UserFacade

Attributes
- + user: User
- + userRepository: UserRepository
- + artManager: ArtManager
- + walletManager: WalletManager
- + userCreator: CreateUser
- + userChanger: ChangeUser
- + logInner: LogIn

Operations
- + UserFacade(User, UserRepository, WalletManager, ArtManager):
- + getWallets(): List<Wallet>
- + changePassword(String, String): void
- + login(String, String): boolean
- + getWalletById(UUID): WalletFacade
- + addWallet(String, boolean): void
- + getWalletManager(): WalletManager
- + register(String, String): void
- + getUsername(): String
- + getEventsByType(String): List<Entry<LocalDateTime, String>>
- + getArtManager(): ArtManager
- + createAdminFacade(): AdminFacade
- + getNumberOfWallets(): int
- + logOut(): void
- + getIsAdmin(): boolean
- + getTradeableWallets(): List<WalletFacade>
- + getTotalNetWorth(): double
- + getAllEvents(): List<Entry<LocalDateTime, String>>

User

Attributes
- + isLoggedIn: boolean
- + password: String
- + events: List<Entry<LocalDateTime, String>>
- + username: String
- + wallets: List<Wallet>
- + isAdmin: boolean

Operations
- + User(String, String, boolean):
- + addWallet(Wallet): void
- + getEvents(String): List<Entry<LocalDateTime, String>>
- + getFirstWallet(): Wallet
- + removeWallet(Wallet): void
- + getPassword(): String
- + getIsLoggedIn(): boolean
- + logEvent(String, LocalDateTime): void
- + logEvent(String): void
- + getEvents(LocalDateTime, LocalDateTime): List<Entry<LocalDateTime, String>>
- + validate(String): boolean
- + getUsername(): String
- + setPassword(String): boolean
- + getIsAdmin(): boolean
- + setUsername(String): void
- + getWallets(): List<Wallet>
- + setLoggedIn(boolean): void
- + getEvents(): List<Entry<LocalDateTime, String>>

<<interface>> IBannableUser

Attributes

Operations
- + getIsTempBanned(): boolean
- + getTempBannedUntil(): LocalDateTime
- + setTempBannedUntil(LocalDateTime): void

AdminFacade

Attributes
- + userCreater: CreateUser
- + userBanner: BanUser
- + userFinder: FindUser

Operations
- + AdminFacade(AdminUser, UserRepository, WalletManager, ...):
- + banUser(String, LocalDateTime): void
- + getAllUsers(): List<UserFacade>
- + deleteUser(String): void
- + createUser(String, String, boolean): void

BasicUser

Attributes
- + tempBannedUntil: LocalDateTime

Operations
- + BasicUser(String, String, LocalDateTime):
- + BasicUser(String, String):
- + setTempBannedUntil(LocalDateTime): void
- + getTempBannedUntil(): LocalDateTime
- + getIsTempBanned(): boolean

