[Testing 4](#_Toc9368257)

[Entities 5](#_Toc9368258)

[User 5](#_Toc9368259)

[School 5](#_Toc9368260)

[Organization 5](#_Toc9368261)

[StudyProgram 5](#_Toc9368262)

[StudyModule 5](#_Toc9368263)

[StudyCompetence 5](#_Toc9368264)

[Internship 6](#_Toc9368265)

[AgreementArticle 6](#_Toc9368266)

[InternShipAgreement 6](#_Toc9368267)

[Grades 6](#_Toc9368268)

[FileUpload 6](#_Toc9368269)

[ROLES 7](#_Toc9368270)

[JWT User Login 8](#_Toc9368271)

[GET /loginJWT/{jwttoken} 8](#_Toc9368272)

[GET /loginJWT?User=jwttoken 8](#_Toc9368273)

[POST /loginJWT 8](#_Toc9368274)

[Authorization: Bearer 8](#_Toc9368275)

[REST calls 8](#_Toc9368276)

[User management 8](#_Toc9368277)

[GET /loginGET/{username}/{password} 8](#_Toc9368278)

[POST /login 8](#_Toc9368279)

[POST /createUser 8](#_Toc9368280)

[DELETE /deleteUser 8](#_Toc9368281)

[POST /modifyUserPermission 8](#_Toc9368282)

[GET /listUsers 8](#_Toc9368283)

[POST /modifyUserPassword 9](#_Toc9368284)

[GET /logout 9](#_Toc9368285)

[GET /currentUser 9](#_Toc9368286)

[GET /listUsersWithinOrganization 9](#_Toc9368287)

[POST /createSchooladminUser 9](#_Toc9368288)

[POST /createOrgadminUser 9](#_Toc9368289)

[POST /createMentorUser 9](#_Toc9368290)

[School management 10](#_Toc9368291)

[GET /listSchools 10](#_Toc9368292)

[GET /createSchool 10](#_Toc9368293)

[DELETE /deleteSchool/{school\_id} 10](#_Toc9368294)

[GET /editSchool/{new\_school\_id} 10](#_Toc9368295)

[StudyProgram management 11](#_Toc9368296)

[GET /listStudyPrograms 11](#_Toc9368297)

[POST createStudyProgram 11](#_Toc9368298)

[DELETE /deleteStudyProgram/{studyprogram\_id} 11](#_Toc9368299)

[POST /editStudyProgram 11](#_Toc9368300)

[Student actions 12](#_Toc9368301)

[POST /selfRegister 12](#_Toc9368302)

[Organization management 12](#_Toc9368303)

[GET /listOrganizations 12](#_Toc9368304)

[GET /listOrganizationsByType/{organization\_type} 12](#_Toc9368305)

[POST /createOrganization 12](#_Toc9368306)

[DELETE /deleteOrganization/{organization\_id} 12](#_Toc9368307)

[POST /editOrganization 12](#_Toc9368308)

[Internship management 13](#_Toc9368309)

[GET /listInternships 13](#_Toc9368310)

[GET / listInternships/{study\_program\_id} 13](#_Toc9368311)

[POST / createInternship 13](#_Toc9368312)

[DELETE /deleteInternship/{internship\_id} 13](#_Toc9368313)

[POST / editInternship 13](#_Toc9368314)

[Management of AgreementArticles (for schools agreement Template) 14](#_Toc9368315)

[GET /listTemplateArticles 14](#_Toc9368316)

[POST /createTemplateArticle 14](#_Toc9368317)

[DELETE /deleteTemplateArticle/{article\_id} 14](#_Toc9368318)

[POST /editTemplateArticle 14](#_Toc9368319)

[Management of InternShipAgreement 14](#_Toc9368320)

[GET /listInternshipAgreements 14](#_Toc9368321)

[GET /getInternshipAgreement/{internship\_agreement\_id} 14](#_Toc9368322)

[GET /listUnsignedInternshipAgreements 14](#_Toc9368323)

[DELETE /deleteInternshipAgreement/{internship\_agreement\_id} 14](#_Toc9368324)

[POST /createInternshipAgreement 14](#_Toc9368325)

[POST /editInternshipAgreement 14](#_Toc9368326)

[GET /signSchoolInternshipAgreements/{internship\_agreement\_id} 14](#_Toc9368327)

