Swagger Datastore-Service 1.0.0

[Base url: datastore-service.swagger.io/v1]

REST API route documentation for DataStore Microservice.

Schemes

HTTP

docs

GET `/docs` Gets all documents

POST `/docs` Creates a new document

PUT `/docs` Updates an existing document

GET `/docs/{docId}` Find document by id

DELETE `/docs/{docId}` Deletes a document

GET `/docs/{docId}/keywords` Finds all document keywords

POST `/docs/{docId}/keywords` Creates a new document keyword

DELETE `/docs/{docId}/keywords` Deletes all document keywords

GET `/docs/{docId}/sentances` Finds all document sentences

POST `/docs/{docId}/sentances` Creates a new document sentence

PUT `/docs/{docId}/sentances` Updates an existing document sentence

DELETE `/docs/{docId}/sentances` Deletes all document Sentences

GET `/docs/{docId}/sentances/{sentanceId}` Finds a specific document sentence

GET `/docs/{docId}/entities` Finds document entities

participants

GET `/participants` Finds all existing participants

POST `/participants` Creates a new participant

DELETE `/participants` Deletes an exisiting participant

GET `/participants/{participantId}` Finds a participants by Id

GET `/participants/{participantId}/annotations` Finds all annotations generated by participant

POST `/participants/{participantId}/updateStartTime` Updates the annotation start time of participants

POST `/participants/{participantId}/updateEndTime` Updates the annotation end time of participants

annotations

GET `/annotations` Finds all annotations

POST `/annotations` Creates a new annotation

GET `/annotations/{annotationId}` Finds annotation by id

PUT `/annotations/{annotationId}` Updates an existing annotation

/annotations/{annotationId}

DELETE **/annotations/{annotationId}** Deletes an existing annotation

entities

GET **/entities** Finds all entities

POST **/entities** Creates a new entity

GET **/entities/{entityId}** Finds an entity by id

PUT **/entities/{entityId}** Updates an existing entity

DELETE **/entities/{entityId}** Deletes an existing entitiy

PUT **/entities/{entityId}/resolve** Resolves/Disambiguates an existing entity

PUT **/entities/{entityId}/threshold** Updates the ambiguity threshold of an existing entity

GET **/entities/{entityId}/collocations** Finds all collocations associated with the entity

POST **/entities/{entityId}/collocations** Associates a new collocation with the entity

DELETE **/entities/{entityId}/collocations** Deletes all collocations associated with the entity

GET **/entities/{entityId}/candidates** Finds all candidates associated with the entity

POST **/entities/{entityId}/candidates** Associates a new candidate with the entity

DELETE **/entities/{entityId}/candidates** Deletes all candidates associated with the entity

brooker

POST `/brookerInvoke` Invokes the (amqp) message brooker that initiates the framework named entity linking procedure

game

POST `/game/authenticate` Authenticates player into the game

POST `/game/register` Registers players' authentication information into the system

GET `/game/categories/{categoryId}` Finds all game categories

POST `/game/categories/{categoryId}` Creates a new game category

GET `/game/playerStats/{playerId}` Finds game statistics of a player

POST `/game/score/{playerId}` Persists new scores to the payer

POST `/game/wpm/{playerId}` Persists new wpm score for the player

POST `/game/level/{playerId}/levelUpPlayer` Checks (based on accumulated points) if player has leveled up and assigns a new level to the player

POST `/game/addGameRound` Creates a new game round (annotation version for the game)

GET `/game/getChallengers/{wpm}/player/{playerId}` Finds possible challengers for player to challenge based on WPM

POST `/game/challengePlayers` Creates a new game challenge

GET `/game/getPlayerChallenges/{playerId}` Finds all (pending) challanges assigned to the player

GET `/game/getChallengeInfo/{challengeId}` Fetches game challenge information

Updates a game challenge, deciding who the looser and the winner of

POST `/game/updateChallenge` the challenge

GET `/game/getProfileStats/{playerId}` Finds players' profile information

GET `/game/getUpdatedChallenges/{playerId}` Fetches all challenges that have been completed (Won, Lost, Draw)

GET `/game/getLeaderBoard` Fetches the leaderbord with all players and theircorresponding rankings

GET `/game/getPlayerPositionInLeaderboard/{playerId}` Finds the players' exact position on the leaderboard