreationDto (JourneyCreationDTO) : Data Transfer Object containing the information for the new journey. | journeyMapper.creationDtoToEntity(), journeyRepository.save(), journeyMapper.entityToRequestDto() |
| updateJourney | Updates an existing journey. | updatedJourney (JourneyCreationDTO) : Data Transfer Object containing the updated information for the journey. id (Long) : The ID of the journey to update. | journeyRepository.findById(), journeyRepository.save(), journeyMapper.entityToRequestDto(), ResourceNotFoundException.ResourceNotFoundException() |
| delete | Deletes a journey. | id (Long) : The ID of the journey to delete. | journeyRepository.deleteById() |
| findJourneysByPracticeId | Finds journeys by their associated 'practice id'. | id (Long) : The 'practice id' to find journeys by. | journeyRepository.findJourneysByPracticeId(), journeyMapper.entityToRequestDto() |
| findJourneyByPracticeIdAndSiteId | Finds a journey by both 'practice id' and 'site id'. | practiceId (Long) : The 'practice id' to find the journey by. siteId (Long) : The 'site id' to find the journey by. | journeyRepository.findJourneyByPracticeIdAndSiteId(), ResourceNotFoundException.ResourceNotFoundException() |

# General

**Nom du fichier:** JourneyCreationDTO.java

**Langage:** Java

**Objectif du script:** This code is used to create a new 'Journey' with specific attributes such as name, site id, and practice id.

# Explanation

The script defines a 'Journey' in our system. A 'Journey' represents a certain path or process, it requires a name, a site id, and a practice id for it to be created. The script ensures that these fields are correctly provided before a 'Journey' can be created.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** JourneyResponseDTO.java

**Langage:** Java

**Objectif du script:** This script is a blueprint for a data transfer object that represents a journey. It contains information such as the journey's id, name, site, and practice.

# Explanation

This script is a template for creating a 'Journey' object. This object is used to transfer data about a journey, including the journey's id, name, the site where the journey takes place, and the practice associated with the journey.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** Floor.java

**language:** Java

**Objectif du script:** This code represents a 'Floor' in a building. It includes details such as the floor's name, its order in the building, and a map of the floor.

# Explanation

The 'Floor' class is a blueprint for creating 'Floor' objects. Each 'Floor' has an ID, name, display order, map data, map type, and associated site. It's used to store and manage information about different floors in a building.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** FloorController

**Langage:** Java

**Objectif du script:** This is a management system for floors in a building. It allows users to create, update, delete, and fetch information about floors.

# Explanation

The FloorController is a component of a building management system. It provides several functionalities such as creating, updating, and deleting floors. It also allows users to fetch information about all floors or a specific floor by its ID. The system ensures that only authorized users (Administrators or Users) can perform these operations.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getAllFloors | Fetches information about all floors. |  | floorService.getAllFloors |
| getFloorById | Fetches information about a specific floor using its ID. | id (Long) : The ID of the floor. | floorService.getFloorById |
| createFloor | Creates a new floor for a specific site. | siteId (Long) : The ID of the site. floorCreationDto (FloorCreationDto) : The information required to create a new floor. | floorService.createFloor |
| updateFloorById | Updates information about a specific floor using its ID. | id (Long) : The ID of the floor. floorCreationDto (FloorCreationDto) : The information required to update a floor. | floorService.updateFloor |
| deleteFloorById | Deletes a specific floor using its ID. | id (Long) : The ID of the floor. | floorService.deleteFloor |
| getFloorsBySiteId | Fetches information about all floors for a specific site. | siteId (Long) : The ID of the site. | floorService.getFloorsBySiteId |
| getFloorsAndKeyspaces | Fetches information about all floors and their associated keyspaces for a specific site. |  | floorService.getFloorsAndKeyspacesForUser |

# General

**Nom du fichier:** FloorDBInit.java

**Langage:** Java

**Objectif du script:** This code initializes the floor data in the database by reading image files for each floor and storing them along with their respective details.

# Explanation

The script is used to set up floor data in a database. It reads image files for each floor, then stores these images and their details in the database. This is useful for applications that need to display or use floor information, such as a building navigation app.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| run | This function reads and stores the details of each floor in the database. | args (ApplicationArguments) : Arguments provided when the application is run. | floorRepository.findById(), floorRepository.saveAndFlush() |
| readFile | This function reads an image file for a specific floor. | i (long) : The floor number. | ResourceUtils.getFile() |

# General

**Nom du fichier:** FloorRepository

**Langage:** Java

**Objectif du script:** This script is used to interact with the database to fetch data related to Floors and Keyspaces in a gaming application.

# Explanation

The script is part of a gaming application and it helps to fetch data related to Floors and Keyspaces. It interacts with the database to get data based on user, site, and practice identifiers. It is used in the backend to handle data requests.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findBySite | Fetches all the floors associated with a particular site. | site (Site) : The site object for which floors are to be fetched. |  |
| getMapDataAndMapMimeById | Fetches map data for a particular floor based on its ID. | floorId (Long) : The unique identifier of the floor. |  |
| findFloorsAndKeyspacesBySiteAndPractice | Fetches the details of floors and keyspaces based on user, site, and practice identifiers. | userId (Long) : The unique identifier of the user. siteId (Long) : The unique identifier of the site. practiceId (Long) : The unique identifier of the practice. |  |

# General

**Nom du fichier:** FloorService.java

**Langage:** Java

**Objectif du script:** This script is used to manage floors in a building. It allows the creation, updating, deletion, and retrieval of floor information.