AdminUser

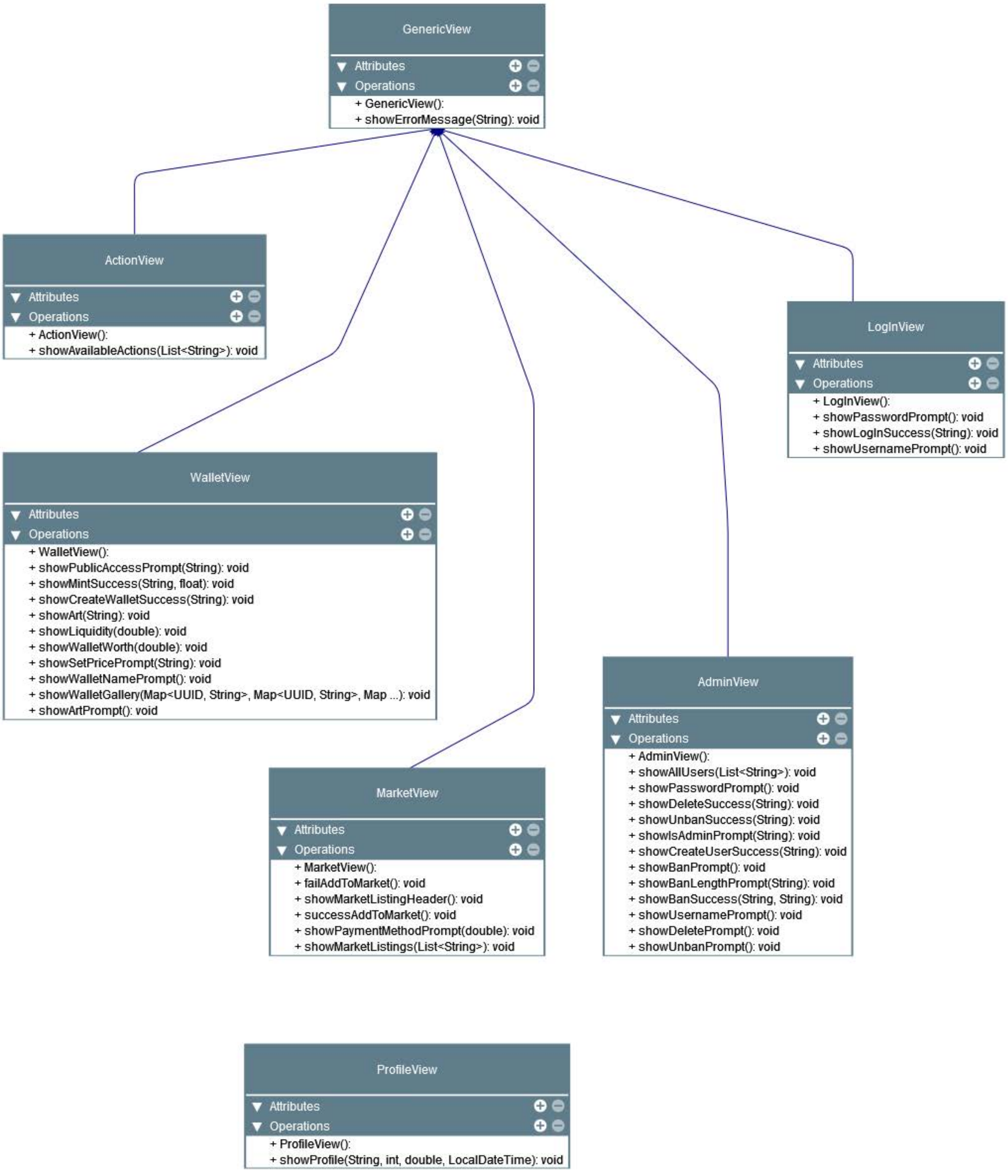
Attributes

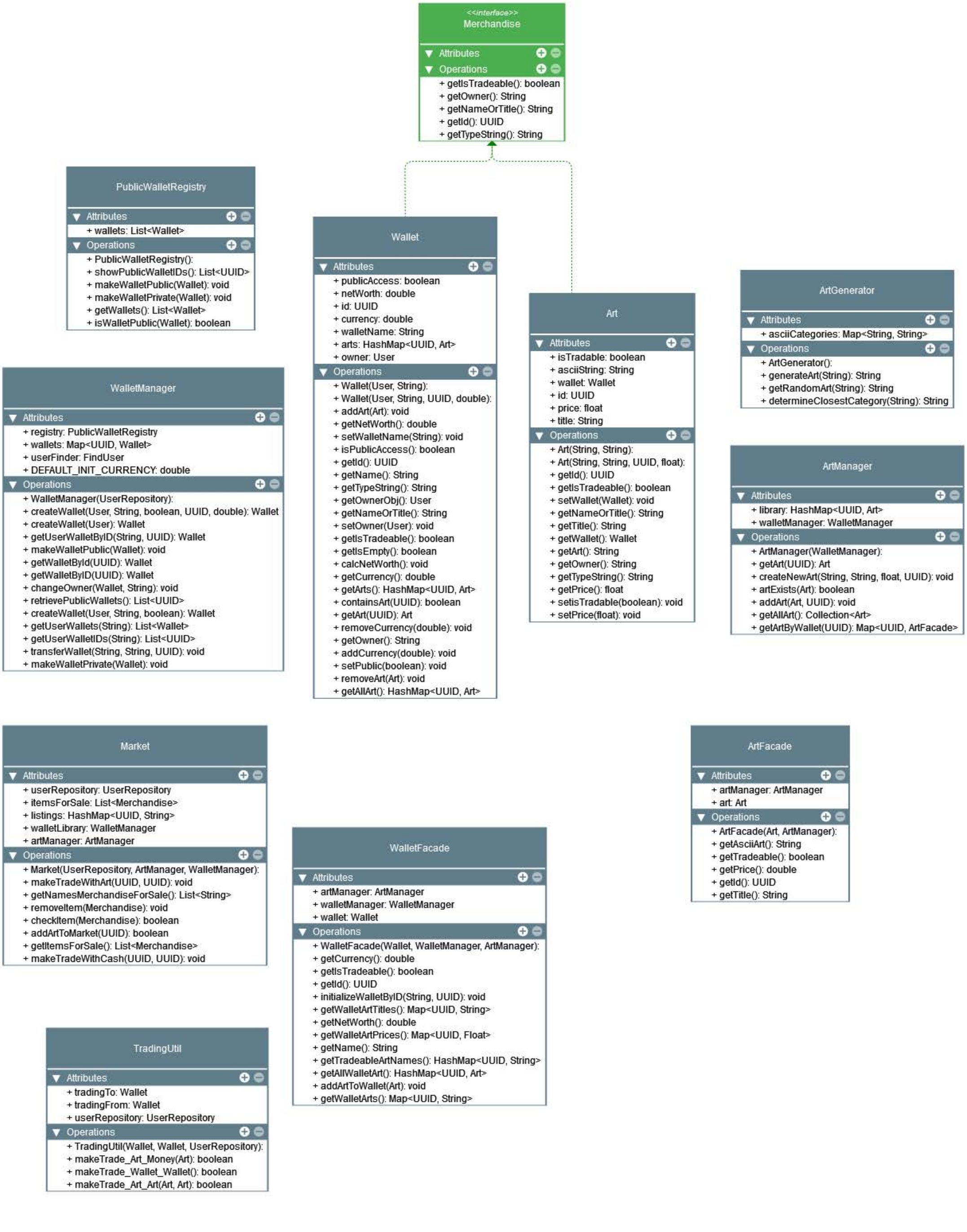
Operations
- + AdminUser(String, String):

