nskDöjo 1.0.0

Detailed Functional Design

2021-08-02

Contents

[1 REVISION HISTORY 2](#_Toc78448808)

[2 PURPOSE 3](#_Toc78448809)

[3 Description 3](#_Toc78448810)

[4 CONCEPTS 3](#_Toc78448811)

[5 PERSISTENCE LAYER 3](#_Toc78448812)

[6 DATABASE SCHEMA 3](#_Toc78448813)

[7 CONCEPTS 4](#_Toc78448814)

[7.1 VENUE INFORMATION 4](#_Toc78448815)

[7.1.1 VenueRec 4](#_Toc78448816)

[7.1.2 StudioRec 4](#_Toc78448817)

[7.2 ACTIONS 5](#_Toc78448818)

[7.2.1 ON VENUES 5](#_Toc78448819)

[7.2.2 ON STUDIOS 7](#_Toc78448820)

# REVISION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 2021/08/02 | 1.0 | Initial version of the document | Paulo Márquez |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# PURPOSE

Döjo is a Next-Generation platform for the management of athletic facilities that puts the control in the member’s hands.

Döjo is the orchestrator, the service that keeps everything flowing smoothly and efficiently.

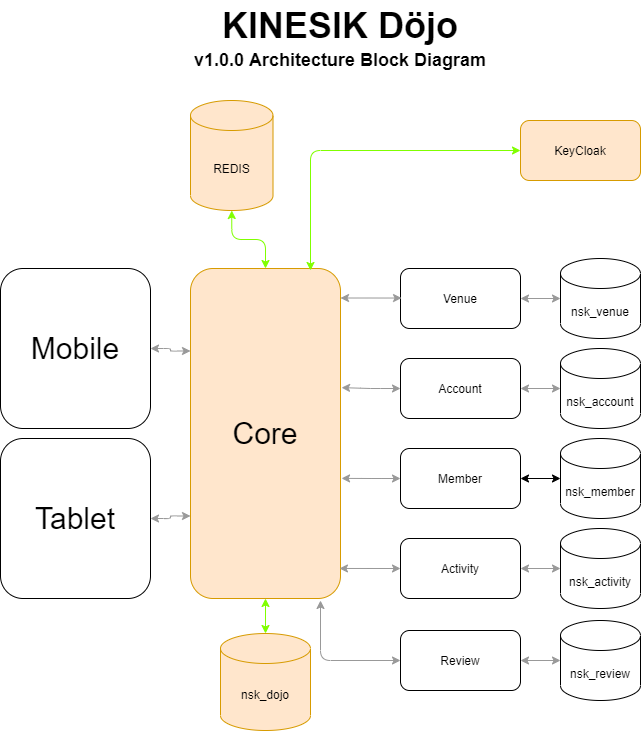
# Description

nskDojo is the Athletic Facility management service for the KINESIK platform.

# CONCEPTS

|  |  |
| --- | --- |
| **VENUE** | Physical building where a Venue is located. |
| **STUDIO** | Room within the venue. |
| **ACCOUNT** | A KINESIK account can have one or more associated venues. |
| **MEMBER** | A person that consumes the services. |
| **ACTIVITY** | The different activities, sports and events related to an athletic venue. |

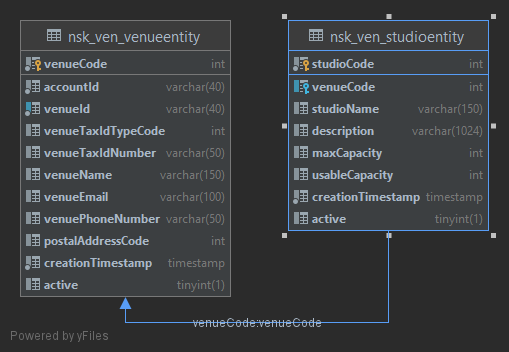
# ARCHITECTURE BLOCK DIAGRAM



# PERSISTENCE LAYER

MySQL is the DBMS of choice; SQL is the implementation

# DATABASE SCHEMA



# WORKFLOW

As an orchestrator, ALL workflows will be driven from Döjo, these workflows are:

* Accounts
* Venues
* Persons
* Memberships
* Activities
* [OTHERS?]