# Explanation

This script is a part of the floor management system. It has functions to add, update, delete, and retrieve floor details. It also allows fetching all floors or floors specific to a site.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getFloorById | Fetches the floor details using the floor ID. | id (Long) : Unique identifier of the floor. |  |
| getFloorsBySiteId | Fetches all floors related to a specific site. | siteId (Long) : Unique identifier of the site. |  |
| createFloor | Creates a new floor for a specific site. | siteId (Long) : Unique identifier of the site. floorCreationDto (FloorCreationDto) : Data transfer object containing the floor details. |  |
| getAllFloors | Fetches all the floors. |  |  |
| updateFloor | Updates the details of a specific floor. | floorId (Long) : Unique identifier of the floor. floorUpdateDto (FloorCreationDto) : Data transfer object containing the updated floor details. |  |
| deleteFloor | Deletes a specific floor. | floorId (Long) : Unique identifier of the floor. |  |

# General

**Nom du fichier:** FloorServiceImpl.java

**Langage:** Java

**Objectif du script:** This script manages the operations related to floors in a building, such as getting floor details, creating, updating, and deleting floors.

# Explanation

The script contains a service that handles functionalities related to floors. It can fetch floor details by its ID, create a new floor, update existing floor details, delete a floor, and retrieve all floors. It also supports getting floor map data and fetching floor details for a specific user.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getFloorById | Fetches the details of a floor using its ID. | id (Long) : The ID of the floor. | floorRepository.findById(), floorMapper.apply() |
| getFloorsBySiteId | Fetches the floors associated with a specific site. | siteId (Long) : The ID of the site. | siteRepository.findById(), floorRepository.findBySite(), floorMapper.apply() |
| createFloor | Creates a new floor for a specific site. | siteId (Long) : The ID of the site where the floor will be created. floorCreationDto (FloorCreationDto) : The details of the floor to be created. | siteRepository.findById(), floorMapper.mapToEntity(), floorRepository.save(), floorMapper.apply() |
| updateFloor | Updates the details of a specific floor. | floorId (Long) : The ID of the floor to be updated. floorUpdateDto (FloorCreationDto) : The new details of the floor. | floorRepository.findById(), floorRepository.save(), floorMapper.apply() |
| deleteFloor | Deletes a specific floor. | floorId (Long) : The ID of the floor to be deleted. | floorRepository.findById(), floorRepository.delete() |
| getMapDataByFloorId | Fetches map data of a specific floor. | floorId (Long) : The ID of the floor. | floorRepository.getMapDataAndMapMimeById() |
| getFloorsAndKeyspacesForUser | Fetches floor and keyspace details for the currently authenticated user. |  | SecurityUtils.getAuthenticatedUser(), appUserRepository.findByEmail(), retrieveFloorKeySpacesForUser() |
| retrieveFloorKeySpacesForUser | Fetches floor and keyspace details for a specific user. | appUser (AppUser) : The user for whom floor and keyspace details are to be fetched. | floorRepository.findFloorsAndKeyspacesBySiteAndPractice(), createFloorDTO() |
| createFloorDTO | Creates a new FloorDTO object from a FloorKeyspaceProjection object. | projection (FloorKeyspaceProjection) : The projection object from which the FloorDTO object will be created. |  |

# General

**Nom du fichier:** FloorKeyspaceProjection

**Langage:** Java

**Objectif du script:** This code contains a list of functions that are used to get information about a specific floor in a game. It includes details such as the floor name, its coordinates, and whether a user has visited it.

# Explanation

The 'FloorKeyspaceProjection' is an interface in Java that contains methods to retrieve various details about a game floor. It provides the ability to fetch the floor name, location, clues consumed, participation ID, display order, and user visit status.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getFloorId | Fetches the unique identifier of the floor. |  |  |
| getFloorName | Fetches the name of the floor. |  |  |
| getKeyspaceId | Fetches the unique identifier of the keyspace. |  |  |
| getSpaceName | Fetches the name of the space. |  |  |
| getHelp | Fetches the help text. |  |  |
| getFloorX | Fetches the X-coordinate of the floor. |  |  |
| getFloorY | Fetches the Y-coordinate of the floor. |  |  |
| getConsumedClue | Fetches the number of clues consumed. |  |  |
| getParticipateId | Fetches the unique identifier of the participant. |  |  |
| getDisplayOrder | Fetches the display order of the floor. |  |  |
| getFloorVisitedByUser | Checks if the floor has been visited by the user. |  |  |

# General

**Nom du fichier:** FloorMapDataView.java

**Langage:** Java

**Objectif du script:** This file provides a blueprint for objects that hold information about a floor map, such as the map data itself and the type of data.

# Explanation

This script is a Java interface named FloorMapDataView. It's a guideline for creating objects that hold floor map data. Objects created from this blueprint will have two properties: the actual map data and the type of this data.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| getMapData | This function retrieves the map data. |  |  |
| getMapMime | This function retrieves the type of the map data. |  |  |

# General

**Nom du fichier:** FloorCreationDto.java

**Langage:** Java

**Objectif du script:** This script is used to create a new 'floor' in our system. A floor has a name, an order, and an image associated with it.

# Explanation

This script defines the data structure for a 'floor' in our system, which includes the floor name, the order in which it is displayed, and the image of the floor. This is used when a new floor is created.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** FloorDTO.java

**Langage:** Java

**Objectif du script:** This script is used to manage the details of a 'Floor'. It includes attributes like floor id, name, display order, and related keyspaces.

# Explanation

The 'FloorDTO' script is a representation of a 'Floor' in the system. It holds information such as the unique identifier of the floor, its name, the order in which it is displayed, and the keyspaces associated with it.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** FloorMapper.java

**Langage:** Java

**Objectif du script:** This script is responsible for converting data between two objects: 'Floor' and 'FloorResponseDto'. It also converts 'FloorCreationDto' into 'Floor'. This is necessary when we want to present data in a different format or when we receive data in a format and need to convert it to our application's format.

