CLASS DIAGRAM

REPOSITORY

<<Interface>> GenreRepository

+ findByTitle(title: String): List<Genre>
+ findAll(): List<Genre>
+ save(genre: Genre): Review
+ delete(genre: Genre): void

<<Interface>> GameRepository

+ findById(id: Long): Game
+ findAll(spec: Specification<Game>): List<Game>
+ findByTitle(title: String): Game
+ save(game: Game): Game
+ delete(game: Game): void

<<Interface>> ReviewRepository

+ findByGame(game: Game): List<Review>
+ findAll(): List<Review>
+ save(review: Review): Review
+ delete(review: Review): void

<<Interface>> UserRepository

+ findById(Id: Long): User
+ findAll(): List<User>
+ findByEmail(email: String): List<User>
+ findByUsername(username: String): List<User>
+ save(user: User): User
+ delete(user: User): void

SERVICE

@Service GenreServiceImplementation

- genreRepository: GenreRepository

+ getAllGenres(): Genre
+ getGenreByTitle(): List<Genre>
+ addGenre(genre: Genre): Genre
+ updateGenre(genreID: Long, genre: Genre): Genre
+ deleteGenre(genre: Genre): void
+ addGameToGenre(genre: Genre, game: Game): Game

@Service GameServiceImplementation

- gameRepository: GameRepository

+ findById(id: Long): Game
+ findAll(): List<Game>
+ findByTitle(title: String): Game
+ save(game: Game): Game
+ update(id: Long, game: Game): Game
+ delete(game: Game): void
+ search(criteria: SearchCriteria): List<Game>

@Service ReviewServiceImplementation

- reviewRepository: ReviewRepository

+ findByGame(game: Game): List<Review>
+ save(review: Review): Review
+ delete(review: Review): Void
+ update(review: Review, id: Long): Review
+ findById(id: Long): Review

@Service UserServiceImplementation

- userRepository: UserRepository

+ findById(id: Long): User
+ findAll(): List<User>
+ findByName(title: String): User
+ save(user: User): User
+ delete(user: User): void
+ updateUser(id: Long, user: User): User

@Service AuthServiceImplementation

- userRepository: UserRepository

+ save(user: User): User

<<Interface>> GenreService

+ getAllGenres(): Genre
+ getGenreByTitle(): List<Genre>
+ addGenre(genre: Genre): Genre
+ deleteGenre(genre: Genre): void
+ addGameToGenre(genre: Genre, game: Game): Game

<<Interface>> GameService

+ findById(id: Long): Game
+ findAll(): List<Game>
+ findByTitle(title: String): Game
+ save(game: Game): Game
+ update(id: Long, game: Game): Game
+ delete(game: Game): void
+ search(criteria: SearchCriteria): List<Game>

<<Interface>> ReviewService

+ findByGame(game: Game): List<Review>
+ save(review: Review): Review
+ delete(review: Review): void
+ update(review: Review, id: Long): Review
+ findById(id: Long): Review

<<Interface>> UserService

+ findById(id: Long): User
+ findAll(): List<User>
+ findByName(title: String): User
+ save(user: User): User
+ delete(user: User): void
+ updateUser(id: Long, user: User): User

<<Interface>> AuthService

+ save(user: User): User

CONTROLLER

@Controller GenreController

- genreService: GenreService

+ getAllGenres(): List<Genre>
+ getGenreByTitle(title: String): List<Genre>
+ createGenre(genre: Genre): Genre
+ updateGenre(id: Long, genre: Genre): Genre
+ deleteGenre(id: Long): void

@Controller GameController

- gameService: GameService
- reviewService: ReviewService
- genreService: GenreService

+ getGameById(id: Long): Game
+ getAllGames(): List<Game>
+ getGameByTitle(title: String): List<Game>
+ createGame(game: Game): Game
+ updateGame(id: Long, game: Game): Game
+ deleteGame(id: Long): void
+ searchGameFilter(title: String,minPrice: Float, maxPrice: Float, releaseDateFrom: String, releaseDateTo: String, developer: String, publisher: String, genres: List<String>): List<Game>
+ getReviews(game: Game): List<Review>

@Controller UserController

- userService: UserService
- reviewService: ReviewService
- gameService: GameService

+ getUserByID(id: long): User
+ getAllUsers(): List<User>
+ getUserByName(name: string): List<User>
+ addFavouriteGames(gameID: Long, userID: Long): Boolean
+ getFavouriteGames(userID: Long): List<Game>
+ deleteFavouriteGames(gameID: Long, userID: Long): Boolean
+ addWantsToPlayGames(gameID: Long, userID: Long): Boolean
+ getWantsToPlayGames(userID: Long): List<Game>
+ deleteWantsToPlayGames(gameID: Long, userID: Long): Boolean
+ addHasPlayedGames(gameID: Long, userID: Long): Boolean
+ getHasPlayedGames(userID: Long): List<Game>
+ deleteHasPlayedGames(gameID: Long, userID: Long): Boolean
+ updateUser(id: Long, user: User): User
+ deleteUser(id: Long): void
+ postReview(review: Review): Review
+ changeReview(reviewID: Long, review: Review): Review
+ deleteReview(reviewID: Long): Boolean
+ follow(followerID: Long, followingID: Long): Boolean
+ unFollow(followerID: Long, followingID: Long): Boolean
+ getFollowers(userID: Long): List<User>
+ getFollowing(userID: Long): List<User>

