viewModel

AfterQuestionScorePageViewModel

- repository: Repository

+ getScoreScreenContents(): List<Player>
+ getMyPlayerId(): String
+ isRoundOver(): boolean
+ isHotSwap(): boolean
+ isHost(): boolean
+ resetPlayersReady(): void
+ showNextView(): void

LotteryViewModel

- mapWinningsLiveData: MutableLiveData<Map<Player, Item>
- itemListLiveData: MutableLiveData<List<Item>>
- playerListLiveData: MutableLiveData<List<Player>>

+ onLotteryDrawEvent(LotteryDrawEvent): void

QuestionViewModel

- repository: Repository
- questionText: MutableLiveData<String>
- questionAlts: MutableLiveData<List<String>>
- questionTime: MutableLiveData<Integer>
- playerName: MutableLiveData<String>
- questionEvent: QuestionEvent
- isCorrectAnswer: boolean
- numOfRepeats: int

+ getQuestionText(): MutableLiveData<String>
+ getQuestionAlts(): MutableLiveData<List<String>>
+ getQuestionTime(): MutableLiveData<Integer>
+ nextQuestion(): void
+ receiveQuestion(QuestionEvent): void
+ onAnswerClicked(int, ObjectAnimator): void
+ showNextView(): void
+ isMoveOn(): boolean
+ repeatQuestion(): void

CategoryViewModel

- repository: Repository
- categories: Category[]
- votes: int[]
- totNmbVotes: int
- curNmbVotes: int

+ generateCategories(): void
+ updateCategories(): void
+ isMe(Player): boolean
+ isHotSwap(): boolean
+ isHost(): boolean
+ sendIsReady(): void
+ startCategoryPlaylist(Context): void
+ getMyPlayerId(): String
+ resetPlayerReady(): void
+ showNextView(): void
+ resetVote(): void
+ getCategoryIndex(String): int
+ onCategoryVoteEvent(String): void
+ voteCategory(int): void
+ getVoteResult(int): void

CreateLobbyNetViewModel

-repository: Repository

+ setupNetwork(Context): void
+ getRoomName(): String
+ setRoomName(String): void
+ getPlayerName(): String
+ setPlayerName(String): void

ChooseGameViewModel

+ resetApp(Context): void

SettingViewModel

+ turnOnMusic(): void
+ turnOffMusic(): void

HostCreateRoomViewModel

~ listForView: MutableLiveData: <List<String>>

+ getListForView(): MutableLiveData<List<String>>

HotSwapGameModeViewModel

- gameModes: MutableLiveData<List<String>>

+ getGameModes(): MutableLiveData<List<String>>

LobbyNetViewModel

- repository: Repository
- playerListForView: MutableLiveData<List<Pair<Player, Connection>>
- teamListForView: MutableLiveData<List<Team>>
- myPlayerId: MutableLiveData<String>
- lobbyName: MutableLiveData<String>
- isHost: boolean
- hasStartedGame: boolean

+ onTeamChangeEvent(TeamChangeEvent): void
+ onMyPlayerIdChangedEvent(MyPlayerIdChangedEvent): void
+ onLobbyNameChangeEvent(LobbyNameChangeEvent): void
+ restoreNetInCommunication(): void
+ startHostBeacon(): void
+ stopHostBeacon(): void
+ clearConnections(): void
+ startGame(): void
+ requestTeamChange(String): void
+ requestSetReady(boolean): void
+ fireTeamChangeEvent(): void
+ areAllPlayersReady(): boolean
+ removePlayer(Player): void

preGameCountdownViewModel

- context: Context
- toast: Toast
- repository: Repository

+ startToastMessage(): void
+ isHotSwap(): boolean
+ isHost(): boolean
+ sendIsReady(): void
+ resetPlayerReady(): void
+ showNextView(): void
+ finishToastMessage(): void

HotSwapAddPlayersViewModel

- playerListForView: MutableLiveData<List<Pair<Player, Integer>>>

+ getPlayerListForView(): MutableLiveData<List<Pair<Player, Integer>>
- loadPlayerList(): void
+ onTeamChangeEvent(TeamChangeEvent): void
+ addNewPlayer(): void
+ removePlayer(int): void
+ onTeamChange(int, int): void

TeamArrangementViewModel

- listForView: MutableLiveData<List<Pair<Player, Integer>>>

+ getListForView(): MutableLiveData<List<Pair<Player, Integer>>>
+ onTeamChangeEvent(TeamChangeEvent): void
+ addNewPlayer(): void
+ removePlayer(Player): void
+ resetPlayerData(): void
+ updatePlayerData(int, int): void
- onCleared(): void

JoinLobbyViewModel

-listForView: MutableLiveData: <List<Connection>>

+ getListForView(): MutableLiveData<List<String>>
+ setupNetwork(Context): void
+ joinRoom(Connection): void
+ startScan(): void
+ stopScan():void
+ cleanConnections(): void
+ getNewGeneratedPlayerName(): String
+ getPlayerName(): String
+ setPlayerName(String): void
+ setupNewGameInstance(): void

StoreViewModel

+ isItemBuyable(int): boolean
+ buy(int): void
+ getItem(int): Item
+ getPlayerPoints(): int
+ isItemBought(int): boolean