# Explanation

The script contains two main functions: 'apply' and 'mapToEntity'. The 'apply' function takes a 'Floor' object and converts it into a 'FloorResponseDto' object. The 'mapToEntity' function takes a 'FloorCreationDto' object and converts it into a 'Floor' object. These conversions are necessary to ensure the data is in the correct format for our application.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| apply | Converts a 'Floor' object into a 'FloorResponseDto' object. | floor (Floor) : The 'Floor' object to be converted. |  |
| mapToEntity | Converts a 'FloorCreationDto' object into a 'Floor' object. | floorCreationDto (FloorCreationDto) : The 'FloorCreationDto' object to be converted. |  |

# General

**Nom du fichier:** FloorResponseDto.java

**Langage:** Java

**Objectif du script:** This file is used to structure the response data when information about a floor is requested.

# Explanation

The FloorResponseDto class is a data structure that contains the information that is sent back when a request for information about a floor is made. It extends the FloorCreationDto class, meaning it includes all the data from that class, plus an additional 'id' field.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** KeyspaceDTO.java

**language:** Java

**Objectif du script:** This file is used to store and manage information related to a specific area or 'Keyspace'. It includes details such as the Keyspace's ID, name, coordinates, and whether it has been visited or not.

# Explanation

The KeyspaceDTO class is a data transfer object that is used to transfer data about a 'Keyspace', which could be a specific area or location in a game. It keeps track of various properties including its ID, name, coordinates, and information related to player interaction like whether it has been visited or not.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |

# General

**Nom du fichier:** AccountDeletionException.java

**language:** Java

**Objectif du script:** This is a custom error message that is shown when there is a problem deleting a user account.

# Explanation

This script defines a custom error message that is used when there's an issue with deleting a user account. It is designed to handle and communicate specific problems that can occur in this scenario, making it easier for the user to understand what went wrong.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| AccountDeletionException(String message) | Creates a new instance of the AccountDeletionException with a specific error message. | message (String) : The specific error message. |  |
| AccountDeletionException(String message, Throwable cause) | Creates a new instance of the AccountDeletionException with a specific error message and the cause of the error. | message (String) : The specific error message. cause (Throwable) : The underlying cause of the error. |  |

# General

**Nom du fichier:** AlreadyExistsException.java

**Langage:** Java

**Objectif du script:** This code is a custom error message that is displayed when there is an attempt to create a resource that already exists in the system.

# Explanation

The code defines a custom exception called 'AlreadyExistsException'. It's used to indicate a conflict, specifically when trying to create a resource that already exists. The error message includes the name of the resource, the field associated with the resource, and the value of the field.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| AlreadyExistsException | This is a constructor for the AlreadyExistsException class. It initializes the resourceName, fieldName, and fieldValue variables and constructs an error message. | resourceName (String) : The name of the resource that was found fieldName (String) : The name of the field associated with the resource fieldValue (String) : The value of the field associated with the resource |  |

# General

**Nom du fichier:** CustomAccessDeniedHandler.java

**Langage:** Java

**Objectif du script:** This script handles the situation when a user tries to access a part of the system they do not have permission for, by providing a custom error message.

# Explanation

The script is a custom error handler for access denied (403 Forbidden) responses. When a user tries to access a resource they don't have permission for, this handler generates a clear error message explaining the situation, which is then returned to the user.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| handle | Handles the Access Denied exception by writing a custom error response. | request (HttpServletRequest) : The original request that resulted in an error. response (HttpServletResponse) : The response object where the error message will be written. accessDeniedException (AccessDeniedException) : The exception that was thrown due to access being denied. | jakarta.servlet.http.HttpServletRequest, jakarta.servlet.http.HttpServletResponse, org.springframework.security.access.AccessDeniedException |

# General

**Nom du fichier:** CustomAuthenticationEntryPoint.java

**Langage:** Java

**Objectif du script:** This script handles the scenario where a user tries to access a resource they are not authorized to, by providing a custom response.

# Explanation

The script is part of a security system. It intercepts unauthorized access attempts and responds with a custom message. It is designed to enhance the system's security by providing more information about unauthorized access attempts.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| commence | This function is triggered when an unauthorized access attempt is detected. It creates a custom error response and sends it back to the user. | request (HttpServletRequest) : The request that was made by the user. response (HttpServletResponse) : The response that will be sent to the user. authException (AuthenticationException) : The exception that was thrown due to unauthorized access. | jakarta.servlet.http.HttpServletRequest, jakarta.servlet.http.HttpServletResponse, org.springframework.security.core.AuthenticationException |

# General

**Nom du fichier:** EmailRegistrationException.java

**Langage:** Java

**Objectif du script:** This code is used to display an error message when an invalid email is used during registration.

# Explanation

The script is a custom exception in Java, which is triggered when a user tries to register with an invalid email. It then sends a specific error message to the user.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| EmailRegistrationException() | This function generates a custom error message when an invalid email is used for registration. |  | org.springframework.http.HttpStatus, org.springframework.web.bind.annotation.ResponseStatus |

# General

**Nom du fichier:** ErrorResponse.java

**Langage:** Java

**Objectif du script:** This code is for an error response system. It gathers information about an error, such as its status and message, and returns this information when an error occurs.

# Explanation

This is a Java class called 'ErrorResponse' that handles errors occurring in the system. It collects important details about the error, such as the HTTP status, path, message, and timestamp. It also allows for the addition of specific error messages related to certain fields.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| ErrorResponse | This is a constructor for the ErrorResponse class. It initializes the error response with the status, path, and message of the error. | status (HttpStatus) : The HTTP status of the error. path (String) : The path where the error occurred. message (String) : The error message. |  |
| addError | This function allows to add specific error messages related to certain fields. | field (String) : The field where the error occurred. message (String) : The specific error message for the field. |  |

# General

**Nom du fichier:** GlobalExceptionHandler.java

**Langage:** Java

**Objectif du script:** This script manages how different types of errors in the application are handled and reported to the user.