[GET /listStudentInternshipAgreements 15](#_Toc9368328)

[GET /listUnsignedStudentInternshipAgreements 15](#_Toc9368329)

[GET /signStudentInternshipAgreements/{internship\_agreement\_id} 15](#_Toc9368330)

[GET /applyForInternship/{internship\_id} 15](#_Toc9368331)

[POST /studentProposal/{study\_program\_id} 15](#_Toc9368332)

[GET /listOrganizationInternshipAgreements 15](#_Toc9368333)

[GET / listUnsignedOrganizationInternshipAgreements 15](#_Toc9368334)

[GET /signOrganizationInternshipAgreements/{internship\_agreement\_id} 15](#_Toc9368335)

[POST /storeGrades 15](#_Toc9368336)

[Uploading and managing FileUploads 16](#_Toc9368337)

[POST /uploadFile 16](#_Toc9368338)

[POST /uploadFile/{table\_name}/{table\_id} 16](#_Toc9368339)

[GET /download/{file\_id} 16](#_Toc9368340)

[DELETE /deleteFile/{file\_id} 16](#_Toc9368341)

[Use case scenarios 17](#_Toc9368342)

[Logging into a system and managing own account (all ROLEs) 17](#_Toc9368343)

[ADMIN 17](#_Toc9368344)

[Creating and managing USERS 17](#_Toc9368345)

[Creating SCHOOL setup 17](#_Toc9368346)

[SCHOOLADMIN 18](#_Toc9368347)

[Setup StudyPrograms 18](#_Toc9368348)

[UserManagement 18](#_Toc9368349)

[Setup Internships 18](#_Toc9368350)

[Setup InternshipAgreement 18](#_Toc9368351)

[Managing InternshipAgreements 18](#_Toc9368352)

[STUDENT 19](#_Toc9368353)

[Register 19](#_Toc9368354)

[Search and apply for internship 19](#_Toc9368355)

[Proposing self “contracted” internship 19](#_Toc9368356)

[Managing and signing InternshipAgreement 19](#_Toc9368357)

[ORGADMIN 19](#_Toc9368358)

[Managing and signing InternshipAgreement 19](#_Toc9368359)

[UserManagement 20](#_Toc9368360)

[MENTOR 20](#_Toc9368361)

[Grading students 20](#_Toc9368362)

# Testing

Test installation is hosted at ovh.nizetic.hr:8080

All REST API urls for this installation should be prefixed with /SAT-backend/json

i.e.

for GET /loginGET/{username}/{password}

actual call should be placed at (for username admin and password admin – ADMIN role user by default – for production password should be changed)

<http://ovh.nizetic.hr:8080/SAT-backend/json/loginGET/admin/admin>

every fulfilled request returns entity object in JSON form

{

"username":"admin",

"password":"admin",

"school":"TestSchool",

"roles":["ROLE\_ADMIN"],

"phone":null,

"email":null,

"cityname":null,

"cityzipcode":null,

"streetname":null,

"organization":null

}

For actual entities see Entities chapter of this document

RECOMMENDED STARTING POINT FOR UI DEVELOPMENT: **Use case scenarios**

# Entities

## User

(

username String NOT NULL PK

password String NOT NULL

school String NOT NULL FK (schools)

roles String[] NOT NULL

phone String NULL

email String NULL

cityname String NULL

cityzipcode String NULL

streetname String NULL

organization Integer NULL FK (organizations)

)

## School

(

school\_id String NOT NULL PK

public\_key String NULL

)

## Organization

(

id int NOT NULL PK

orgtype string NOT NULL {“SCHOOL“,“ORGANIZATION“}

name String NOT NULL

cityname String NOT NULL

streetname String NOT NULL

contact\_username String NULL FK (USERS)

school\_id String NOT NULL FK (SCHOOLS)

studyPrograms StudyProgram[]

)

## StudyProgram

(

id int NOT NULL PK

name String NOT NULL

description String NOT NULL

studyModules StudyModule[]

studyCompetences StudyCompetence[]

)

## StudyModule

(

id int NOT NULL PK

name String NOT NULL

description String NOT NULL

studyCompetences StudyCompetence[]

)

## StudyCompetence

(

id int NOT NULL PK

name String NOT NULL

description String NOT NULL

)

