REST API for Skjutsgruppen Final Version: 1.0

The API is divided in to different root resources. They are:

- Seat*BookingResource*
- *EventLogResource,*
- *FriendshipResource*,
- *SettingsResource*,
- *LiftResource*
- *JourneyResource*.

Below the different resources will be described.

The SeatBookingResource

This resource will contain the operations needed to book and cancel a booking of a seat on an existing carpooling journey.

Precondition:	The member must be authenticated.
Path:	*/api/v1/seatbooking/{journeyId}*
HTTPMethod:	POST
Produces:	application/json
Operation name:	*+bookSeat(journeyId:Long): Boolean*
Operation description:	Make a seatbooking for a specific member on an existing carpooling journey.
Parameter description:	***journeyId*** *"The unique identifier of a carpooling journey",*
Response:	A boolean value which indicates wether the operation succeded or not.

Precondition:	The member must be authenticated.
Path:	*/api/v1/seatbooking/{journeyId}*
HTTPMethod:	DELETE
Produces:	application/json
Operation name:	*+cancelSeat(journeyId:Long): Boolean*
Operation description:	Cancel a seatbooking for a specific member on an existing carpooling journey.
Parameter description:	***journeyId*** *"The unique identifier of a carpooling journey"*
Response:	A boolean value which indicates wether the operation succeded or not.

The EventLogResource

This resource is supposed to update read status on an event.

Precondition:	The member must be authenticated.
Path:	*/api/v1/eventlog/modifyreadstatus/{eventId}/{readStatus}*
HTTPMethod:	PUT
Produces:	application/json
Operation name:	*+modifyReadStatus(eventId:Long, readStatus:Boolean): Boolean*
Operation description:	Modify the readstatus on a particular event.
Parameter description:	***eventId*** *"The unique identifier of a event",* ***readStatus*** *"The read status"*
Response:	A boolean value which indicates wether the operation succeded or not.

The FriendshipResource

This resource is all about add and remove friends for a specific friend(the logged in member) and search friends from Skjutsgruppen and determine how we know each other.

Precondition:	The member must be authenticated.
Path:	*/api/v1/friends?friends={friends(0), friends(1) .. }*
HTTPMethod:	POST
Produces:	application/json
Operation name:	*+addFriends(friends:List<Long>): Boolean*
Operation description:	Adds friends for a particular friend which is member at Skjutsgruppen.
Parameter description:	***friends*** *"The friends id list"*
Response:	A boolean value which indicates wether the operation succeded or not.

Precondition:	The member must be authenticated.
Path:	*/api/v1/friends?friends={friends(0), friends(1) .. }*
HTTPMethod:	DELETE
Produces:	application/json
Operation name:	*+removeFriends(friends:List<Long>): Boolean*

Operation description:	Adds friends for a particular friend which is member at Skjutsgruppen.
Parameter description:	***friends*** *"The friends id list"*
Response:	A boolean value which indicates wether the operation succeded or not.

Precondition:	The member must be authenticated.
Path:	*/api/v1/friends/relationtree?userId={userId}*
HTTPMethod:	GET
Produces:	application/json
Operation name:	*+howWeKnowEachother(userId:Long): Map<Integer,String>*
Operation description:	Adds friends for a particular friend which is member at Skjutsgruppen.
Parameter description:	***userId*** *"The unique identifier of a friend"*
Response:	A Map with the key as the order in which level they know eachother and the value is the name of the friend which is most relevant. Which makes it possible to present how we know each other. **JSON Format:** { "1":"friendOnLevel1", "2":"friendOnLevel2", "3":"friendOnLevel3", "4":"friendOnLevel4", "5":"friendOnLevel5", "6":"friendOnLevel6" }

Precondition:	The member must be authenticated.
Path:	*/api/v1/friends/{friendId}*
HTTPMethod:	GET
Produces:	application/json
Operation name:	*+searchFriends(friendId:Long): List<Friend>*
Operation description:	Search for car pooling friends.
Parameter description:	**friendId** *"The unique identifier of a friend"*

Response:	A list of car pooling friends. **Abstract JSON Format *:** [{"id":type:Long, "firstName":type:String, "lastName":type:String, "email": type:String, "cellPhone":type:String, "activated":type:Boolean, "eventLog":type:Eventlog, "profile":type:Profile, "friends":type:List<Friend>, "journeys":type:List<Journey>, }, ...] * Note, instead of having example values the type of each JSON property is set.