# Explanation

This script is responsible for handling various types of errors that can occur in the application. It ensures that these errors are appropriately reported to the user, providing details about what went wrong and where the error occurred. It handles a wide range of errors, from validation failures to access denials.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| handleValidationException | Handles validation errors in the application. | ex (MethodArgumentNotValidException) : The exception that occurred. request (HttpServletRequest) : The HTTP request during which the exception occurred. | ThrowableUtil.getStackTrace() |
| handleAlreadyExistsException | Handles errors when an entity already exists. | ex (AlreadyExistsException) : The exception that occurred. request (HttpServletRequest) : The HTTP request during which the exception occurred. | ThrowableUtil.getStackTrace() |
| handleEmailRegistrationException | Handles errors related to email registration. | ex (EmailRegistrationException) : The exception that occurred. request (HttpServletRequest) : The HTTP request during which the exception occurred. | ThrowableUtil.getStackTrace() |
| handleInvalidTokenException | Handles errors related to invalid tokens. | ex (InvalidTokenException) : The exception that occurred. request (HttpServletRequest) : The HTTP request during which the exception occurred. | ThrowableUtil.getStackTrace() |
| handleException | Handles general exceptions in the application. | ex (Exception) : The exception that occurred. request (HttpServletRequest) : The HTTP request during which the exception occurred. | ThrowableUtil.getStackTrace() |
| mapFieldErrors | Maps field errors for data binding and validation. | bindingResult (BindingResult) : Result of the binding and validation process. |  |

# General

**Nom du fichier:** InvalidTokenException

**Langage:** Java

**Objectif du script:** This code handles situations where an incorrect or invalid token is provided, to ensure the system's security.

# Explanation

This program is a special kind of error handler in Java, specifically for situations where an invalid token is provided. It helps maintain the system's security by ensuring only valid tokens are accepted. If an invalid token is detected, it triggers an error message to alert the system.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| InvalidTokenException | This function triggers an error message when an invalid token is detected. | errorMessage (String) : This is the specific error message that will be displayed when an invalid token is detected. |  |
| InvalidTokenException | This function triggers an error message when an invalid token is detected and also includes the cause of the error. | errorMessage (String) : This is the specific error message that will be displayed when an invalid token is detected. cause (Throwable) : This is the underlying cause of the error. |  |

# General

**Nom du fichier:** KeyspaceValidationException.java

**Langage:** Java

**Objectif du script:** This file contains a custom error that is triggered when there is a problem with the validation of a keyspace.

# Explanation

This piece of code is a custom error handling class in Java, used in cases when there's an issue with the validation of a keyspace. It returns a 'Bad Request' HTTP status to indicate that the server couldn't understand the request due to invalid syntax.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| KeyspaceValidationException | This function is a constructor for the KeyspaceValidationException class. It takes a message as an argument, which is then passed to the superclass constructor. | message (String) : This is the error message that will be displayed when this exception is thrown. | RuntimeException.super |

# General

**Nom du fichier:** RegistrationException.java

**Langage:** Java

**Objectif du script:** This script is designed to handle errors that may occur during user registration. It provides detailed error messages to help understand what went wrong.

# Explanation

The script defines a custom exception, RegistrationException, which is used when there is an issue during the registration process. This exception provides detailed error messages to help troubleshoot the issue.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| RegistrationException(String errorMessage) | This function creates a new RegistrationException with a specific error message. | errorMessage (String) : The detailed error message that describes what went wrong during registration. |  |
| RegistrationException(String errorMessage, Throwable cause) | This function creates a new RegistrationException with a specific error message and the original cause of the error. | errorMessage (String) : The detailed error message that describes what went wrong during registration. cause (Throwable) : The original cause of the error. |  |

# General

**Nom du fichier:** ResourceNotFoundException.java

**Langage:** Java

**Objectif du script:** This code is used to handle situations where a requested resource is not found in the system.

# Explanation

The script is a custom exception handler in Java for when a specific resource isn't found in the system. It provides detailed information about the missing resource such as its name, the field name, and the field value.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| ResourceNotFoundException | This function creates a new instance of the ResourceNotFoundException with the given resource name, field name, and field value. | resourceName (String) : The name of the resource that was not found fieldName (String) : The name of the field associated with the resource fieldValue (Object) : The value of the field associated with the resource |  |

# General

**Nom du fichier:** ApplicationConfig

**Langage:** Java

**Objectif du script:** This script is designed to manage user authentication and security in an application. It verifies user credentials, manages login sessions, and encrypts passwords for security.

# Explanation

The script includes functions for user authentication, password encryption, and user management. It ensures that only authorized users can access the application, and it protects user data by encrypting passwords.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| userDetailsService | This function verifies user credentials during login. | username (String) : The username of the user trying to login. | AppUserRepository.findByEmail() |
| authenticationProvider | This function manages user authentication. |  | userDetailsService(), passwordEncoder() |
| authenticationManager | This function manages the authentication configuration. | configuration (AuthenticationConfiguration) : The configuration details for authentication. |  |
| passwordEncoder | This function encrypts user passwords for security. |  |  |

# General

**Nom du fichier:** OpenApiConfig.java

**Langage:** Java

**Objectif du script:** This script is used to configure the OpenAPI documentation for the application. It includes details such as contact information, license details, server details, and security requirements.