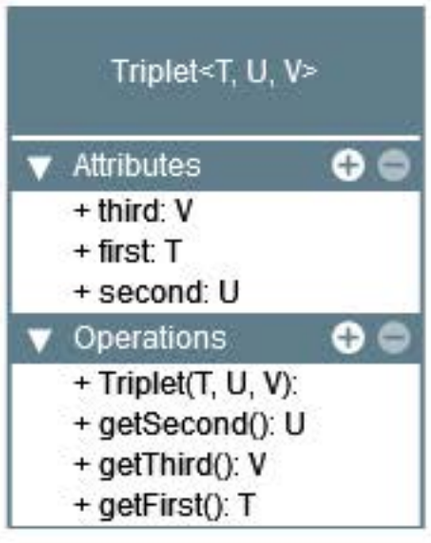
UserRepository

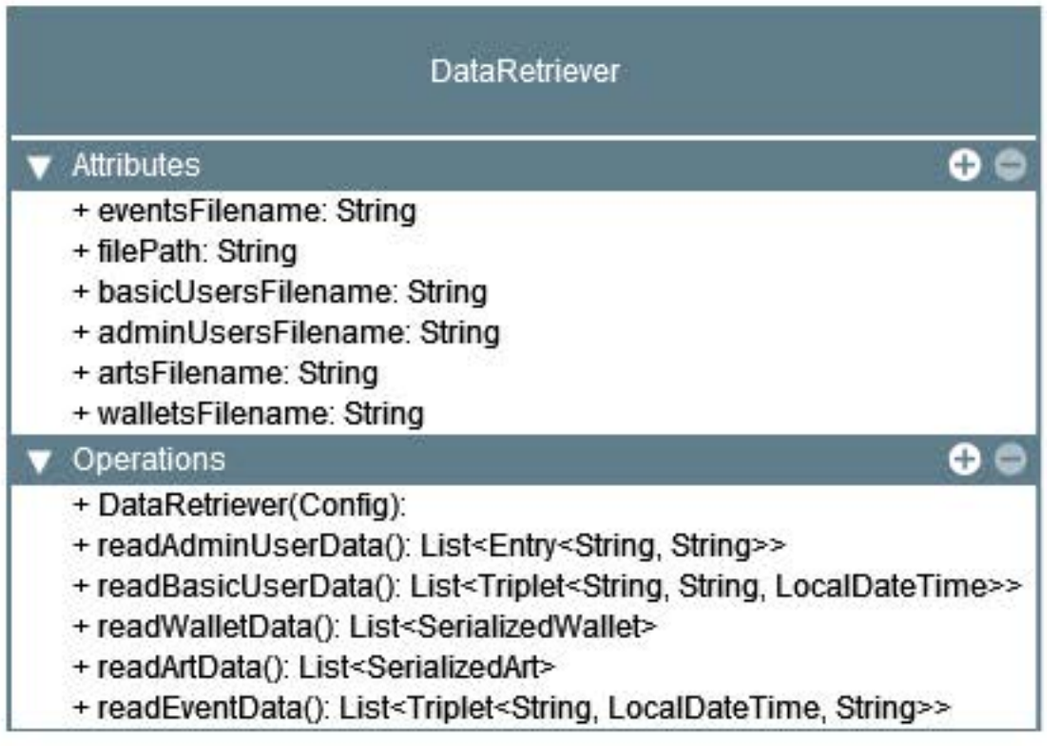
Attributes
- + users: HashMap<String, User>

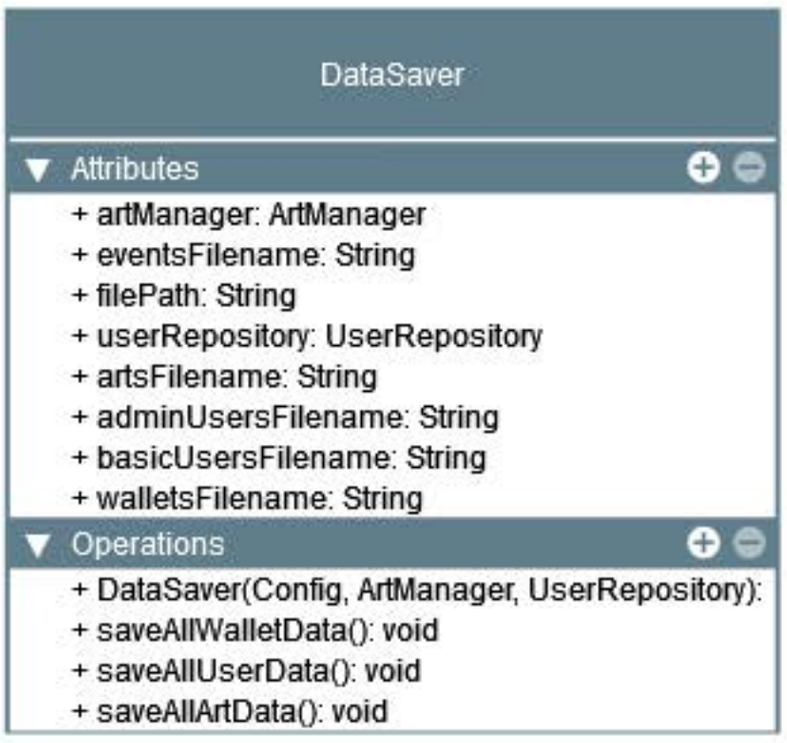
Operations
- + UserRepository():
- + getByUsername(String): Optional<User>
- + resetUserData(List<Entry<String, String>>, List<Triplet< ...): void
- + createUser(User): void
- + removeUser(User): void
- + getAllUsers(): List<User>
- + getAllUsersByType(boolean): List<User>


