Universidad Diego Portales

Facultad de Ingeniería

Desarrollo Web: Documentación Backend

Fecha de entrega: 3 de Mayo del 2018.

Profesor:
Maximiliano Andrés Vega
Integrantes:
Álvaro Gutiérrez
Juan Hahn
Eduardo Peña
Alejandro Zúñiga

Índice

1. Funciones Backend

1.1. Summoner

- get /find/:server/:summonerName
  - Entradas: Recibe el nombre del summoner y el servidor al que pertenece.
  - Salidas: Se buscará en la base de datos si el summoner solicitado esta registrado. Si existe se notifica con un status 1 que el summoner ya se ingreso, en caso contrario se ingresará la información obtenido de la API a la base de datos. Si hay problemas con el ingresó del summoner o el nombre entregado es invalido se notificara con un status 0, de lo contrario se notifica un status 1.
- get /update/:server/:summonerName
  - Entradas: Recibe el nombre del summoner y el servidor al que pertenece.
  - Salidas: Se buscará en la base de datos de datos si el summoner solicitado está registrado. Si el nombre del summoner no es valido, no exsite o ocurre un error en la consulta se notificara con un status 0. Si existe se realizan los cambios en sus atributos y se notifica con un status 1 en caso de una actualización correcta, de lo contrario se notifica con status 0.
- get /all
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Regresa la información de todos los summoners contenidos en el modelo summoner. Si la consulta esta vacía o bien ocurre algún problema al momento de la consulta se notificara con un status 0, de lo contrario se notifica un status 1.

1.2. Match List

- get /find/:server/:accountId
  - Entradas: Recibe el id de la cuenta y el servidor al que pertenece.
  - Salidas: Busca si existe la cuenta solicitada en la base de datos. Si la cuenta no existe se notifica con un status 0, de lo contrario se buscan todas las partidas asociadas a este. Si no hay partidas asociadas se solicitan a la API y se registran en la base de datos notificando con un status 1. Si hubieran 1 o más partidas registradas al momento de consultar se solicitan a la API todas las partidas después de la última fecha registrada y se ingresan a la base de datos notificando con un status 1. Si ocurre algun problema al momento de solicitar o registrar los datos de la base de datos se notificara con un status 0.

- get /all
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Entrega el listado de todas las matchlist registradas en la base de datos y notifica con un status 1. De lo contrario si la consulta esta vacía u ocurre un problema con la base de datos se notifica con status 0.
- get /matchEs/:gameId/:server/:accountId/:summonerId
  - Entradas: Recibe un gameId, server, accountId y summonerId.
  - Salidas: Entrega la información de un match en especifico
- get /getMatchlist/:server/:accountId
  - Entradas: Recibe un server, accountId.
  - Salidas: Entrega todos los match de un summoner

1.3. Associated Accounts

- post /assig
  - Entradas: Recibe el token del usuario de Firebase y el id del summoner.
  - Salidas: Asocia el summoner indicado con el usuario que se encuentra actualmente logueado y notifica con un status 1 en caso de un registro correcto. En caso contrario si ocurre algún problema con la base de datos o el hay problemas para verificar el usuario se notificará con un status 0.
- post /create
  - Entradas: Recibe un email y contraseña referentes a una cuenta a registrar.
  - Salidas: Se creará el usuario en Firebase con los parámetros de entrada mencionados y se notificará con un status 1 si la creación resulta exitosa. En caso contrario se notificará con un status 0 si ocurre algún problema con Firebase o si el usuario ya existe.
- post /get
  - Entradas: Recibe el token del usuario en Firebase.
  - Salidas: Regresa todos los summoners asociados a la cuenta actualmente logueada y notifica con status 1. De lo contrario si hay un problema con la base de datos, la verificación del usuario o un summoner no existe se notificará con status 0.
- post /delete

- Entradas: Recibe el token del usuario en Firebase y un summonerId.
- Salidas: Borra la relación entre un usuario y un summoner, entrega status 1 y delete ok si realiza el borrado y status 0 si no lo logra.

1.4. League

- get /find/:server/:summonerId
  - Entradas: Recibe el id del summoner y el servidor al que pertenece.
  - Salidas: Solicita los datos de la liga del summoner a la API y los ingresa en el modelo league, si hay problemas con el ingreso de la liga se notifica son un status 0. En caso contrario se relaciona la liga con el summoner y se notifica con un status 1. Si ocurre un problema con la base de datos se notifica con un status 0.
- get /update/:server/:summonerId
  - Entradas: Recibe el id del summoner y el servidor al que pertenece.
  - Salidas: Actualiza los datos de la liga del summoner con los datos entregados por la API y notfica con un status 1. Si ocurre algún error o el id del summoner es invalido se notifica con un status 0.
- get /all
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Entrega el listado de todas las ligas de summoners registradas en la base de datos y notifica con un status 1. De lo contrario si la consulta esta vacía u ocurre un problema con la base de datos se notifica con status 0.

1.5. Champions

- get /champions
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Inserta la información de la API referente a los campeones en el modelo champions. Si hay un problema con el registro del campeón se notificará con un status 0, de lo contrario con un status 1.

1.6. Spells

- get /spells
  - Entradas: No recibe parámetros de entrada
  - Salidas: Inserta la información de la API referente a las spells de los campeones en el modelo spells. Si hay un problema con el registro de una spell se notificará con un status 0, de lo contrario con un status 1.

1.7. Items

- get /insertitems
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Inserta la información de la API referente a los items en el modelo item. Si hay una problema con el registro del item se notificará con un status 0, de lo contrario se notifica con un status 1.
- get /assign
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Inserta las relaciones de dependencia entre los items en el modelo fromItem. Las dependencias de los items (modelo fromItem) son aquellas en las que un item (una fila del modelo item) se puede construir con otros items (otras filas del modelo item). Si hay una problema con la creación de la relación se notificara con un status 0, de lo contrario se notifica con un status 1.
- get /recover/all
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Regresa la información de todos los items contenidos en el modelo item. Si la consulta esta vacía o bien ocurre algún problema al momento de la consulta se notificara con un status 0, de lo contrario se notifica un status 1.
- get /recover/allrelations
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Regresa todas las relaciones entre 2 items contenidas en el modelo fromItem. Estas relaciones están dadas por el id de un item padre y el id de un item del cual proviene. Si la consulta esta vacía o bien ocurre algún problema al momento de la consulta se notificara con un status 0, de lo contrario se notifica un status 1.

1.8. Simulador

- get /
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Regresa todos los datos de los items y campeones que se encuentren registrados en la base de datos. Si se genera algún problema al momento de obtener los items o los campeones se notificara con un status 0, de lo contrario se notificara con un status 1.

1.9. Builds

- post /build
  - Entradas:Recibe el token de usuario y las ids de campeones e items seleccionados (2 campeones por obligación y hasta 6 items para cada uno).
  - Salidas: Inserta la build a la base de datos para poder mostrarla posteriormente en el perfil del usuario.

1.10. Stats

- get /insertStatistics
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Inserta la información de la API referente a las estadísticas de los personajes, según sus roles, en el modelo stat. Si hay una problema con el registro del item se notificará con un status 0, de lo contrario se notifica con un status 1.
- get /recover/all
  - Entradas: No recibe parámetros de entrada.
  - Salidas: Regresa la información de todos las estadísticas contenidos en el modelo stat. Si la consulta esta vacía o bien ocurre algún problema al momento de la consulta se notificara con un status 0, de lo contrario se notifica un status 1.