# Explanation

This Java script is responsible for setting up the OpenAPI documentation for the software. It provides essential details like contact information, license, server details for both development and production environments, and security requirements. This helps users to understand the API's functionalities, usage, and security protocols.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| OpenApiConfig | A constructor that initializes the OpenAPI documentation properties and logs a message. | documentationProperties (OpenApiProperties) : Properties to be used for OpenAPI documentation | LoggerFactory.getLogger |
| openAPI | A method that creates and returns an OpenAPI object with all the necessary details filled in. |  | Contact.name, Contact.email, Contact.url, License.name, License.url, Server.description, Server.url, Info.title, Info.version, Info.contact, Info.license, Info.description, SecurityRequirement.addList, SecurityScheme.name, SecurityScheme.type, SecurityScheme.scheme, SecurityScheme.bearerFormat |

# General

**Nom du fichier:** Clue.java

**Langage:** Java

**Objectif du script:** This script is used to manage and interact with clues in a game. It allows for the creation, reading, and modification of clues, each identified by a unique ID, a text, and an order number.

# Explanation

This Java class 'Clue' represents a clue in a gamification system. Each clue has a unique ID, a text, and an order number. It's used to store and manage information about different clues in the game, allowing the system to keep track of them.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** ClueController

**Langage:** Java

**Objectif du script:** This is a controller file in a gamification application that handles requests related to 'clue' management. It is mainly used by administrators for managing keyspace.

# Explanation

The ClueController is a part of the gamification application that is responsible for managing the keyspace. It is used by administrators and is not intended to be used by regular users. It is a REST controller that handles HTTP requests and responses.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** ClueMapper

**Langage:** Java

**Objectif du script:** This code is used to convert the data associated with clues in a game into a format that can be easily understood and used by other parts of the system.

# Explanation

The ClueMapper code takes in game clues and transforms them into a more user-friendly format. It does this for both individual clues and lists of clues.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| entityToResponseDTO | Converts a single clue into a user-friendly format. | clue (Clue) : The clue to be converted. |  |
| entityListToResponseDTO | Converts a list of clues into a user-friendly format. | list (List<Clue>) : The list of clues to be converted. | java.util.stream.Collectors.toList |

# General

**Nom du fichier:** ClueRepository.java

**Langage:** Java

**Objectif du script:** This file is a link between the system and the database for all operations related to 'Clue'.

# Explanation

This script is used to interact with the 'Clue' data in the database. It allows the system to retrieve, update, delete, and create new entries for 'Clue'.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** ClueService

**Langage:** Java

**Objectif du script:** This is a placeholder for a service that manages clues in a game. Currently, it doesn't have any defined operations.

# Explanation

This file, named 'ClueService', is a blueprint for a service that will handle game clues. It's written in Java. At the moment, it doesn't specify any operations or actions. Once these operations are defined, they will dictate how clues are managed in the game.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** ClueServiceImpl.java

**Langage:** Java

**Objectif du script:** This file is a placeholder for a service in the application related to 'clues'. Currently, it doesn't have any functionality.

# Explanation

This is a Java file named 'ClueServiceImpl'. It's a part of the 'gamification' module of the application, specifically dealing with 'clues'. It is marked as a Service, meaning it's meant to perform business operations. However, it currently doesn't contain any methods or functions, hence it doesn't perform any operations.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** ClueDTO.java

**Langage:** Java

**Objectif du script:** This is a blueprint for creating 'clue' objects in a game. Each clue has a unique ID, some text, and an order.

# Explanation

This script describes what a 'clue' looks like in a game. It holds an ID for unique identification, a text for the clue content, and an order to determine the sequence of the clues in the game.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** AuthenticationController.java

**Langage:** Java

**Objectif du script:** This script manages user authentication and registration. It includes functions for registering new users, verifying email addresses, logging in, checking token validity, updating passwords, and resetting passwords.

# Explanation

The script is responsible for user authentication and registration. It provides various endpoints for users to register, login, verify their email, validate their registration tokens, update their password, and reset their password.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| register | Registers a new user. | request (RegisterRequestDTO) : Information needed to register a new user. token (String) : Token for user registration. | AuthService.register |
| requestRegistrationEmail | Requests a registration email. | emailVerificationDTO (EmailVerificationDTO) : Email information for verification. | AuthService.requestRegistrationEmail |
| authenticate | Authenticates a user. | request (AuthenticationRequestDTO) : Information needed for user authentication. | AuthService.authenticate |
| checkTokenValidity | Checks the validity of a registration token. | token (String) : Token for checking validity. | AuthService.validateFirstRegistrationToken |
| updateUserPassword | Updates a user's password. | updatePasswordDTO (UpdatePasswordDTO) : Information needed to update a user's password. | AuthService.updateUserPassword |
| forgotPassword | Requests a password reset. | emailVerificationDTO (EmailVerificationDTO) : Email information for password reset. | AuthService.forgotPassword |
| validateResetPasswordToken | Checks the validity of a reset password token. | token (String) : Token for checking validity. | AuthService.validateResetPasswordToken |

# General

**Nom du fichier:** VerificationLink.java

**Langage:** Java

**Objectif du script:** This code defines a 'VerificationLink' object that stores details like email and token for user authentication. It also records the time when the link was created.

# Explanation

This script is part of a user authentication system. It manages the creation of verification links sent to users' emails for account confirmation. The system creates a unique token for each user that is stored alongside the user's email and the creation time of the verification link.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** VerificationLinkRepository

**Langage:** Java

**Objectif du script:** This script is used to access and manage the data related to verification links in the database. It provides functionalities to search for verification links by email, token, or both.

# Explanation

The VerificationLinkRepository is a script that communicates with the database to manage verification links. It retrieves verification links based on email, token, or a combination of both. This is useful in verifying user identities during processes such as account creation and password reset.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findByEmail | Fetches a verification link associated with a specific email. | email (String) : Email address to search for. |  |
| findByToken | Fetches a verification link associated with a specific token. | token (String) : Token to search for. |  |
| findByEmailAndToken | Fetches a verification link associated with a specific email and token. | email (String) : Email address to search for. token (String) : Token to search for. |  |

# General

**Nom du fichier:** VerificationLinkService

**Langage:** Java

**Objectif du script:** This is a service that manages email verification links. It allows creating, checking, and deleting verification links, as well as generating tokens and templates.

