Équipe 304

Protocole de communication

Version 1.3

Historique des révisions

Date	Version	Description	Auteur
2023-03-18	1.0	Réponse à la partie 2 avec les fonctionnalités du sprint 1 et 2	Junaid Qureshi
2023-03-18	1.1	Réponse à la partie 3 avec les fonctionnalités du sprint 1 et 2	Junaid Qureshi
2023-03-19	1.2	Ajout des éléments du sprint 3 et ajout de l'introduction	Junaid Qureshi
2023-03-20	1.3	Révision du contenu	Pierre Tran

Table des matières

Protocole de communication

1. Introduction

Durant la conception d'une architecture client-serveur, les développeurs doivent tenir de plusieurs aspects fondamentaux, dont les communications entre plusieurs postes. Ces communications doivent être correctement implémentées pour assurer le fonctionnement de l'architecture. Voilà pourquoi, il est important de bien définir à partir de ce document. Dans l'ensemble, le contenu fournit une vue d'ensemble des protocoles de communication et des paquets de données utilisés dans l'architecture client-serveur. Ce contenu est divisé en deux sections.

La première section explique l'utilisation des requêtes HTTP et des sockets web dans une communication client-serveur. Elle décrit entre autres les fonctionnalités qui utilisent les requêtes HTTP telles que le chargement des niveaux, la mise à jour des constantes et la réinitialisation des scores. Elle explique également les fonctionnalités qui utilisent les sockets web, telles que la mise à jour de l'interface utilisateur en temps réel, la messagerie, la vérification des clics et la participation ou la création d'un jeu.

La deuxième section décrit les différents paquets utilisés dans le protocole WebSocket et le protocole HTTP. D'une part, pour le WS, cette section énumère le nom de l'événement, la direction du flux de données et les données envoyées ou reçues par le serveur et le client. D'autre part, en ce qui concerne les requêtes HTTP, elle énumère la méthode, la route, le corps, les paramètres et les données de retour.

2. Communication client-serveur

Les requêtes HTTP sont utilisées pour les fonctionnalités suivantes :

- Avant de charger la page de sélection ou la page de configuration, tous les niveaux de la base de données sont chargés dans la page. (GET)
- Avant de commencer un jeu, les informations relatives au niveau du serveur sont chargés GET)
- La création du jeu nécessite également cette technologie, car nous devons envoyer les informations du jeu au serveur. (POST)
- Mise à jour des constantes globales (POST)
- La remise à zéro du classement d'un niveau spécifique (DELETE)
- La réinitialisation des constantes globales (DELETE)

La raison pour laquelle nous avons utilisé des requêtes HTTP pour ces fonctionnalités est qu'aucun autre utilisateur n'est directement affecté par ces requêtes (côté UI et logique).

Nous avons utilisé des sockets web pour les fonctionnalités suivantes :

- Le chronomètre du jeu utilise des sockets web pour mettre à jour activement (à chaque seconde) l'interface utilisateur des joueurs dans un match.
- Le système de messagerie les utilise pour mettre à jour la boîte de dialogue lorsqu'un message est envoyé au serveur ou lorsque le serveur doit envoyer un message spécial.
- La vérification d'un clic est également gérée par des sockets web, car une fois le clic vérifié, l'autre joueur doit également être notifié. Cela permet aux deux joueurs d'être informés du nombre de différences trouvées.
- La suppression d'un niveau doit les utiliser car elle doit vérifier s'il y a actuellement des joueurs qui attendent une partie ou qui sont en jeu dans ce niveau. S'il en trouve, il doit en informer tous les joueurs concernés.
- La fonctionnalité consistant à rejoindre ou à créer un jeu repose aussi fortement sur les sockets web, car il y a une communication constante entre deux joueurs à la recherche d'une partie. Cela inclut l'annulation, le refus ou l'acceptation d'une demande de partie.
- L'abandon d'une partie utilise également des sockets web, car il s'agit de notifier à l'autre joueur que le match est terminé.
- La page de sélection et de configuration en a besoin pour qu'ils se mettent à jour si une page est supprimée.

- Le mode triche les utilisent pour pouvoir changer l'état du jeu et recevoir les données des différences.
- Le système d'indice l'utilise pour savoir quelles différences ont été trouvées.
- La sauvegarde d'un match se fait par des sockets, car c'est la passerelle WS qui détermine la fin d'un match.
- Les messages globaux doivent être gérés par les sockets car tous les joueurs doivent les recevoir.

3. Description des paquets

Protocole WebSocket:

- Nom de l'évènement: OnJoinNewGame
  Direction: Le client vers le serveur.
  Données: Le serveur reçoit un objet «data» qui contient:
    - levelId (number): Le numéro identifiant du niveau.
    - playerName (string): Le nom du joueur.

- Nom de l'évènement: onClick
  Direction: Le client vers le serveur.
  Données: position (number): La position du click sur le canvas.

- Nom de l'évènement: onGameSelection
  Direction: Le client vers le serveur.
  Données: Le serveur reçoit un objet «data» qui contient:
    - levelId (number): Le numéro identifiant du niveau.
    - playerName (string): Le nom du joueur.