## Internship

(

id int NOT NULL PK

organization Organization

studyProgram StudyProgram

studyCompetence StudyCompetence[]

)

## AgreementArticle

(

id int NOT NULL PK

school\_id String NOT NULL FK (SCHOOLS)

ordnum int //ord. Num for article

content String

)

## InternShipAgreement

(

id int NOT NULL PK

organization Organization

student User

periodBegin Date

periodEnd Date

studyProgram StudyProgram

school\_id String

version\_num int // version number

articles AgreementArticle[]

competences StudyCompetendes[]

signed\_student\_username String // student signed agreement and when

signed\_student\_date Date

signed\_school\_username String // school signed agreement and when

signed\_school\_date Date

signed\_organization\_username String // organization signed agreement and when

signed\_ organization\_date Date

)

## Grades

(

internshipAgreement InternshipAgreement

studyCompetence StudyCompetence

grade int // 1-5

grate\_text String

mentor User

)

## FileUpload

(

id int NOT NULL PK

owner\_username String NOT NULL FK (USERS)

table\_name String

table\_id int

mime\_type String

filename String

vrijeme Date

bytes Byte[]

)

# ROLES

ADMIN rola to administer system (superadmin)

STUDENT student role

SCHOOLADMIN school admin role

ORGADMIN organization admin role

MENTOR role for mentor within organization (mentor grades students)

# JWT User Login

Authority for login is school system.

After verifieng and authorization, system should redirect user to SAT-backend site (to one of the following URL/methods)

### GET /loginJWT/{jwttoken}

JWT token is passed as a PATH variable

### GET /loginJWT?User=jwttoken

JTW token is passed as a RequestParam varable

### POST /loginJWT

JWT token is passed as a varable encodes into body of the request (POST parameters)

### Authorization: Bearer

HTTP header can also be used to authenticate user

Header will be accepted for following methods

#### GET /loginJWT

#### POST /loginJWT

### PublicKey

Each school entered into a system can (and should) have own PKCS#1 BASE64 encoded public key (stored as School entity)

# REST calls

## User management

### GET /loginGET/{username}/{password}

ACCESS: \* (no restrictions)

PAYLOAD: -

RETURNS: User entity

### POST /login

ACCESS: \* (no restrictions)

PAYLOAD: User entity

RETURNS: User entity

### POST /createUser

ACCESS: ADMIN

PAYLOAD: User entity

RETURNS: User entity

### DELETE /deleteUser

ACCESS: ADMIN

PAYLOAD: User entity

RETURNS: User entity

### POST /modifyUserPermission

ACCESS: ADMIN

PAYLOAD: User entity

RETURNS: User entity

### GET /listUsers

ACCESS: ADMIN

PAYLOAD: User entity

RETURNS: User entity

### POST /modifyUserPassword

ACCESS: any authenticated user (anyone can change own password)

PAYLOAD: User entity

RETURNS: User entity

### GET /logout

ACCESS: any authenticated user

PAYLOAD:

RETURNS:

### GET /currentUser

ACCESS: any authenticated user

PAYLOAD:

RETURNS: User entity

### GET /listUsersWithinOrganization

ACCESS: any authenticated user

PAYLOAD:

RETURNS: User entity list (array)

### POST /createSchooladminUser

ACCESS: SCHOOLADMIN

PAYLOAD:User entity

RETURNS: User entity

* SCHOOLADMIN can create only another SCHOOLADMIN for his own school/organization

### POST /createOrgadminUser

ACCESS: SCHOOLADMIN, ORGADMIN

PAYLOAD:User entity

RETURNS: User entity

* ORGADMIN can create only another ORGADMIN for his own organization

### POST /createMentorUser

ACCESS: SCHOOLADMIN, ORGADMIN

PAYLOAD:User entity

RETURNS: User entity

* ORGADMIN can create only MENTOR for his own organization

## School management

### GET /listSchools

ACCESS: \* (no restrictions)

PAYLOAD:

RETURNS: School entity list (array)

### GET /createSchool

ACCESS: ADMIN

PAYLOAD: School entity

RETURNS: School entity

### DELETE /deleteSchool/{school\_id}

ACCESS: ADMIN

PAYLOAD:

RETURNS: School entity

### GET /editSchool/{new\_school\_id}

ACCESS: ADMIN

PAYLOAD: School entity

RETURNS: School entity

## StudyProgram management

### GET /listStudyPrograms

ACCESS: SCHOOLADMIN

PAYLOAD:

RETURNS: StudyProgram entity list

### POST createStudyProgram

ACCESS: SCHOOLADMIN

PAYLOAD: StudyProgram entity

RETURNS: StudyProgram entity

### DELETE /deleteStudyProgram/{studyprogram\_id}

ACCESS: SCHOOLADMIN

PAYLOAD:

RETURNS: StudyProgram entity

### POST /editStudyProgram

ACCESS: SCHOOLADMIN

PAYLOAD: StudyProgram entity

RETURNS: StudyProgram entity

## Student actions

### POST /selfRegister

ACCESS: \* (no restrictions)

PAYLOAD:

RETURNS: User entity

Creates user with STUDENT role

## Organization management

### GET /listOrganizations

ACCESS: SCHOOLADMIN, ADMIN

PAYLOAD:

RETURNS: Organization entity list (array)

### GET /listOrganizationsByType/{organization\_type}

ACCESS: any authenticated user (any role)

PAYLOAD:

RETURNS: Organization entity list (array)

### POST /createOrganization

ACCESS: SCHOOLADMIN, ADMIN

PAYLOAD: Organization entity

RETURNS: Organization entity

### DELETE /deleteOrganization/{organization\_id}

ACCESS: SCHOOLADMIN, ADMIN

PAYLOAD:

RETURNS: Organization entity

### POST /editOrganization

ACCESS: SCHOOLADMIN, ADMIN

PAYLOAD: Organization entity

RETURNS: Organization entity

## Internship management

### GET /listInternships

ACCESS: any authenticated user (any role)

PAYLOAD:

RETURNS: Internship entity list (array)

### GET / listInternships/{study\_program\_id}

ACCESS: any authenticated user (any role)

PAYLOAD:

RETURNS: Internship entity list (array)

### POST / createInternship

ACCESS: SCHOOLADMIN

PAYLOAD: Internship entity

RETURNS: Internship entity

### DELETE /deleteInternship/{internship\_id}

ACCESS: SCHOOLADMIN

PAYLOAD:

RETURNS: Internship entity

### POST / editInternship

ACCESS: SCHOOLADMIN

PAYLOAD: Internship entity

RETURNS: Internship entity

## Management of AgreementArticles (for schools agreement Template)

### GET /listTemplateArticles

ACCESS: SCHOOLADMIN

PAYLOAD:

RETURNS: AgreementArticle entity list (array)

### POST /createTemplateArticle

ACCESS: SCHOOLADMIN

PAYLOAD: AgreementArticle entity

RETURNS: AgreementArticle entity

### DELETE /deleteTemplateArticle/{article\_id}

ACCESS: SCHOOLADMIN

PAYLOAD:

RETURNS: AgreementArticle entity (deleted one)

### POST /editTemplateArticle

ACCESS: SCHOOLADMIN

PAYLOAD: AgreementArticle entity

RETURNS: AgreementArticle entity

## Management of InternShipAgreement

### GET /listInternshipAgreements

ACCESS: SCHOOLADMIN

PAYLOAD:

RETURNS: InternshipAgreement entity list (array)

### GET /getInternshipAgreement/{internship\_agreement\_id}

ACCESS: SCHOOLADMIN

PAYLOAD:

RETURNS: InternshipAgreement entity

### GET /listUnsignedInternshipAgreements

ACCESS: SCHOOLADMIN

PAYLOAD:

RETURNS: InternshipAgreement entity list (array)

### DELETE /deleteInternshipAgreement/{internship\_agreement\_id}

ACCESS: SCHOOLADMIN

PAYLOAD:

RETURNS: InternshipAgreement entity

### POST /createInternshipAgreement

ACCESS: SCHOOLADMIN

PAYLOAD: InternshipAgreement entity

RETURNS: InternshipAgreement entity

### POST /editInternshipAgreement

ACCESS: SCHOOLADMIN

PAYLOAD: InternshipAgreement entity

RETURNS: InternshipAgreement entity

### GET /signSchoolInternshipAgreements/{internship\_agreement\_id}

ACCESS: SCHOOLADMIN

PAYLOAD:

RETURNS:

### GET /listStudentInternshipAgreements

ACCESS: STUDENT

PAYLOAD:

RETURNS: InternshipAgreement entity list (array)

### GET /listUnsignedStudentInternshipAgreements

ACCESS: STUDENT

PAYLOAD:

RETURNS: InternshipAgreement entity list (array)

### GET /signStudentInternshipAgreements/{internship\_agreement\_id}

ACCESS: STUDENT

PAYLOAD:

RETURNS:

### GET /applyForInternship/{internship\_id}

ACCESS: STUDENT

PAYLOAD:

RETURNS: InternshipAgreement entity

### POST /studentProposal/{study\_program\_id}

ACCESS: STUDENT

PAYLOAD: Organization entity – Student proposes this organization as new one, and applies for Internship there

RETURNS: InternshipAgreement entity

### GET /listOrganizationInternshipAgreements

ACCESS: ORGADMIN

PAYLOAD:

RETURNS: InternshipAgreement entity list (array)

### GET / listUnsignedOrganizationInternshipAgreements

ACCESS: ORGADMIN

PAYLOAD:

RETURNS: InternshipAgreement entity list (array)

### GET /signOrganizationInternshipAgreements/{internship\_agreement\_id}

ACCESS: ORGADMIN

PAYLOAD:

RETURNS:

GET /getGrades/{internship\_agreement\_id}

ACCESS: any authenticated user (any role)

PAYLOAD:

RETURNS: Grades entity list (array)

🡪 if the role is STUDENT then user can get only own grades

### POST /storeGrades

ACCESS: MENTOR

PAYLOAD: Grades entity list (array)

RETURNS: Grades entity list (array)

## Uploading and managing FileUploads

### POST /uploadFile

ACCESS: any authenticated user

PAYLOAD: Multipart file

RETURNS: FileUpload entity

### POST /uploadFile/{table\_name}/{table\_id}

ACCESS: any authenticated user

PAYLOAD: Multipart file

RETURNS: FileUpload entity

Uploads file that is tagged (connected) to specific table (entity) and its ID

### GET /download/{file\_id}

ACCESS: any authenticated user

PAYLOAD: -

RETURNS: actual file (bytes stream)

### DELETE /deleteFile/{file\_id}

ACCESS: any authenticated user

PAYLOAD: -

RETURNS: -

ADMIN and SCHOOLADMIN can delete anyones files, other (STUDENT, MENTOR,…) can only delete own files

# Use case scenarios

## Logging into a system and managing own account (all ROLEs)

#### POST /login

#### POST /modifyUserPassword

#### GET /logout

#### GET /currentUser

## ADMIN

ADMIN should prepare data needed to run the system

### Creating and managing USERS

#### POST /createUser

#### DELETE /deleteUser

#### POST /modifyUserPermission

#### GET /listUsers

ADMI should create all users (odher ADMIN, SCHOOLADMIN, ORGADMIN, MENOTOR) except STUDENT users (STUDENT user can apply for them selves)

### Creating SCHOOL setup

#### POST /createSchool

#### DELETE /deleteSchool/{school\_id}

#### POST /editSchool/{new\_school\_id}

#### GET /listSchools

System provides dana separation and can be used by more than one school at the same time.

IMPORTANT!!!

All users (except ADMIN) must have SCHOOL and ORGANIZATION connected to them, that means that some ADMIN should setup up at least one organization (with orgtype SCHOOL) in order to create SCHOOL admin

#### GET /listOrganizations

#### GET /listOrganizationsByType/{organization\_type}

We can have two types of organizations SCHOOL and ORGANIZATION (which is Internship provider)

#### DELETE /deleteOrganization/{organization\_id}

#### POST /editOrganization

## SCHOOLADMIN

Once a SCHOOLADMIN user is created, that user can take care of data preparation

### Setup StudyPrograms

#### GET /listStudyPrograms

#### POST /createStudyProgram

#### DELETE /deleteStudyProgram/{studyprogram\_id}

#### POST /editStudyProgram

Once StudyPrograms are ready, SCHOOLADMIN should proceed with setting up organizations (InternshiProviders)

#### GET /listOrganizations

#### GET /listOrganizationsByType/{organization\_type}

We can have two types of organizations SCHOOL and ORGANIZATION (which is Internship provider)

#### DELETE /deleteOrganization/{organization\_id}

#### POST /editOrganization

### UserManagement

If needed SCHOOLADMIN can create additional users

#### POST /createSchooladminUser

#### POST /createOrgadminUser

#### POST /createMentorUser

SCHOOLADMIN can create only users that that are connected to his own school

When creating Schooladmin, this is the admin for his own school

### Setup Internships

Internship refers to possible Internships offered to students, it connects StudyProgram and organization (organization offers internship for particular study program to student). It also defines a list of StudyCompetences that student will gain during Internship.