# Explanation

This service is responsible for all operations related to email verification links. It creates a new link for a given email, checks if a token is valid, generates new tokens, verifies emails and tokens, and deletes verification links.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| createVerificationLink | Creates a new verification link for a given email. | email (String) : The email for which the verification link needs to be created. |  |
| checkTokenValidity | Checks if a token is still valid. | urlToken (String) : The verification token extracted from the URL. |  |
| generateToken | Generates a new random token. |  |  |
| verifyByEmailAndToken | Verifies an email and a token. | email (String) : The email to verify. token (String) : The token to verify. |  |

# General

**Nom du fichier:** VerificationLinkServiceImpl.java

**Langage:** Java

**Objectif du script:** This script is responsible for creating, retrieving, verifying, and deleting verification links. These links are used during the registration process to confirm a user's email address.

# Explanation

The script is part of an email verification process. It creates a unique link for each user, verifies it, and removes it once it's no longer needed. It also checks if a verification link is still valid.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| createVerificationLink | Creates a new verification link for a given email address. | email (String) : The email address for which the verification link is created. | TokenUtils.generateToken, VerificationLinkRepository.save |
| getVerificationLinkByToken | Retrieves a verification link by its token. | token (String) : The token of the verification link. | VerificationLinkRepository.findByToken |
| generateTemplateUrl | Generates the URL for the verification link. | verificationLink (VerificationLink) : The verification link for which the URL is generated. |  |
| verifyByEmailAndToken | Verifies a verification link by its email and token. | email (String) : The email address associated with the verification link. token (String) : The token of the verification link. | VerificationLinkRepository.findByEmailAndToken |
| verifyByEmail | Verifies a verification link by its email. | email (String) : The email address associated with the verification link. | VerificationLinkRepository.findByEmail, checkTokenValidity |
| deleteVerificationLinkByEmailAndToken | Deletes a verification link by its email and token. | email (String) : The email address associated with the verification link. token (String) : The token of the verification link. | VerificationLinkRepository.findByEmailAndToken, VerificationLinkRepository.delete |
| checkTokenValidity | Checks the validity of a token. | urlToken (String) : The token to be checked. | VerificationLinkRepository.findByToken, TokenUtils.isTokenExpired, VerificationLinkRepository.delete |
| generateToken | Generates a unique token. |  | TokenUtils.generateToken, VerificationLinkRepository.findByToken |

# General

**Nom du fichier:** AuthService.java

**Langage:** Java

**Objectif du script:** This script is used to manage user authentication and password updates in the system.

# Explanation

The AuthService script is a service that handles user registration, authentication, email verification, and password updates. It's like a gatekeeper, ensuring only authorized users can access the system and can reset their passwords when necessary.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| register | Registers a new user. | request (RegisterRequestDTO) : Contains the registration details. token (String) : The token for registration. |  |
| authenticate | Authenticates a user. | request (AuthenticationRequestDTO) : Contains the authentication details. |  |
| requestRegistration | Requests registration for an email. | email (String) : The email for registration. |  |
| validateFirstRegistrationToken | Validates the first registration token. | token (String) : The token for validation. |  |
| updateUserPassword | Updates a user's password. | updatePasswordDTO (UpdatePasswordDTO) : Contains the new password details. |  |
| forgotPassword | Handles a forgotten password request. | email (String) : The email to send the reset password link to. |  |
| validateResetPasswordToken | Validates the reset password token. | token (String) : The token for validation. |  |

# General

**Nom du fichier:** AuthServiceImpl

**Langage:** Java

**Objectif du script:** This script is responsible for managing user authentication including registration, password reset, and email verification.

# Explanation

The script contains functions for user registration, authentication, email verification, password reset, and other related tasks. It interacts with various parts of the system like user data, journey data, and site data.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| register | Registers a new user in the system. | request (RegisterRequestDTO) : Contains user registration data. token (String) : Used for user email verification. | appUserRepository.save(), jwtService.generateToken(), verificationLinkService.deleteVerificationLinkByEmailAndToken() |
| authenticate | Authenticates a user using their email and password. | request (AuthenticationRequestDTO) : Contains user's email and password. | authenticationManager.authenticate(), appUserRepository.findByEmail(), jwtService.generateToken() |
| requestRegistration | Sends a registration request for a new user. | email (String) : Email of the user requesting registration. | verificationLinkService.verifyByEmail(), emailService.sendEmail() |
| forgotPassword | Handles user's request for password reset. | email (String) : Email of the user requesting password reset. | appUserRepository.findByEmail(), emailService.sendEmail() |

# General

**Nom du fichier:** PasswordResetToken.java

**Langage:** Java

**Objectif du script:** This script is responsible for managing password reset tokens. It creates a token, assigns it to a user, and sets an expiration date for the token.

# Explanation

The script 'PasswordResetToken' is part of a system that allows users to reset their passwords. It creates a unique token for each password reset request, assigns it to the user, and sets an expiration date for the token. This ensures that the token is only valid for a certain period of time, adding an extra layer of security.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| PasswordResetToken | This is the constructor of the PasswordResetToken class. It initializes the token, user and expiration date. | token (String) : This is the unique token for password reset. user (AppUser) : This is the user who requested the password reset. |  |
| getTokenExpirationDate | This function calculates the expiration date for the token. |  |  |

# General

**Nom du fichier:** PasswordResetTokenRepository.java

**Langage:** Java

**Objectif du script:** This file is a part of the system that manages password resets. It interacts with the database to find specific password reset tokens or to find a user's email.

