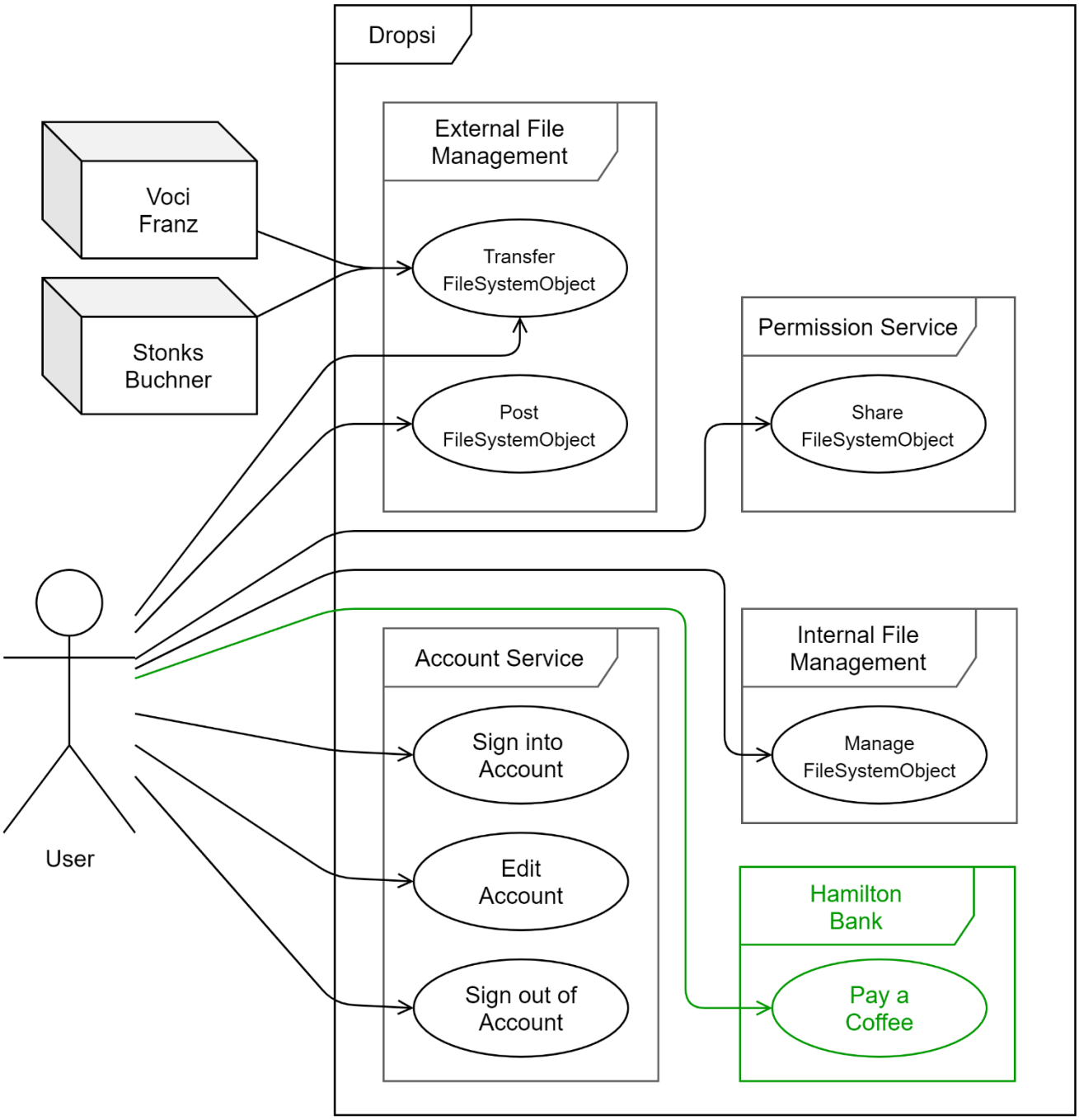
USE CASE DIAGRAM AND DETAILS



Submission: Meilenstein 1 (Update)

ACCOUNTSERVICE

Steps marked with ***[opt]*** are optional

Use Case name	**Sign into account**	
Actors	User	
Trigger	User Action	
Preconditions	The actor is registered	The actor is not registered
Basic flow of events	1. The actor enters his login credentials and logs in	1. The actor fills out the sign-up form and signs in
Exceptions	1a. The given information is invalid • The system informs the actor and lets him retry	1a. The given information is invalid • The system informs the actor and lets him retry
Postconditions	The actor is signed in	The account is persisted in the database The actor is signed in