The SettingsResource

This resource is responsible to maintain the settings for a member at Skjutsgruppen. It is all about modifying the phone number, email and avatar.

Precondition:	User already authenticated. Uses SSL.
Path:	*/api/v1/settings/avtar*
HTTPMethod:	PUT
Produces:	application/json
Operation name:	*+changeAvatar(avatar:Byte[]): Boolean*
Operation description	Changes the avatar of a member at Skjutsgruppen.
Parameter description:	***avatar** 'The member's avatar'*
Response:	*A boolean value which indicates wether the operation succeded or not.*

Precondition:	User already authenticated. Uses SSL.
Path:	*/settings/email*
HTTPMethod:	PUT
Produces:	application/json
Operation name:	*+changeEmail(email:String): Boolean*
Operation description	Changes the email of a member at Skjutsgruppen.
Parameter description:	***email** 'The member's email adress'*
Response:	*A boolean value which indicates wether the operation succeded or not.*

Precondition:	User already authenticated. Uses SSL.
Path:	*/api/v1/settings/phoneNumber*
HTTPMethod:	PUT
Produces:	application/json
Operation name:	*+changePhonenumber(phoneNumber:String): String*
Operation description	Changes the phone number of a member at Skjutsgruppen.
Parameter description:	***phoneNumber*** *'The member's phone number'*
Response:	The system generates a code which will be sent to specific number, and a verification code is returned in the phone. That code must be altered in order to approve the change of the phone number.

The LiftResource

This resource represents the search capabilities for a lift from one destination to another.
Which is calculates the related journeys based on the delta of the towns along the way registered on specific journeys. In the calculation also the time interval is 3 days back and 3 days forward in time are included.

With other words the search result contains both the journeys which represents the the from and to destination and exact date and time and the related journeys in time and space, based on the 5 days back forward and the delta on the towns along the way registered on specific journeys.

It is possible to search for everyone but some information will be hidden, if the user/member is not authenticated.For example only the first name is visible on journeys and its driver and passenger and friends and potential friends.

Precondition:	Not need to be authenticated tough the searchresult will be filtered. Uses SSL.
Path:	*/api/v1/lift/search? from={from}&to={to}&datetime={dateTime}&searchoroffer={searchOrOfferLift}*
HTTPMethod:	GET
Produces:	application/json
Operation name:	*+search(from:String, to:String, dateTime:Date, searchOrOfferLift:Boolean): SearchResult*
Operation description	Searches for lift either you are searching or offerering lift.
Parameter description:	***from*** *'The "from" destination',* ***to*** *'The to idestination',* ***dateTime*** *'The date and time' ,* ***searchOrOfferLift*** *'Search or offer lift, true if search false if offer.'*
Response:	A SearchResult object: ***UML fomat:***

	SearchResult: *journeys:List<Journey>* *relatedJourneys:List<Journey>* ***Abstract JSON format *:*** Searchresult: *{* *"journeys": type:List<Journey>* *"relatedJourneys": type:List<Journey>* *}* ** Note, instead of having example values the type of each JSON property is set.*

The JourneyResource

This resource is responsible for retrieving and adding and removing journeys as well search for journeys and nearby journeys with different serach criteria.