NOTE that StudyProgram has its own set of competences (and that this should correspond with StudyProgram list, but it can, and needs to be modified by the SCHOOLADMIN)

#### GET /listInternships

##### GET /listInternships/{study\_program\_id}

#### POST /createInternship

#### DELETE /deleteInternship/{internship\_id}

#### POST /editInternship

### Setup InternshipAgreement

Before and InternshipAgreement can be created, SCHOOLADMIN needs to setup a template for agreement articles (content of them and also their order)

#### POST /createTemplateArticle

#### DELETE /deleteTemplateArticle/{article\_id}

#### POST /editTemplateArticle

#### GET /listTemplateArticles

### Managing InternshipAgreements

After a student applied for Internship with organization, InternshipAgreement has been created. SCHOOLADMIN needs to manage them (make sure that Agreement is signed by all parties, modify AgreementArticles if needed)…

#### GET /listInternshipAgreements

#### GET /getInternshipAgreement/{internship\_agreement\_id}

#### GET /listUnsignedInternshipAgreements

List of agreement not signed by all parties

#### DELETE /deleteInternshipAgreement/{internship\_agreement\_id}

#### POST /createInternshipAgreement

In case InternshipAgreement needs to be created manually (by hand)

#### POST /editInternshipAgreement

Every time InternshipAgreement is modified, its version is incremented (and that invalidates all signing – new version needs to be signed too in order to Agreement to be valid)

#### POST /signSchoolInternshipAgreements/{internship\_agreement\_id}

InternshipAgreement needs to be signed by the school, so SCHOOLADMIN uses this method to mark it as signed

#### GET /getGrades/{internship\_agreement\_id}

Obtaining the grades STUDENT got from MENTOR

## STUDENT

### Register

If STUDENT has no created account, he/she can register to use the system via

#### POST /selfRegister

In order to fill User entity structure, system will need a list of schools using the system, it can be obtained via

#### GET /listSchools

### Search and apply for internship

Student should search Internship database in order to find Internship for him/herself

#### GET /listInternships

#### GET /listInternships/{study\_program\_id}

And this one is preferred method (Student searches for internship corresponding to his own StudyProgram.

study\_program\_id can be taken from StudyProgram, or from the one of its StudyModule (refined search)

#### POST /applyForInternship/{internship\_id}

Apply for the internship found in the database

### Proposing self “contracted” internship

#### POST /studentProposal/{study\_program\_id}

In case student can not find Internship in the database, Student can find Internship “in the real world”, and then register that with SCHOOL authorities to be handled (Agreement with organization…)

### Managing and signing InternshipAgreement

#### GET /listStudentInternshipAgreements

Lists all students agreements

#### GET /listUnsignedStudentInternshipAgreements

Lists only students unsigned Agreements

#### GET /signStudentInternshipAgreements/{internship\_agreement\_id}

Student signs Agreement

#### GET /getGrades/{internship\_agreement\_id}

Students can check the grades he/she got during internship.

Student can see only his own grades (for his own internship agreements)

## ORGADMIN

### Managing and signing InternshipAgreement

#### GET /listOrganizationInternshipAgreements

List all agreement for the organization

#### GET /listUnsignedOrganizationInternshipAgreements

List only unsigned ones

#### GET /signOrganizationInternshipAgreements/{internship\_agreement\_id}

Sign particular one

### UserManagement

If needed ORGADMIN can create additional users

#### POST /createOrgadminUser

#### POST /createMentorUser

ORGADMIN can create only users that that are connected to his own organization and belong to sam school

When creating ORGADMIN, this is the admin for his own organization

## MENTOR

### Grading students

#### POST /storeGrades

MENTOR should grade the student his organization has Agreement with

NOTE: data structure for grading should beforehand be obtained with

#### GET /getGrades/{internship\_agreement\_id}