# Explanation

This is a repository file in our system that is responsible for managing password reset tokens. It has the ability to search for a specific password reset token or to search for a user's email in the database.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findByToken | Searches for a password reset token in the database. | token (String) : The password reset token to search for. |  |
| findByUserEmail | Searches for a user's email in the database. | email (String) : The user's email to search for. |  |

# General

**Nom du fichier:** PasswordResetTokenService

**Langage:** Java

**Objectif du script:** This is a service that manages password reset tokens. It can generate tokens, check their validity, create or update them, generate a template URL, and check if a token already exists and is valid.

# Explanation

This service is responsible for handling the operations related to password reset tokens. It is used when users forget their password and need to reset it. It ensures that the tokens are valid and secure.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| generateToken | Generates a new password reset token. |  |  |
| checkTokenValidity | Checks if a given token is valid. | urlToken (String) : The token to be checked. |  |
| createResetPasswordTokenOrUpdate | Creates a new password reset token or updates an existing one. | email (String) : The email of the user who needs the password reset token. user (AppUser) : The user who needs the password reset token. |  |
| generateTemplateUrl | Generates a template URL for the password reset token. | token (String) : The token for which the URL is to be generated. |  |
| isAlreadyExistAndValid | Checks if a password reset token already exists and is valid. | email (String) : The email of the user to check for an existing valid token. |  |

# General

**Nom du fichier:** PasswordResetTokenServiceImp

**Langage:** Java

**Objectif du script:** This script is used to manage password reset tokens. It generates tokens, checks their validity, and provides a link for users to reset their passwords.

# Explanation

This code is part of a system's security measures. It helps users who have forgotten their passwords by creating a unique token, which is then used to verify the user's identity and allow them to create a new password. This process ensures that only the user with access to the token can reset the password, thereby preventing unauthorized access.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| createResetPasswordTokenOrUpdate | Creates a new password reset token or updates an existing one. | email (String) : The email address of the user. user (AppUser) : The user who wants to reset their password. | TokenUtils.generateToken, PasswordResetTokenRepository.save, PasswordResetTokenRepository.findByUserEmail |
| generateTemplateUrl | Generates a URL for the user to reset their password. | token (String) : The unique token generated for the password reset. |  |
| getResetPasswordToken | Retrieves the password reset token for a user. | email (String) : The email address of the user. | PasswordResetTokenRepository.findByUserEmail |
| generateToken | Generates a unique token for the password reset. |  | TokenUtils.generateToken, PasswordResetTokenRepository.findByToken |
| checkTokenValidity | Checks if a password reset token is valid. | urlToken (String) : The token to be checked. | PasswordResetTokenRepository.findByToken, TokenUtils.isTokenExpired, PasswordResetTokenRepository.delete |
| isAlreadyExistAndValid | Checks if a valid password reset token already exists for a user. | email (String) : The email address of the user. | PasswordResetTokenRepository.findByUserEmail, TokenUtils.isTokenExpired |

# General

**Nom du fichier:** AuthenticationRequestDTO.java

**Langage:** Java

**Objectif du script:** This script is used to handle user login requests. It contains information such as the user's email and password.

# Explanation

This script is a simple data transfer object (DTO) used to carry the user's login information. It includes the user's email and password. This information is used when a user attempts to log in to the system.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** AuthenticationResponseDTO.java

**Langage:** Java

**Objectif du script:** This code is for a data transfer object that carries the access token after a user has been successfully authenticated.

# Explanation

This Java file defines a data structure that holds an access token. This access token is used to validate the user's session after they have successfully logged in. The access token is labeled 'access\_token' when converted into JSON format.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** EmailVerificationDTO

**Langage:** Java

**Objectif du script:** This file is used to check if an email is valid or not.

# Explanation

The script is used to verify the validity of an email. It checks if the email is not blank, if it follows the standard email format, and if it has a valid domain. This helps ensure that users are providing a legitimate email address.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** EmailVerificationResponseDTO

**Langage:** Java

**Objectif du script:** This script is a blueprint for creating responses to email verification attempts. It can create three types of responses: success, already sent, and failure, each carrying a different message.

# Explanation

This script is designed to handle responses to email verification requests. It's a template that generates different types of responses based on the result of the email verification attempt. It can generate a success message when the verification is successful, an already sent message when the verification has already been done, and a failure message when the verification fails.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| success | Creates a success response for email verification | message (String) : The success message to be included in the response |  |
| alreadySent | Creates a response indicating that the email verification has already been sent | message (String) : The message indicating that the verification has already been sent |  |
| failure | Creates a failure response for email verification | message (String) : The failure message to be included in the response |  |

# General

**Nom du fichier:** RegisterRequestDTO

**Langage:** Java

**Objectif du script:** This script is used to handle the registration requests of users. It ensures that the necessary information such as first name, last name, password, avatar, practice ID and site ID are properly provided by the user.

# Explanation

This script is part of a registration process. It checks that the user provides their first and last name, a password that is at least 8 characters long and contains a special character, an avatar, and IDs for their practice and site. If any of these are not provided or do not meet the requirements, the registration request will not be processed.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** TokenValidityResponseDTO.java

**Langage:** Java

**Objectif du script:** This file is used to check if a token is valid or not and return the result.

# Explanation

The script is a simple data transfer object (DTO) in Java. It's responsible for carrying data between processes. In this case, it's used to carry the validity status of a token.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** UpdatePasswordDTO.java

**Langage:** Java

**Objectif du script:** This script is used to update a user's password. It ensures that the new password meets certain criteria for security purposes.

# Explanation

This script represents a data transfer object (DTO) for updating a user's password in our system. The new password must meet certain requirements to ensure security, such as including a mix of upper and lower case letters, numbers, and special characters. A token is also required for authentication purposes.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** AppUser.java

**language:** Java

**Objectif du script:** This file describes the properties and relationships of a user in a gamification system. It includes information such as the user's first and last name, email, password, and the date their account was created and last updated. It also connects the user to their role, site, practice, and participation records.