Precondition:	The member must be authenticated.
Path:	*/api/v1/journeys/{journeyId}*
HTTPMethod:	GET
Produces:	application/json
Operation name:	*+getJourney(journeyId:Long): Journey*
Operation description:	Retrievs a specific journey.
Parameter description:	***journeyId*** *"The unique identifier of a carpooling journey",*
Response:	***UML format:*** Journey: id:Long, 'Unique Identifier of a Journey' journeyType:Enum, 'The journey type' dateTime:DateTime, ' The date time' from:Destination, 'The from destination' to:Destination, ' The to destination" townsAlongTheWay:String[], 'The towns where driver stops along the way to the to destination' eventlog:EventLog, ' The eventlog' comments:List<Comment>, 'The comments ' seats:Integer, 'The number of seats on a journey' friends:List<Friend>, 'The carpooling friends' *journeyLinkUrl:String ' The unique link shortened url of a Journey'* ***Abstract JSON format *:*** Journey: {

	"id": type:Long, "journeyType":type:Enum, "dateTime": type:DateTime, "to":type:Destination, "from": type:Destination, "townsAlongTheWay": type:String[], "eventlog" type:EventLog, "comments": type:List<Comment>, "seats":type:Integer, "friends"type:List<Friend>, "journeyLinkShortUrl":type:String } ** Note, instead of having example values the type of each JSON property is set.*

Precondition:	The member must be authenticated.
Path:	*/api/v1/journeys/{journeyId}*
HTTPMethod:	DELETE
Produces:	application/json
Operation name:	*+removeJourney(journeyId:Long): Boolean*
Operation description:	Removes a specific journey.
Parameter description:	***journeyId*** *"The unique identifier of a carpooling journey",*
Response:	*A boolean value which indicates wether the operation succeded or not.*

Precondition:	The member must be authenticated.
Path:	*/api/v1/journeys/search/lift/addjourney*
HTTPMethod:	POST
Produces:	application/json
Consumes:	application/json
Operation name:	*+addJourney(from:Destination,*to:Destination, dateTime:Date,comment:String,publishToTwitter:Boolean , *publishToFacebook:Boolean): Boolean*
Operation description:	Adds a journey for a person which searches lift and are a potential passenger on a carpooling journey which matches the search critera.
Parameter description:	***UML format:*** *from:Destination,* *to:Destination,* *dateTime:Date,* *comment:String,*

	publishToTwitter:Boolean , *publishToFacebook:Boolean* **Abstract JSON format*:** ”from” : { "id":type:Long, "destinationName":type:String, "lang":type:String, "lat":type:String} , ”to” : { "id":type:Long, "destinationName":type:String, "lang":type:String, "lat":type:String} **,** ”dateTime” : type:Date, ”comment” : type:String, *”publishToTwitter” : type:String,* *”publishToFacebook” : type: String*
Response:	***UML format:*** Journey: id:Long, 'Unique Identifier of a Journey' journeyType:Enum, 'The journey type' dateTime:DateTime, ' The date time' from:Destination, 'The from destination' to:Destination, ' The to destination" townsAlongTheWay:String[], 'The towns where driver stops along the way to the to destination' eventlog:EventLog, ' The eventlog' comments:List<Comment>, 'The comments ' seats:Integer, 'The number of seats on a journey' friends:List<Friend>, 'The carpooling friends' *journeyLinkUrl:String ' The unique link shortened url of a Journey'* ***Abstract JSON format *:*** Journey: { "id": type:Long, "journeyType":type:Enum, "dateTime": type:DateTime, "to":type:Destination, "from": type:Destination, "townsAlongTheWay": type:String[], "eventlog" type:EventLog, "comments": type:List<Comment>, "seats":type:Integer, "friends"type:List<Friend>, "journeyLinkShortUrl":type:String }