## Accounts

This is one of the most important workflows in the platform, it will drive the creation and maintenance of the different Accounts, which are the basic building block where we will be adding Venues, Persons, Activities and all the other possible workflows. In short, the Account is the outermost layer of the operation.

For starters, the Accounts workflow will have the responsibility of creating a wrapper-space where all other elements will be grouped, for this, it must have information of what venues belong to an account, this will create the external limits of the scope of an account.

All access to an Account will be protected via an OAuth2 token generated by KeyCloak. The roles supported will be:

* Admin - Account owner
* Operator – A person that is part of the organization represented by the Account.
* User - The Persons that will bear memberships in a Venue.
* [Others?]

## VENUE INFORMATION

The information we are keeping about a Venue is.

### VenueRec

int venueCode;

String venueId;

int accountCode;

String venueName;

String venueEmail;

String venuePhoneNumber;

LocalDateTime creationTimestamp;

List<StudioRec> studios;

boolean active;

Since a Venue can have one or more Studios, we also need the information about them.

### StudioRec

int studioCode;

int venueCode;

String studioName;

String description;

int maxCapacity;

int usableCapacity;

LocalDateTime creationTimestamp;

boolean active;

## API

### ON VENUES

##### Create Venue

Endpoint that creates a new Venue.

**Method:** POST

**Endpoint:** /venuesAPI/1.0/venue/

**Parameters**: VenueRec venue: The required data to create the venue.

**Returns**: void

**Headers**: Location = {Venue Resource Location}

**Status Codes**:

201: Created.

400: Could not create the Studio with the supplied data.

500: Internal Server Error.

##### Retrieve Venues

Endpoint to retrieve all Venues within a list of Ids.

##### Retrieve Venue

Retrieves a single Venue by Id

**Method:** GET

**Endpoint:** /venuesAPI/1.0/venue/{venueId}

**Parameters**: venueId: The UUID that represents the venue.

**Returns**: VenueRec

**Headers**: none

**Status Codes**:

200: OK

404: Venue not found

##### Search for Venue

Retrieves zero or more Venues by any of a number of possible search criteria.

##### Delete Studio from Venue

Deletes a Studio from a Venue.

**Method:** PATCH

**Endpoint:** /venuesAPI/1.0/venue/studio/{studioCode}

**Parameters**: studioCode: code of the Studio to delete.

**Returns**: void

**Status Codes**:

204: No Content

404: Studio not found

##### Enable Venue

Mary

##### Disable Venue

Pepe

##### Delete Venue

Deletes a Venue.

**Method:** DELETE

**Endpoint:** /venuesAPI/1.0/venue/{venueId}

**Parameters**: studioCode: Id of the Venue to delete.

**Returns**: void

**Status Codes**:

204: No Content

404: Studio not found

### ON STUDIOS

##### Add Studio from Venue

Adds a Studio to a Venue

**Method:** PATCH

**Endpoint:** /venuesAPI/1.0/venue/{venueId}/studio

**Parameters**: venueId: The UUID that represents the venue.

**Returns**: void

**Headers**: Location = {Venue Resource Location}

**Status Codes**:

201: Created.

400: Could not create the Studio with the supplied data.

404: Venue not found.

500: Internal error when attempting the creation of the Studio.

##### Retrieve Studios

Endpoint to retrieve all Venues that belong to the current account.

##### Retrieve Studio

Retrieves a single Studio by Id.

##### Search for Studio

Retrieves zero or more Studios by any of a number of possible search criteria.

##### Update Studio

Updates the information related to a Studio.

##### Delete Studio from Venue

Deletes a Studio from a Venue.

**Method:** PATCH

**Endpoint:** /venuesAPI/1.0/venue/studio/{studioCode}

**Parameters**: studioCode: code of the Studio to delete.

**Returns**: void

**Status Codes**:

204: No Content

404: Studio not found

##### Enable Studio

Enables the usability of a Studio

##### Disable Studio

Disables a Studio from being used.