GenericView
Attributes
Operations
+ GenericView():
+ showErrorMessage(String): void
ActionView
Attributes
Operations
+ ActionView():
+ showAvailableActions(List<String>): void
LogInView
Attributes
Operations
+ LogInView():
+ showPasswordPrompt(): void
+ showLogInSuccess(String): void
+ showUsernamePrompt(): void
WalletView
Attributes
Operations
+ WalletView():
+ showPublicAccessPrompt(String): void
+ showMintSuccess(String, float): void
+ showCreateWalletSuccess(String): void
+ showArt(String): void
+ showLiquidity(double): void
+ showWalletWorth(double): void
+ showSetPricePrompt(String): void
+ showWalletNamePrompt(): void
+ showWalletGallery(Map<UUID, String>, Map<UUID, String>, Map ...): void
+ showArtPrompt(): void
AdminView
Attributes
Operations
+ AdminView():
+ showAllUsers(List<String>): void
+ showPasswordPrompt(): void
+ showDeleteSuccess(String): void
+ showUnbanSuccess(String): void
+ showIsAdminPrompt(String): void
+ showCreateUserSuccess(String): void
+ showBanPrompt(): void
+ showBanLengthPrompt(String): void
+ showBanSuccess(String, String): void
+ showUsernamePrompt(): void
+ showDeletePrompt(): void
+ showUnbanPrompt(): void
MarketView
Attributes
Operations
+ MarketView():
+ failAddToMarket(): void
+ showMarketListingHeader(): void
+ successAddToMarket(): void
+ showPaymentMethodPrompt(double): void
+ showMarketListings(List<String>): void
ProfileView
Attributes
Operations
+ ProfileView():
+ showProfile(String, int, double, LocalDateTime): void


<<interface>>
Merchandise
Attributes
Operations
+ getIsTradeable(): boolean
+ getOwner(): String
+ getNameOrTitle(): String
+ getId(): UUID
+ getTypeString(): String
PublicWalletRegistry
Attributes
+ wallets: List<Wallet>
Operations
+ PublicWalletRegistry():
+ showPublicWalletIDs(): List<UUID>
+ makeWalletPublic(Wallet): void
+ makeWalletPrivate(Wallet): void
+ getWallets(): List<Wallet>
+ isWalletPublic(Wallet): boolean
Wallet
Attributes
+ publicAccess: boolean
+ netWorth: double
+ id: UUID
+ currency: double
+ walletName: String
+ arts: HashMap<UUID, Art>
+ owner: User
Operations
+ Wallet(User, String):
+ Wallet(User, String, UUID, double):
+ addArt(Art): void
+ getNetWorth(): double
+ setWalletName(String): void
+ isPublicAccess(): boolean
+ getId(): UUID
+ getName(): String
+ getTypeString(): String
+ getOwnerObj(): User
+ getNameOrTitle(): String
+ setOwner(User): void
+ getIsTradeable(): boolean
+ getIsEmpty(): boolean
+ calcNetWorth(): void
+ getCurrency(): double
+ getArts(): HashMap<UUID, Art>
+ containsArt(UUID): boolean
+ getArt(UUID): Art
+ removeCurrency(double): void
+ getOwner(): String
+ addCurrency(double): void
+ setPublic(boolean): void
+ removeArt(Art): void
+ getAllArt(): HashMap<UUID, Art>
Art
Attributes
+ isTradable: boolean
+ asciiString: String
+ wallet: Wallet
+ id: UUID
+ price: float
+ title: String
Operations
+ Art(String, String):
+ Art(String, String, UUID, float):
+ getId(): UUID
+ getIsTradeable(): boolean
+ setWallet(Wallet): void
+ getNameOrTitle(): String
+ getTitle(): String
+ getWallet(): Wallet
+ getArt(): String
+ getOwner(): String
+ getTypeString(): String
+ getPrice(): float
+ setisTradable(boolean): void
+ setPrice(float): void
ArtGenerator
Attributes
+ asciiCategories: Map<String, String>
Operations
+ ArtGenerator():
+ generateArt(String): String
+ getRandomArt(String): String
+ determineClosestCategory(String): String
WalletManager
Attributes
+ registry: PublicWalletRegistry
+ wallets: Map<UUID, Wallet>
+ userFinder: FindUser
+ DEFAULT_INIT_CURRENCY: double
Operations
+ WalletManager(UserRepository):
+ createWallet(User, String, boolean, UUID, double): Wallet
+ createWallet(User): Wallet
+ getUserWalletByID(String, UUID): Wallet
+ makeWalletPublic(Wallet): void
+ getWalletById(UUID): Wallet
+ getWalletByID(UUID): Wallet
+ changeOwner(Wallet, String): void
+ retrievePublicWallets(): List<UUID>
+ createWallet(User, String, boolean): Wallet
+ getUserWallets(String): List<Wallet>
+ getUserWalletIDs(String): List<UUID>
+ transferWallet(String, String, UUID): void
+ makeWalletPrivate(Wallet): void
ArtManager
Attributes
+ library: HashMap<UUID, Art>
+ walletManager: WalletManager
Operations
+ ArtManager(WalletManager):
+ getArt(UUID): Art
+ createNewArt(String, String, float, UUID): void
+ artExists(Art): boolean
+ addArt(Art, UUID): void
+ getAllArt(): Collection<Art>
+ getArtByWallet(UUID): Map<UUID, ArtFacade>
Market
Attributes
+ userRepository: UserRepository
+ itemsForSale: List<Merchandise>
+ listings: HashMap<UUID, String>
+ walletLibrary: WalletManager
+ artManager: ArtManager
Operations
+ Market(UserRepository, ArtManager, WalletManager):
+ makeTradeWithArt(UUID, UUID): void
+ getNamesMerchandiseForSale(): List<String>
+ removeItem(Merchandise): void
+ checkItem(Merchandise): boolean
+ addArtToMarket(UUID): boolean
+ getItemsForSale(): List<Merchandise>
+ makeTradeWithCash(UUID, UUID): void
WalletFacade
Attributes
+ artManager: ArtManager
+ walletManager: WalletManager
+ wallet: Wallet
Operations
+ WalletFacade(Wallet, WalletManager, ArtManager):
+ getCurrency(): double
+ getIsTradeable(): boolean
+ getId(): UUID
+ initializeWalletByID(String, UUID): void
+ getWalletArtTitles(): Map<UUID, String>
+ getNetWorth(): double
+ getWalletArtPrices(): Map<UUID, Float>
+ getName(): String
+ getTradeableArtNames(): HashMap<UUID, String>
+ getAllWalletArt(): HashMap<UUID, Art>
+ addArtToWallet(Art): void
+ getWalletArts(): Map<UUID, String>
ArtFacade
Attributes
+ artManager: ArtManager
+ art: Art
Operations
+ ArtFacade(Art, ArtManager):
+ getAsciiArt(): String
+ getTradeable(): boolean
+ getPrice(): double
+ getId(): UUID
+ getTitle(): String
TradingUtil
Attributes
+ tradingTo: Wallet
+ tradingFrom: Wallet
+ userRepository: UserRepository
Operations
+ TradingUtil(Wallet, Wallet, UserRepository):
+ makeTrade_Art_Money(Art): boolean
+ makeTrade_Wallet_Wallet(): boolean
+ makeTrade_Art_Art(Art, Art): boolean