Use Case name	**Edit account**
Actors	User
Trigger	User Action
Preconditions	The actor is registered The actor is signed in
Basic flow of events	1. **[opt]** The actor edits his account information 2. **[opt]** The actor links his account to Retrogram via a token
Exceptions	1a. The given information is not valid • The system informs the actor and lets him retry ~~2a. The given token is false~~ ~~• The system informs the actor the given token is invalid~~
Postconditions	The changes are persisted in the database

Checking if a token is valid via a REST interface would make it possible to acquire tokens via brute force. Therefore, we decided that it is the user's responsibility to check if the token is correct. A wrong token will show an error message.

Use Case name	**Sign out of account**
Actors	User
Trigger	User Action
Preconditions	The actor is registered The actor is signed in
Basic flow of events	1. **[opt]** The actor deletes his account 2. The actor signs out
Exceptions	~~1a. The actor cannot verify himself~~ • ~~The system informs the actor and lets him retry~~
Postconditions	The actor is signed out If the account was deleted the changes are persisted in the database

The user does no longer have to identify themselves.

PERMISSIONSERVICE

Use Case name	**Share FileSystemObject**	
Actors	User	
Trigger	User Action	
Preconditions	The actor is registered The actor is signed in The actor owns a FileSystemObject other than the root-folder	The actor is registered The actor is signed in The actor has been given access to a shared FileSystemObject
Basic flow of events	1. The actor chooses a FileSystemObject other than the root-folder 2. The actor gives and / or removes permissions to other accounts	1. The actor chooses a FileSystemObject which has been shared with him 2. **[opt]** The actor deletes the permission
Exceptions	-	-
Postconditions	The required permissions are updated in the database The required AccessLogEntries where generated and persisted in the database	**[opt]** The attached permission to the FileSystemObject is deleted in the database The required AccessLogEntries where generated and persisted in the database

INTERNAL FILE MANAGEMENT

Use Case name	**Manage FileSystemObject**
Actors	User
Trigger	User Action
Preconditions	The actor is registered The actor is signed in The actor owns a FileSystemObject other than the root-folder
Basic flow of events	1. The actor chooses a FileSystemObject other than the root folder 2. **[opt]** The actor chooses a location to move the FileSystemObject to 3. **[opt]** The actor chooses a location to copy the FileSystemObject to 4. **[opt]** The actor deletes the FileSystemObject
Exceptions	2a/3a. The chosen location contains a FileSystemObject of the same type and name • ~~The system informs the actor that two FileSystemObjects of the same type and name are going to be in the same location~~ • The FileSystemObject is renamed to "*filename (x)*" with x being a number ~~3b. The copied File had permissions attached~~ • ~~The system informs the actor that permissions are not also copied~~ 4a. The chosen FileSystemObject is a folder • ~~The system informs the actor that all contents of the folder will also be deleted~~ • The actor deletes the folder, and all contained files are also deleted
Postconditions	All changed, copied or deleted FileSystemObjects are updated and persisted in the database Attached permissions to FileSystemObjects are updated in the database The required AccessLogEntries where generated and persisted in the database

Having dynamic popups in Spring MVC and Thymeleaf was difficult to achieve. Therefor most Modals which where supposed to inform the users where removed.

EXTERNAL FILE MANAGEMENT

Use Case name	**Transfer FileSystemObject**	
Actors	User, Voci Franz, Stonks Buchner	
Trigger	User Action	
Preconditions	The user is registered The user is signed in	The user is registered The user is signed in The user owns a FileSystemObject other than the root-folder or has been shared a FileSystemObject
Basic flow of events	1. The actor navigates to the location he wants to upload to 2. The actor chooses a FileSystemObject from his local Filesystem and uploads it to the cloud Filesystem	1. The actor chooses a FileSystemObject to download 2. The chosen FileSystemObject is downloaded to the actors Filesystem
Exceptions		2a. The chosen FileSystemObject was a folder • The Downloaded File will be a .zip archive
Postconditions	The FileSystemObject was persisted in the database The required AccessLogEntries where generated and persisted in the database	The required AccessLogEntries where generated and persisted in the database

Use Case name	**Post FileSystemObject**
Actors	User
Trigger	User Action
Preconditions	The actor is registered The actor is signed in The actor owns a File which is of type Image or has been shared such a file ~~The actor has a linked Retrogram account~~
Basic flow of events	1. The actor chooses a File of type image 2. The image is posted to the linked Retrogram account
Exceptions	
Postconditions	The image is posted to Retrogram The required AccessLogEntries where generated and persisted in the database