# Explanation

This is a Java class file that defines an 'AppUser' entity in a gamification system. It includes various attributes like first name, last name, email, password, and timestamps for creation and updates. It also establishes relationships with other entities like Role, Site, Practice, and Participate.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** AppUserController.java

**Langage:** Java

**Objectif du script:** This script is responsible for managing user accounts. It allows users to update their profile, delete their account, and get their account details.

# Explanation

The script provides three main functionalities: updating user's profile, deleting user's account, and retrieving user's account details. These functionalities are only accessible to authenticated users with 'Administrator' or 'User' roles.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| updateUser | This function allows users to update their profile. | updatedUser (UpdateUserDTO) : This is the updated user data. | AppUserService.updateUser() |
| deleteAccount | This function allows users to delete their account. |  | AppUserService.deleteAccount() |
| getUserDetails | This function allows users to retrieve their account details. |  | AppUserService.getUserDetails() |

# General

**Nom du fichier:** AppUserMapper.java

**Langage:** Java

**Objectif du script:** This script is a tool for converting between different data formats for user information. It allows the system to use user data in the most convenient format for each specific task.

# Explanation

This script is a mapper for user data. It converts user data to and from different formats. It has two main functions, one for converting a user entity to a response DTO (Data Transfer Object), and another for converting a creation DTO to a user entity.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| entityToResponseDto | Converts a user entity into a format that can be easily sent and received over the network. | appUser (AppUser) : The user entity to be converted. |  |
| creationDtoToEntity | Converts a creation DTO into a user entity that can be stored in the database. | appUserCreationDTO (AppUserCreationDTO) : The creation DTO to be converted. |  |

# General

**Nom du fichier:** AppUserRepository.java

**Langage:** Java

**Objectif du script:** This script is responsible for managing the data related to 'AppUsers', particularly fetching user data based on their email.

# Explanation

The script is a part of the application that handles the user data. It's like a librarian who knows exactly where to find the information about a user when given their email.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findByEmail | This function retrieves user information using their email. | email (String) : An email address of a user. | JpaRepository.findOne() |

# General

**Nom du fichier:** AppUserService.java

**Langage:** Java

**Objectif du script:** This is a service interface for managing user accounts. It provides methods for finding, updating, and deleting users based on their details such as ID and email. It also allows for password updates and account deletions.

# Explanation

The service interface 'AppUserService' provides a set of methods for user management. These methods include finding a user by ID or email, getting a list of all users, updating a user's details or password, and deleting a user account. The service is designed to be implemented by a class that will provide the actual logic for these operations.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findUserById | Finds a user by their ID. | id (Long) : The ID of the user. |  |
| findUserByEmail | Finds a user by their email. | email (String) : The email of the user. |  |
| findAllUsers | Gets a list of all users. |  |  |
| updateUser | Updates a user's details. | updatedUser (UpdateUserDTO) : The updated user details. |  |
| updatePassword | Updates a user's password. | password (String) : The new password. email (String) : The email of the user. |  |
| deleteUserById | Deletes a user by their ID. | id (Long) : The ID of the user. |  |
| getUserDetails | Gets the details of a user. |  |  |
| deleteAccount | Deletes a user account. |  |  |

# General

**Nom du fichier:** AppUserServiceImpl.java

**Langage:** Java

**Objectif du script:** This script manages the user data in the application. It includes functions for finding, updating, and deleting user accounts.

# Explanation

This file is part of a user management system for an application. It provides services like searching for users by their ID or email, retrieving all users, updating user details and passwords, and deleting accounts.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
| findUserById | Finds a user by their ID. | id (Long) : The unique ID of the user. | appUserRepository.findById(), appUserMapper.entityToResponseDto() |
| findUserByEmail | Finds a user by their email address. | email (String) : The email address of the user. | appUserRepository.findByEmail(), appUserMapper.entityToResponseDto() |
| findAllUsers | Retrieves all users. |  | appUserRepository.findAll(), appUserMapper.entityToResponseDto() |
| updateUser | Updates the details of a user. | updatedUser (UpdateUserDTO) : Data transfer object containing the updated user details. | appUserRepository.findByEmail(), appUserRepository.save() |
| updatePassword | Updates the password of a user. | password (String) : The new password. email (String) : The email of the user. | appUserRepository.findByEmail(), appUserRepository.save() |
| deleteUserById | Deletes a user by their ID. | id (Long) : The unique ID of the user. | appUserRepository.deleteById() |
| deleteAccount | Deletes the account of the currently authenticated user. |  | appUserRepository.findByEmail(), appUserRepository.delete() |
| getUserDetails | Retrieves the details of the currently authenticated user. |  | appUserRepository.findByEmail(), appUserMapper.entityToResponseDto() |

# General

**Nom du fichier:** AppUserCreationDTO

**Langage:** Java

**Objectif du script:** This code is used to create a new user for an application. It validates the provided information such as email, first name, last name, password, practice ID, and site ID.

# Explanation

The code is part of an application that manages users. It checks if the information provided by a user during registration, like email, name, and password, is valid and meets certain criteria. It also associates the user with a particular practice and site.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** AppUserResponseDTO

**Langage:** Java

**Objectif du script:** This script is for a user's profile in an application. It includes details like name, email, avatar, site, and when the profile was created or updated.

# Explanation

The script is a model of a user's profile in an application. It validates the input data such as the user's name, email, avatar, practice, and site. It also records the timestamps of when the profile was updated or created.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |

# General

**Nom du fichier:** UpdateUserDTO.java

**Langage:** Java

**Objectif du script:** This file is used to validate and manage the data when updating user's information.

# Explanation

This is a Java file that validates the data provided when updating a user's information. It checks that the first name, last name, email, avatar, practice, and site are provided and meet certain criteria. For instance, the email must be valid and the avatar must be at least 1.

# **Function Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function Name** | **Description** | **Arguments** | **Dependencies** |
|  |  |  |  |