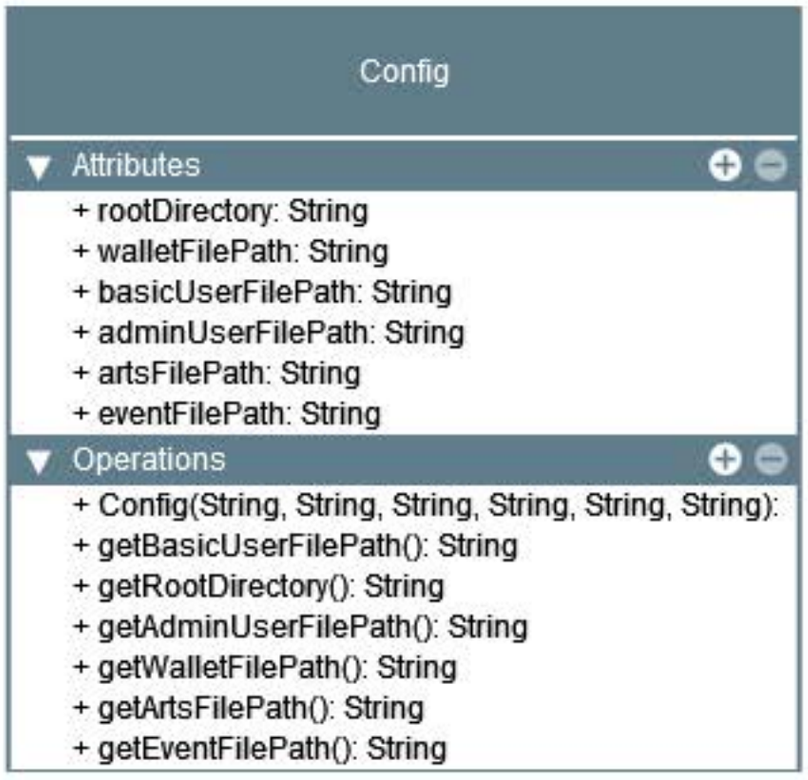

Triplet<T, U, V>
Attributes
+ third: V
+ first: T
+ second: U
Operations
+ Triplet(T, U, V):
+ getSecond(): U
+ getThird(): V
+ getFirst(): T

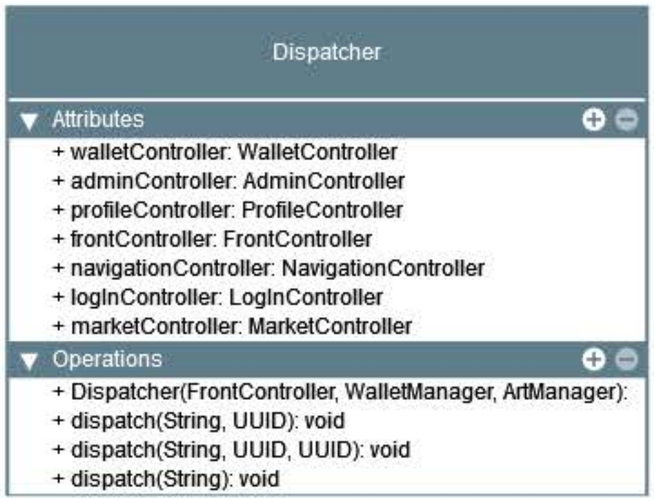

DataRetriever
Attributes
+ eventsFilename: String
+ filePath: String
+ basicUsersFilename: String
+ adminUsersFilename: String
+ artsFilename: String
+ walletsFilename: String
Operations
+ DataRetriever(Config):
+ readAdminUserData(): List<Entry<String, String>>
+ readBasicUserData(): List<Triplet<String, String, LocalDateTime>>
+ readWalletData(): List<SerializedWallet>
+ readArtData(): List<SerializedArt>
+ readEventData(): List<Triplet<String, LocalDateTime, String>>