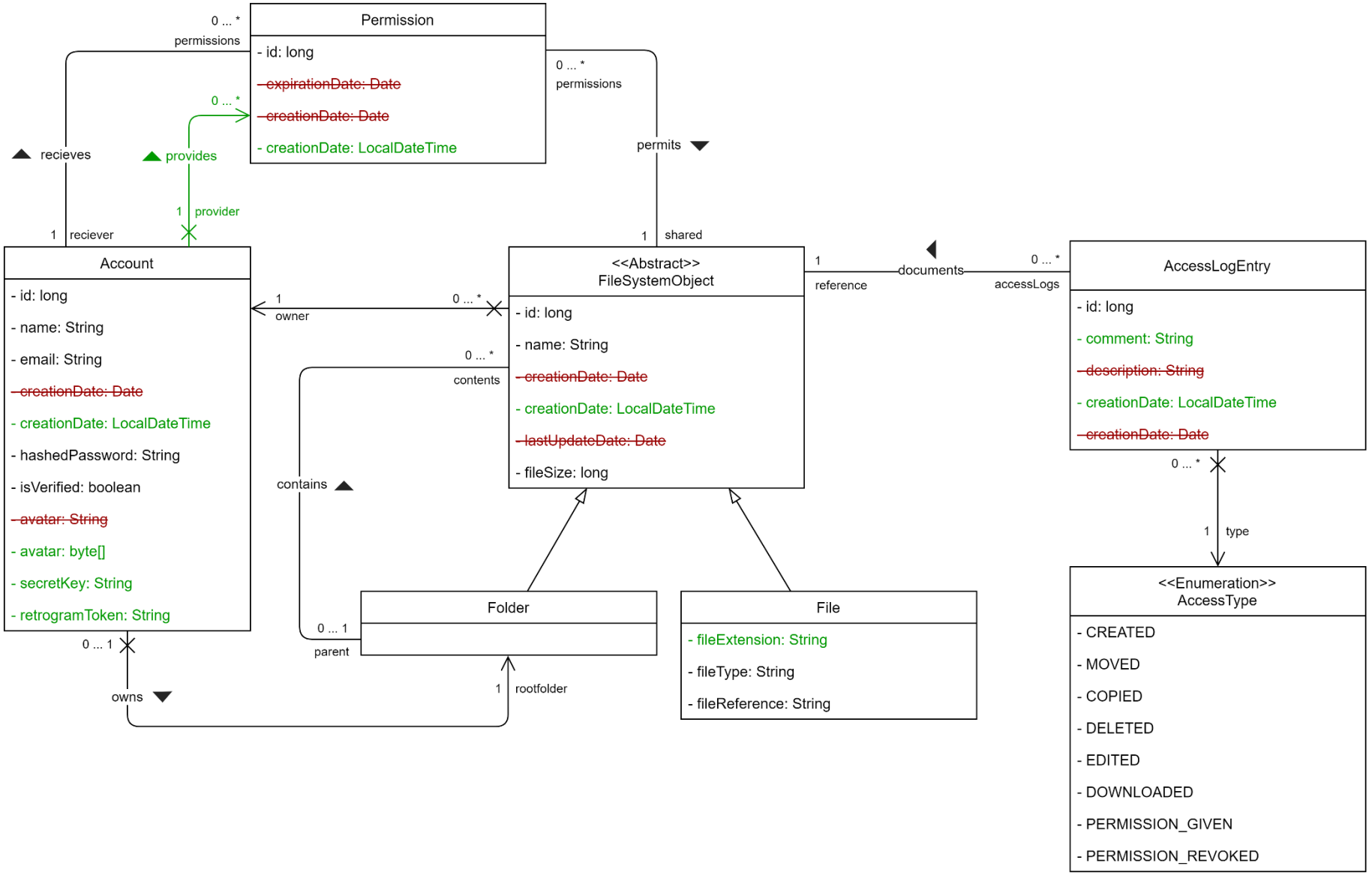
As explained above it suffices to have entered a code, not necessarily being a correct Retrogram token.