	** Note, instead of having example values the type of each JSON property is set.*

Precondition:	The member must be authenticated.
Path:	*/api/v1/journeys/offer/lift/addjourney*
HTTPMethod:	POST
Produces:	application/json
Consumes:	application/json
Operation name:	*+addJourney(from:Destination,*to:Destination, dateTime:Date, townsAlongTheWay:String[],seats:Integer,comment:String, publishToTwitter:Boolean , *publishToFacebook:Boolean): Journey*
Operation description:	Adds a journey for a person which offers lift and are driver on a carpooling journey.
Parameter description:	***UML format:*** *from:Destination,* *to:Destination,* *dateTime:Date,* *townsAlongTheWay:String[],* *seats:List<Seat>* *comment:String,* *publishToTwitter:Boolean ,* *publishToFacebook:Boolean* **Abstract JSON format*:** ”from” : { "id":type:Long, "destinationName":type:String, "lang":type:String, "lat":type:String} , ”to” : { "id":type:Long, "destinationName":type:String, "lang":type:String, "lat":type:String} **,** ”dateTime” : type:Date, *”townsAlongTheWay”: type:String[],* *”seats”: type:Integer,* ”comment” : type:String, *”publishToTwitter” : type:String,* *”publishToFacebook” : type: String*
Response:	***UML format:***

	Journey: id:Long, 'Unique Identifier of a Journey' journeyType:Enum, 'The journey type' dateTime:DateTime, ' The date time' from:Destination, 'The from destination' to:Destination, ' The to destination'' townsAlongTheWay:String[], 'The towns where driver stops along the way to the to destination' eventlog:EventLog, ' The eventlog' comments:List<Comment>, 'The comments ' seats:Integer, 'The number of seats on a journey' friends:List<Friend>, 'The carpooling friends' *journeyLinkUrl:String ' The unique link shortened url of a Journey'* ***Abstract JSON format *:*** Journey: { "id": type:Long, "journeyType":type:Enum, "dateTime": type:DateTime, "to":type:Destination, "from": type:Destination, "townsAlongTheWay": type:String[], "eventlog" type:EventLog, "comments": type:List<Comment>, "seats":type:List<Seat>, "friends"type:List<Friend>, "journeyLinkShortUrl":type:String } ** Note, instead of having example values the type of each JSON property is set.*

Precondition:	The member must be authenticated.
Path:	*/api/v1/search/nearby/by/member?dateTime={dateTime}&from={from}&to= {to}*
HTTPMethod:	GET
Produces:	application/json
Consumes:	application/json
Operation name:	*+searchNearbyJourneysByMember(dateTime:Date, from:Destination, to:Destination): List<Journey>*
Operation description:	Search for nearby journeys for a specific member.
Parameter description:	**UML Format:** DateTime:Date ”The date time”, from:Destination 'The from destination'

	to:Destination ”The to destination” ***Abstract JSON format *:*** Destination: ”dateTime” : type:Date ”from” : { ”id” : type:Long, ”destinationName” : type: String, ”lang” : type:String, ”lat” : type:String }, ”to” : { ”id” : type:Long, ”destinationName” : type: String, ”lang” : type:String, ”lat” : type:String } *Note, instead of having example values the type of each JSON property is set.*
Response:	A list of journeys. ***Abstract JSON format *:*** Journey: [{ "id": type:Long, "journeyType":type:Enum, "dateTime": type:DateTime, "to":type:Destination, "from": type:Destination, "townsAlongTheWay": type:String[], "eventlog" type:EventLog, "comments": type:List<Comment>, "seats":type:List<Seat>, "friends"type:List<Friend>, "journeyLinkShortUrl":type:String } ...] ** Note, instead of having example values the type of each JSON property is set.*

Precondition:	The member must be authenticated.
Path:	*/api/v1/search/by/member?dateTime={dateTime}&from={from}&to= {to}*
HTTPMethod:	GET
Produces:	application/json
Consumes:	application/json