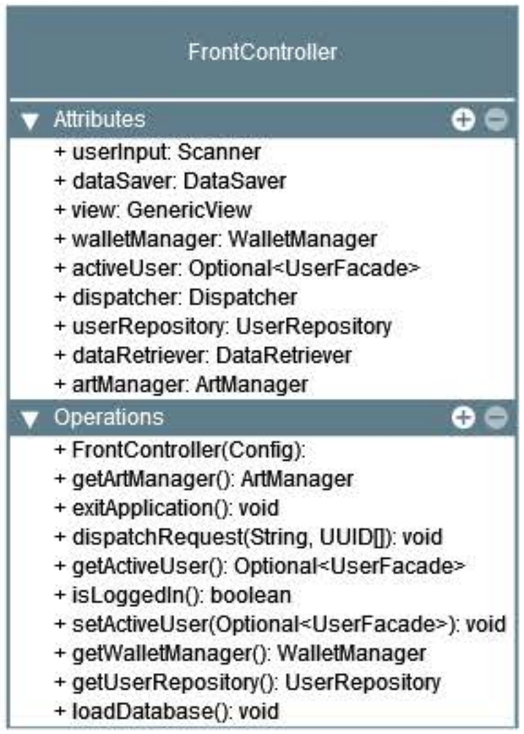

DataSaver
Attributes
+ artManager: ArtManager
+ eventsFilename: String
+ filePath: String
+ userRepository: UserRepository
+ artsFilename: String
+ adminUsersFilename: String
+ basicUsersFilename: String
+ walletsFilename: String
Operations
+ DataSaver(Config, ArtManager, UserRepository):
+ saveAllWalletData(): void
+ saveAllUserData(): void
+ saveAllArtData(): void

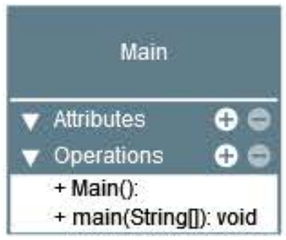

SerializedWallet
Attributes
+ walletName: String
+ ownerUsername: String
+ walletID: UUID
+ isTradeable: boolean
+ currency: double
Operations
+ SerializedWallet(String, String, String, String, String):
+ isTradeable(): boolean
+ getCurrency(): double
+ getWalletID(): UUID
+ getWalletName(): String
+ getOwnerUsername(): String

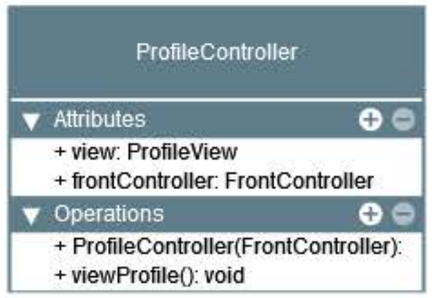

SerializedArt
Attributes
+ artTitle: String
+ walletID: UUID
+ art: String
+ price: float
+ artID: UUID
Operations
+ SerializedArt(String, UUID, UUID, String, float):
+ getArtID(): UUID
+ getArtTitle(): String
+ getPrice(): float
+ getWalletID(): UUID
+ getArt(): String


Config
Attributes
+ rootDirectory: String
+ walletFilePath: String
+ basicUserFilePath: String
+ adminUserFilePath: String
+ artsFilePath: String
+ eventFilePath: String
Operations
+ Config(String, String, String, String, String, String):
+ getBasicUserFilePath(): String
+ getRootDirectory(): String
+ getAdminUserFilePath(): String
+ getWalletFilePath(): String
+ getArtsFilePath(): String
+ getEventFilePath(): String

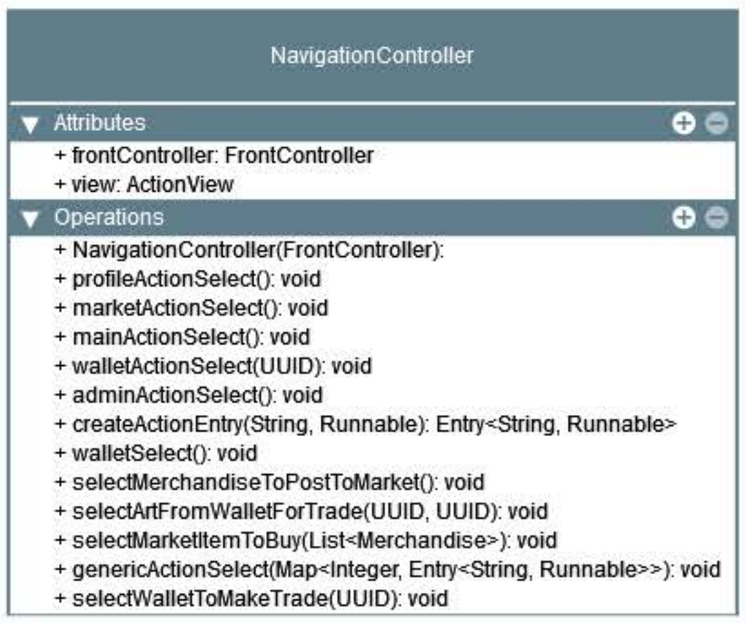

Dispatcher
Attributes
+ walletController: WalletController
+ adminController: AdminController
+ profileController: ProfileController
+ frontController: FrontController
+ navigationController: NavigationController
+ logInController: LogInController
+ marketController: MarketController
Operations
+ Dispatcher(FrontController, WalletManager, ArtManager):
+ dispatch(String, UUID): void
+ dispatch(String, UUID, UUID): void
+ dispatch(String): void