HAMILTON BANK

Use Case name	**Buy a coffee**
Actors	User
Trigger	User Action
Preconditions	The actor is registered The actor is signed in
Basic flow of events	1. The Actor Buys a coffee using the Hamilton bank interface 2. A Thank you message is displayed
Exceptions	If the Hamilton bank service is not available this option is not available
Postconditions	If the user accepted the payment the money is in the linked dropsi bank account The thank you message is displayed until the next time the user logs in

Submission: Meilenstein 1 (Update)

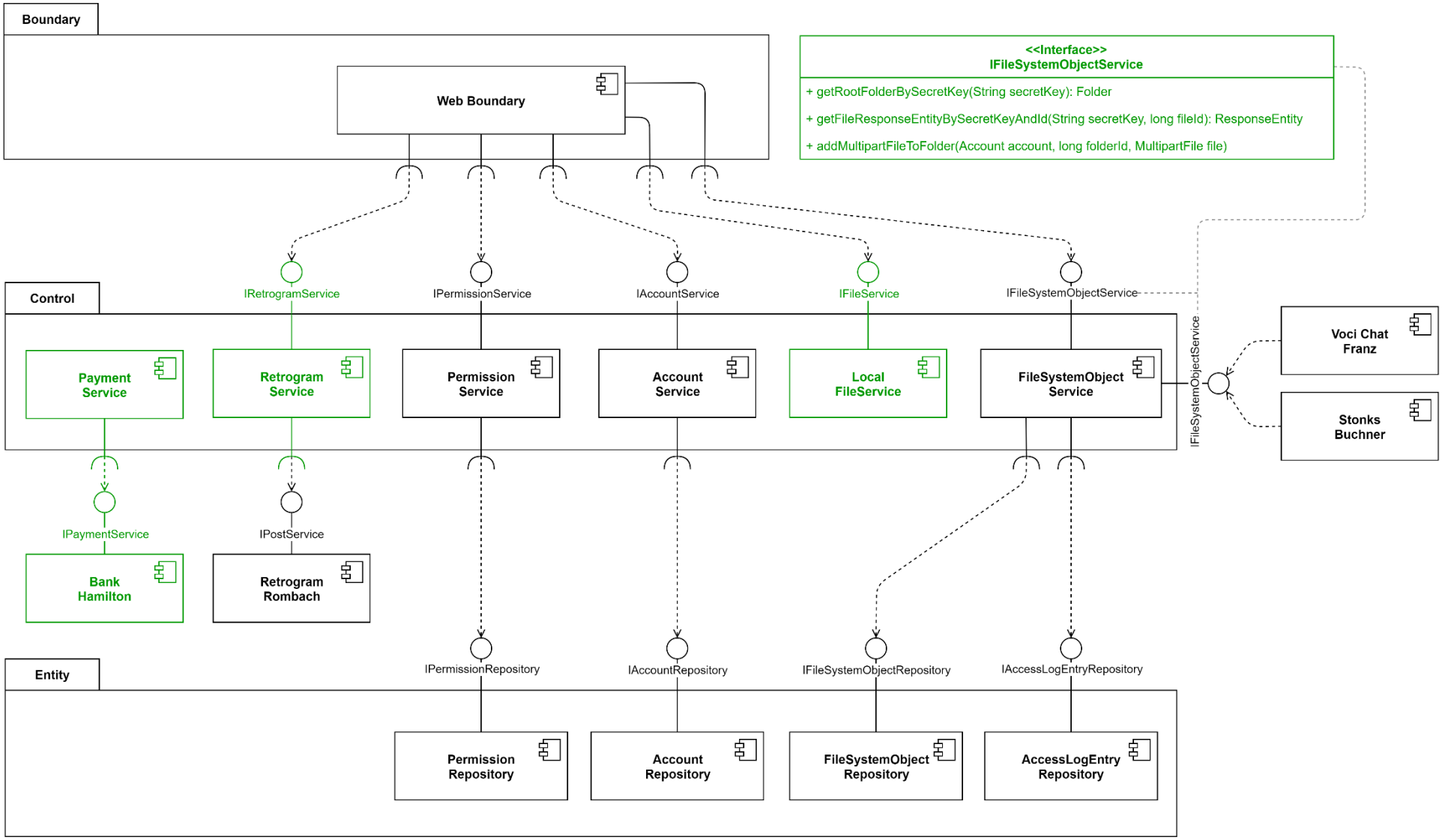
DOMAIN MODEL



Mathias Schöttle: Dropsi, MtrNr: 3147337

Submission: Meilenstein 1 (Update)

COMPONENT DIAGRAM



Removed thrown Exceptions for visibility

Mathias Schöttle: Dropsi, MtrNr: 3147337