Operation name:	*+searchNearbyJourneysByMember(dateTime:Date, from:Destination, to:Destination): List<Journey>*
Operation description:	Search for all journeys for a specific member.
Parameter description:	**UML Format:** DateTime:Date ”The date time”, from:Destination 'The from destination' to:Destination ”The to destination” ***Abstract JSON format *:*** Destination: ”dateTime” : type:Date ”from” : { ”id” : type:Long, ”destinationName” : type: String, ”lang” : type:String, ”lat” : type:String }, ”to” : { ”id” : type:Long, ”destinationName” : type: String, ”lang” : type:String, ”lat” : type:String } *Note, instead of having example values the type of each JSON property is set.*
Response:	A list of journeys. ***Abstract JSON format *:*** Journey: [{ "id": type:Long, "journeyType":type:Enum, "dateTime": type:DateTime, "to":type:Destination, "from": type:Destination, "townsAlongTheWay": type:String[], "eventlog" type:EventLog, "comments": type:List<Comment>, "seats":type:List<Seat>, "friends"type:List<Friend>, "journeyLinkShortUrl":type:String } …] ** Note, instead of having example values the type of each JSON property is set.*

Precondition:	The member must be authenticated.
Path:	*/api/v1/search/nearby/by/friends?dateTime={dateTime}&from={from}&to=*

	{to}&friends={friends(0),friends(1) ..}
HTTPMethod:	GET
Produces:	application/json
Consumes:	application/json
Operation name:	*+searchNearbyJourneysByFriends(dateTime:Date, from:Destination, to:Destination, friends:List<Long>): List<Journey>*
Operation description:	Search for nearby journeys for a friends list.
Parameter description:	**UML Format:** DateTime:Date 'The date time', from:Destination 'The from destination' to:Destination 'The to destination' friends:List<Long> 'Friendslist' ***Abstract JSON format *:*** Destination: ”dateTime” : type:Date ”from” : { ”id” : type:Long, ”destinationName” : type: String, ”lang” : type:String, ”lat” : type:String }, ”to” : { ”id” : type:Long, ”destinationName” : type: String, ”lang” : type:String, ”lat” : type:String }, ”friends” : type:List<Long> *Note, instead of having example values the type of each JSON property is set.*
Response:	A list of journeys. ***Abstract JSON format *:*** Journey: [{ "id": type:Long, "journeyType":type:Enum, "dateTime": type:DateTime, "to":type:Destination, "from": type:Destination, "townsAlongTheWay": type:String[], "eventlog" type:EventLog, "comments": type:List<Comment>, "seats":type:List<Seat>,

	"friends"type:List<Friend>, "journeyLinkShortUrl":type:String } ...] ** Note, instead of having example values the type of each JSON property is set.*

Precondition:	The member must be authenticated.
Path:	*/api/v1/search/by/friends?dateTime={dateTime}&from={from}&to={to}&friends={friends(0),friends(1) ..}*
HTTPMethod:	GET
Produces:	application/json
Consumes:	application/json
Operation name:	*+searchJourneysByFriends(dateTime:Date, from:Destination, to:Destination, friends:List<Long>): List<Journey>*
Operation description:	Search for all journeys for a friends list.
Parameter description:	**UML Format:** DateTime:Date 'The date time', from:Destination 'The from destination' to:Destination 'The to destination' friends:List<Long> 'Friendslist' ***Abstract JSON format *:*** Destination: ”dateTime” : type:Date ”from” : { ”id” : type:Long, ”destinationName” : type: String, ”lang” : type:String, ”lat” : type:String }, ”to” : { ”id” : type:Long, ”destinationName” : type: String, ”lang” : type:String, ”lat” : type:String }, ”friends” : type:List<Long> *Note, instead of having example values the type of each JSON property is set.*

Response:	A list of journeys. ***Abstract JSON format *:*** Journey: [{ "id": type:Long, "journeyType":type:Enum, "dateTime": type:DateTime, "to":type:Destination, "from": type:Destination, "townsAlongTheWay": type:String[], "eventlog" type:EventLog, "comments": type:List<Comment>, "seats":type:List<Seat>, "friends"type:Integer, "journeyLinkShortUrl":type:String } ...] ** Note, instead of having example values the type of each JSON property is set.*