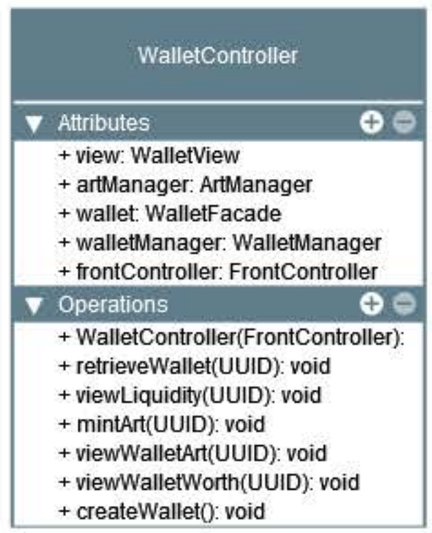

FrontController
Attributes
+ userInput: Scanner
+ dataSaver: DataSaver
+ view: GenericView
+ walletManager: WalletManager
+ activeUser: Optional<UserFacade>
+ dispatcher: Dispatcher
+ userRepository: UserRepository
+ dataRetriever: DataRetriever
+ artManager: ArtManager
Operations
+ FrontController(Config):
+ getArtManager(): ArtManager
+ exitApplication(): void
+ dispatchRequest(String, UUID[]): void
+ getActiveUser(): Optional<UserFacade>
+ isLoggedIn(): boolean
+ setActiveUser(Optional<UserFacade>): void
+ getWalletManager(): WalletManager
+ getUserRepository(): UserRepository
+ loadDatabase(): void


Main
Attributes
Operations
+ Main():
+ main(String[]): void

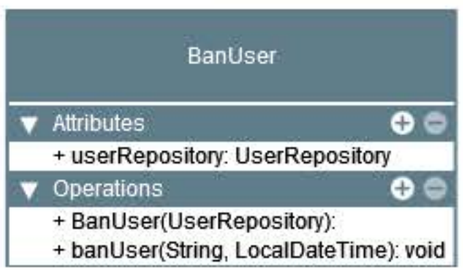

ProfileController
Attributes
+ view: ProfileView
+ frontController: FrontController
Operations
+ ProfileController(FrontController):
+ viewProfile(): void

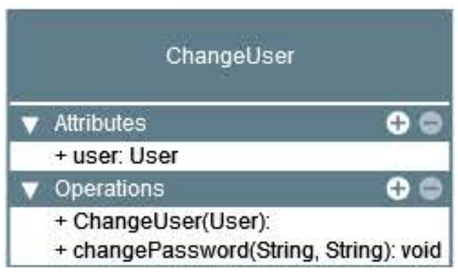

AdminController
Attributes
+ view: AdminView
+ frontController: FrontController
Operations
+ AdminController(FrontController):
+ seeAllUsers(): void
+ createUser(): void
+ unbanUser(): void
+ banUser(): void
+ deleteUser(): void
+ presentAllUsers(): void

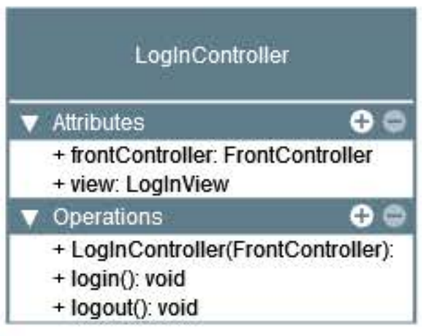

NavigationController
Attributes
+ frontController: FrontController
+ view: ActionView
Operations
+ NavigationController(FrontController):
+ profileActionSelect(): void
+ marketActionSelect(): void
+ mainActionSelect(): void
+ walletActionSelect(UUID): void
+ adminActionSelect(): void
+ createActionEntry(String, Runnable): Entry<String, Runnable>
+ walletSelect(): void
+ selectMerchandiseToPostToMarket(): void
+ selectArtFromWalletForTrade(UUID, UUID): void
+ selectMarketItemToBuy(List<Merchandise>): void
+ genericActionSelect(Map<Integer, Entry<String, Runnable>>): void
+ selectWalletToMakeTrade(UUID): void

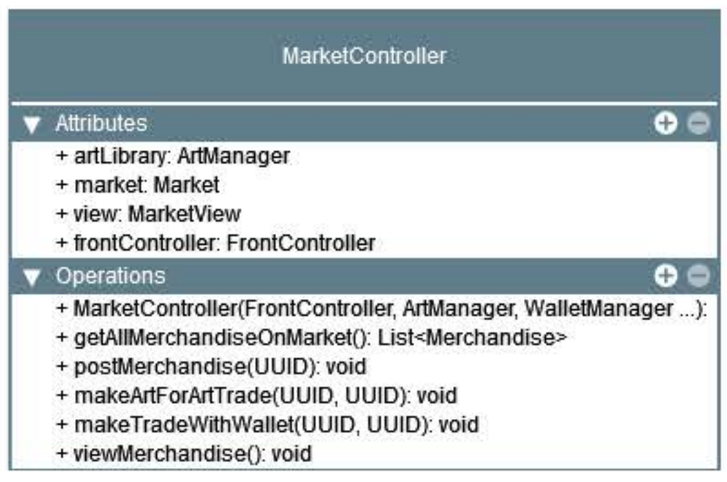

WalletController
Attributes
+ view: WalletView
+ artManager: ArtManager
+ wallet: WalletFacade
+ walletManager: WalletManager
+ frontController: FrontController
Operations
+ WalletController(FrontController):
+ retrieveWallet(UUID): void
+ viewLiquidity(UUID): void
+ mintArt(UUID): void
+ viewWalletArt(UUID): void
+ viewWalletWorth(UUID): void
+ createWallet(): void


CreateUser
Attributes
+ userRepository: UserRepository
+ walletManager: WalletManager
Operations
+ CreateUser(UserRepository, WalletManager):
+ createUser(String, String, boolean): void
+ deleteUser(String): void


BanUser
Attributes
+ userRepository: UserRepository
Operations
+ BanUser(UserRepository):
+ banUser(String, LocalDateTime): void