@Controller AuthController

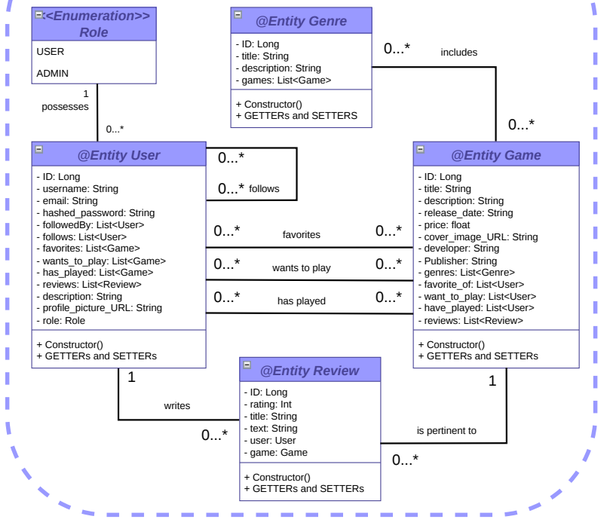
- loginService: LoginService

+ createAccount(email: String, password: String): User
+ logUserIn(email: String, password: String): User
+ logUserOut(id: Long): void
+ changePassword(newPass: String, oldPass: String, userId: Long): Boolean

About Method Color

Black: Any user can call this
Blue: Authenticated/logged-in users can call this
Red: Only admins can call this

ENTITIES



DTO (Data Transfer Object)

SearchCriteria

- title: String
- minPrice: float
- maxPrice: float
- releaseDateFrom: String
- releaseDateTo: String
- developer: String
- publisher: String
- genres: List<Genre>

+ Constructor()
+ Getters and Setters

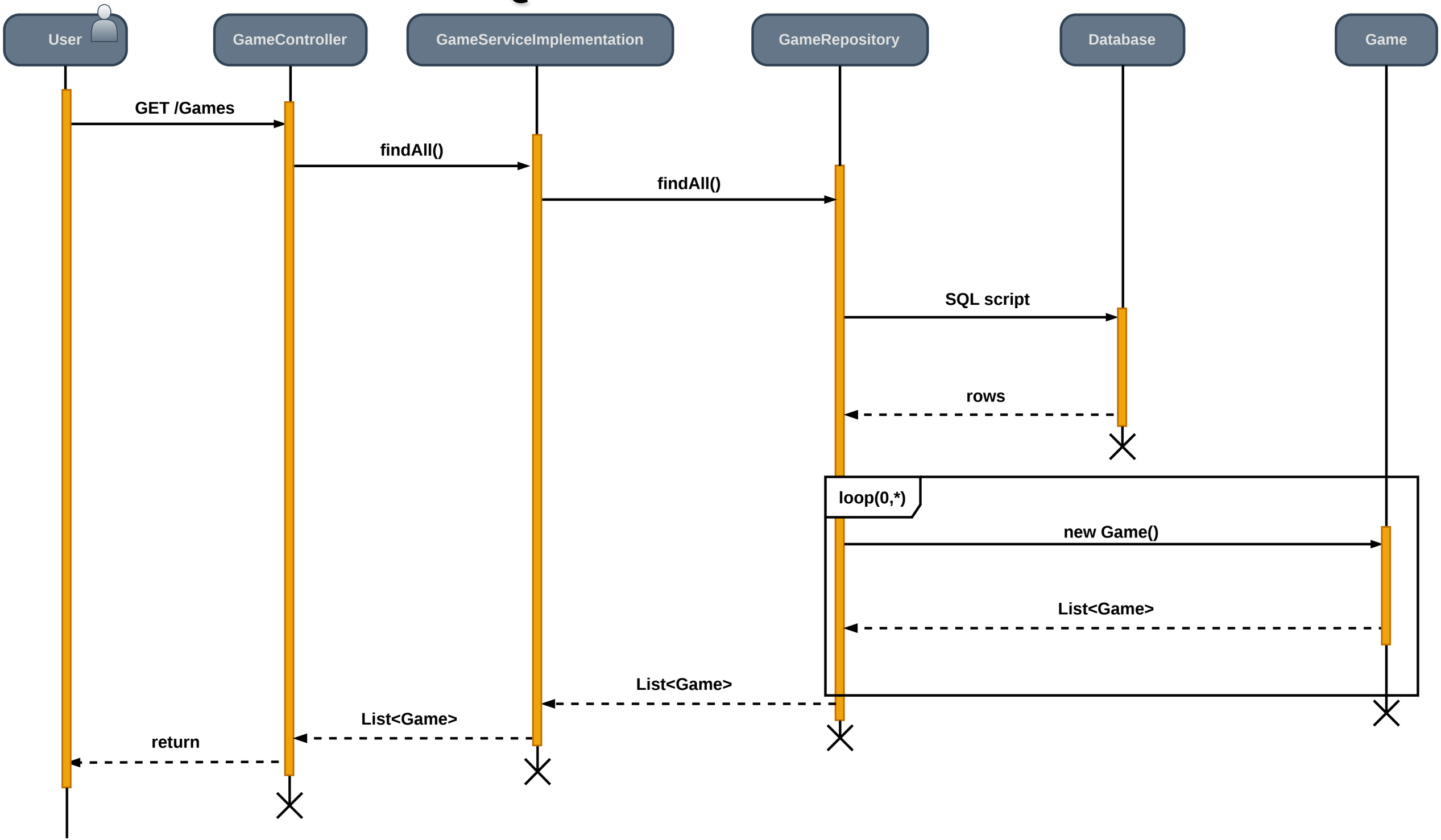
PaginatedResponse

- status: int
- data: List<Any>
- total: int
- pageNr: int
- perPage: int

+ Constructor()
+ Getters and Setters

Credentials

- email: String
- password: String

+ Constructor()
+ Getters and Setters

SEQUENCE DIAGRAM