- Nom de l'évènement: onGameAccepted
  Direction: Le client vers le serveur.
  Données: Aucune

- Nom de l'évènement: onGameCancelledWhileWaitingForSecondPlayer
  Direction: Le client vers le serveur.
  Données: Aucune

- Nom de l'évènement: onGameRejected
  Direction: Le client vers le serveur.
  Données: Aucune

- Nom de l'évènement: onDeleteLevel
  Direction: Le client vers le serveur.
  Données: levelId (number) Le numéro identifiant du niveau.

- Nom de l'évènement: onAbandonGame
  Direction: Le client vers le serveur.
  Données: Aucune
-

- Nom de l’évènement:onDemandForHint
  Direction: Le client vers le serveur.
  Données: Aucune

- Nom de l’évènement: processClick
  Direction: Le serveur vers le ou les clients d’une partie.
  Données: Le serveur envoie un objet «GameData» qui contient:
  - differencePixels (number[]): Le tableau de pixels qui appartiennent à la différence cliquée.
  - totalDifferences (number): Le nombre de différences de l’image.
  - amountOfDifferencesFound (number): Le nombre de différences trouvées par le joueur.
  - amountOfDifferencesFoundSecondPlayer (number, optionelle): Le nombre de différence trouvées par le deuxième joueur.

- Nom de l’évènement:onStartCheatMode
  Direction: Le client vers le serveur.
  Données: Aucune

- Nom de l’évènement:onStopCheatMode
  Direction: Le client vers le serveur.
  Données: Aucune

- Nom de l’évènement: victory
  Direction: Le serveur vers le client.
  Données: Aucune

- Nom de l’évènement: defeat
  Direction: Le serveur vers l’autre client.
  Données: Aucune

- Nom de l’évènement: invalidName
  Direction: Le serveur vers le client.
  Données: Aucune

- Nom de l’évènement:sendInvite
  Direction: Le serveur vers l’autre client.
  Données: Aucune

- Nom de l’évènement: updateSelectionPage
  Direction: Le serveur vers tous les clients.
  Données: levelId (number) : Le numéro identifiant du niveau.
  canJoin (boolean) : Un état qui détermine si le niveau est joignable.

- Nom de l’évènement: startMultiplayerGame
  Direction: Le serveur vers un client.
  Données: levelId: Le numéro identifiant du niveau.
  playerName: le nom du joueur.

secondPlayerName: le nom de l'autre joueur

- Nom de l'évènement: rejectGame
  Direction: Le serveur vers l'autre client
  Données: Aucunes

- Nom de l'évènement: shutdown
  Direction: Le serveur vers un client
  Données: Aucune

- Nom de l'évènement: startCheatMode
  Direction: Le serveur vers le client
  Données: data (number[]) Liste de pixels de différence retrouvés dans l'image

- Nom de l'évènement:opponentAbandoned
  Direction: Le serveur vers l'autre client
  Données: Aucune

- Nom de l'évènement:messageSent
  Direction: Le serveur vers un client
  Données: Le serveur renvoie un object Message qui contient:

- Nom de l'évènement:sendGlobalMessage
  Direction: Le serveur vers tous les clients
  Données: Le serveur renvoie un object Message qui contient:

- Nom de l'évènement:sendHint
  Direction: Le serveur vers le client
  Données: hintPixels (number []): Une liste de tous les pixels utilisé pour l'indice

Protocole HTTP:

Méthode: Get
Route: api/image/allLevels
Corps: aucun
Paramètre: aucun
Code de retour: OK (200)
Contenu du retour: Une liste de toute l'information de tous les niveaux

Méthode: Get
Route: api/image/:id
Corps: aucun
Paramètre: id: Le numéro identifiant du niveau
Code de retour: OK (200)
Contenu du retour: Toute l'information sur le niveau particulier

Méthode: Post
Route: api/game/postGlobalVariables
Corps: formData (GlobalVariablesData):
countdownTime (number) : temps initiale du compte à rebours
penalityTime (number) : temps perdu à cause d'une pénalité
discoveryTime (number) : temps gagné à cause d'une découverte d'une différence
Paramètre: aucun
Code de retour: CREATED (201)
Contenu du retour: Rien

Méthode: Post
Route: api/image/postLevel
Corps: formData (LevelFormData):
imageOriginal (File): Le fichier représentant l'image originale
imageDiff (File): Le fichier représentant l'image de différence
name (string): Le nom du niveau
isEasy (string): La difficulté du niveau
clusters (string): La de pixels de différence
nbDifferences (string): Le nombre de différence dans le niveau
Paramètre: aucun
Code de retour: Created (201)
Contenu du retour: Un objet Message qui contient soit une confirmation de la création du niveau, sinon un message d'erreur.

Méthode: Delete
Route: api/game/deleteLeaderboard/:id
Corps: aucun
Paramètre: id: Le numéro identifiant du niveau
Code de retour: ACCEPTED (202)
Contenu du retour: Rien

Méthode: Delete
Route: api/game/deleteHistory
Corps: aucun
Paramètre: aucun
Code de retour: ACCEPTED (202)
Contenu du retour: Rien

Méthode: Delete
Route: api/game/resetGlobalVariables
Corps: aucun
Paramètre: aucun
Code de retour: ACCEPTED (202)
Contenu du retour: Rien