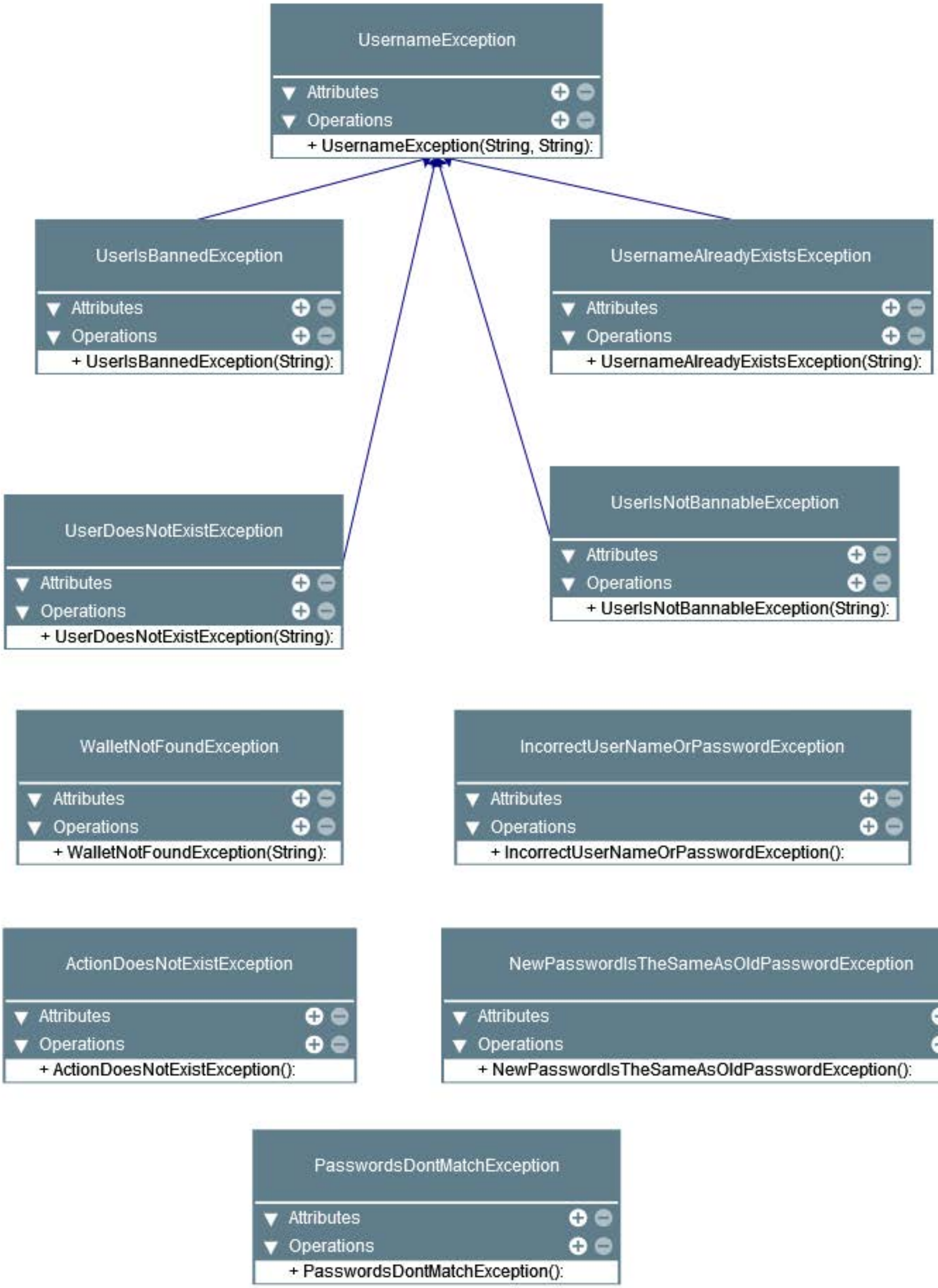

ChangeUser
Attributes
+ user: User
Operations
+ ChangeUser(User):
+ changePassword(String, String): void


LogInController
Attributes
+ frontController: FrontController
+ view: LogInView
Operations
+ LogInController(FrontController):
+ login(): void
+ logout(): void


MarketController
Attributes
+ artLibrary: ArtManager
+ market: Market
+ view: MarketView
+ frontController: FrontController
Operations
+ MarketController(FrontController, ArtManager, WalletManager ...):
+ getAllMerchandiseOnMarket(): List<Merchandise>
+ postMerchandise(UUID): void
+ makeArtForArtTrade(UUID, UUID): void
+ makeTradeWithWallet(UUID, UUID): void
+ viewMerchandise(): void


FindUser
Attributes
+ walletManager: WalletManager
+ userRepository: UserRepository
Operations
+ FindUser(UserRepository, WalletManager):
+ getUserByUsername(String): User


LogIn
Attributes
+ userRepository: UserRepository
Operations
+ LogIn(UserRepository):
+ logIn(String, String): User


UsernameException
Attributes
Operations
+ UsernameException(String, String):
UserIsBannedException
Attributes
Operations
+ UserIsBannedException(String):
UsernameAlreadyExistsException
Attributes
Operations
+ UsernameAlreadyExistsException(String):
UserDoesNotExistException
Attributes
Operations
+ UserDoesNotExistException(String):
UserIsNotBannableException
Attributes
Operations
+ UserIsNotBannableException(String):
WalletNotFoundException
Attributes
Operations
+ WalletNotFoundException(String):
IncorrectUserNameOrPasswordException
Attributes
Operations
+ IncorrectUserNameOrPasswordException():
ActionDoesNotExistException
Attributes
Operations
+ ActionDoesNotExistException():
NewPasswordIsTheSameAsOldPasswordException
Attributes
Operations
+ NewPasswordIsTheSameAsOldPasswordException():
PasswordsDontMatchException
Attributes
Operations
+ PasswordsDontMatchException():