

## Allgemeines

- Server URL <https://vs.uniks.de>.
- Das Registrieren und Einloggen am Server sind die einzigen Aktionen, die ohne Login funktionieren.
- Basis URL für REST <https://vs.uniks.de/api>.
- Basis URL für WS <wss://vs.uniks.de/ws>.
- Alle Anfragen müssen als JSON gesendet werden, Ausnahme ist die *noop* (no operation) Anweisung für Asynchrone Verbindungen.
- Der Server antwortet immer im JSON Format, Ausnahme sind binäre Daten wie Bilder.
- REST Anfragen die ein Login voraussetzen, benötigen einen zusätzlichen Header *userKey*. Der Server erstellt bei jedem Login ein neuen Key und invalidiert den alten.
- Nutzer werden nach 15 min. automatisch ausgeloggt, sofern keine Aktion ausgeführt wird.
- Spiele werden nach 5 min. gelöscht, sofern diese nicht korrekt gestartet werden.
- Asynchrone Verbindungen müssen regelmäßig benutzt werden, da ansonsten die Verbindung wegen eines Timeouts geschlossen wird.
- Alle Asynchronen Verbindungen akzeptieren den *noop* Befehl. Dieser kann als Klartext ohne Format gesendet werden und dient lediglich zum Erhalt der Verbindung.

# Synchron REST (<https://vs.uniks.de/api>)

## POST /user

### Beschreibung:

Registriert einen neuen Nutzer am Server. Der Name muss einzigartig sein.

### Body:

- name: String
- password: String

### Beispiel:

```
> curl -X POST https://vs.uniks.de/api/user -d '{ "name": "Mr", "password": "Spock" }'
```

### Antwort:

```
{  
  "status": "success",  
  "message": "User created",  
  "data": {}  
}
```

## GET /user

### Beschreibung:

Liefert eine Liste aller Benutzer, die sich in der Lobby befinden.

### Beispiel:

```
> curl -X GET -H "userKey: xxxxx" https://vs.uniks.de/api/user
```

### Antwort:

```
{  
  "status": "success",  
  "message": "",  
  "data": [  
    "kali"  
  ]  
}
```

## POST /user/login

### Beschreibung:

Führt ein Login durch.

### Body:

- name: String
- password: String

### Beispiel:

```
> curl -X POST https://vs.uniks.de/api/user/login -d '{ "name": "Mr", "password": "Spock" }'
```

### Antwort:

```
{  
  "status": "success",  
  "message": "",  
  "data": {  
    "userKey": "c653b568-d987-4331-8d62-26ae617847bf"  
  }  
}
```

## POST /user/logout

### Beschreibung:

Führt ein Logout aus.

### Beispiel:

```
> curl -X POST -H "userKey: xxxxx" https://vs.uniks.de/api/user/logout
```

### Antwort:

```
{  
  "status": "success",  
  "message": "Logged out",  
  "data": {}  
}
```

## POST /user/temp

### Beschreibung:

Erstellt einen temporalen Benutzer. Dieser läuft nach 24 Stunden aus.

### Beispiel:

```
> curl -X POST https://vs.uniks.de/api/user/temp
```

### Antwort:

```
{  
  "status": "success",  
  "message": "",  
  "data": {  
    "name": "Hendry Bracken",  
    "password": "Hendry Bracken"  
  }  
}
```

## POST /map

### Beschreibung:

Erstellt eine neue Karte, benötigt dafür ein \*.jpg Datei als Hintergrund.

### Body:

- uploaded\_map: File (Multipart Upload)
- metadata: Object
  - creator: String
  - name: String
  - fields: Object
    - noneFields: String[]
    - waypoints: String[]

### Beispiel:

```
> curl -X POST -H "userKey: xxxxx" https://vs.uniks.de/api/map \  
-F 'metadata={ "creator": "se", "name": "TestMap", \  
  "fields": { "noneFields": ["0;5", "11;2", ...], "waypoints": ["1;2", "1;3", ...]}' \  
-F "uploaded_map=C:/tmp/img.jpg"
```

### Antwort:

```
{  
  "status": "success",  
  "message": "",  
  "data": {  
    "id": "5e14741269dcf01cb837600e",  
    "creator": "se",  
    "name": "map_TestMap ",  
    "noneFields": [  
      "0;5",  
      "11;2",  
      ...  
    ],  
    "waypoints": [  
      "1;2",  
      "1;3",  
      ...  
    ]  
  }  
}
```

## GET /map

### Beschreibung:

Liefert eine Liste aller Karten zurück.

### Beispiel:

```
> curl -X GET -H "userKey: xxxxx" https://vs.uniks.de/api/map
```

### Antwort:

```
{
  "status": "success",
  "message": "",
  "data": [
    {
      "id": "5e14741269dcf01cb837600e",
      "creator": "se",
      "name": "map_asteroid",
      "noneFields": [
        "0;0",
        "1;0",
        "2;0",
        ...
      ],
      "waypoints": [
        "12;15",
        "12;14",
        "12;13",
        ...
      ]
    },
    ...
  ]
}
```

## GET /map/:id

### Beschreibung:

Liefert eine einzelne Karte zurück.

### Beispiel:

```
> curl -X GET -H "userKey: xxxxx" https://vs.uniks.de/api/map/5e14741269dcf01cb837600e
```

### Antwort:

```
{
  "status": "success",
  "message": "",
  "data": {
    "id": "5e14741269dcf01cb837600e",
    "creator": "se",
    "name": "map_asteroid",
    "noneFields": [
      "0;0",
      "1;0",
      "2;0",
      "3;0",
      ...
    ],
    "waypoints": [
      "12;15",
      "12;14",
      ...
    ]
  }
}
```

## GET /map/:id/download

### Beschreibung:

Liefert ein Bild der Karte als Base64 codierter String zurück, ohne JSON.

### Beispiel:

```
> curl -X GET -H "userKey: xxxxx" https://vs.uniks.de/api/map/5e14741269dcf01cb837600e/download
```

### Antwort:

```
/9j/4AAQSkZJRgABAQEASABIAAD/4RUfRXhpZgAASUkqAAgAAAADABoBBQABAAAAMgAAABsBBQABAAAAOgAAACgBAwABAAAAAg...
```

## DELETE /map/:id

### Beschreibung:

Löscht eine selbst erstellte Karte. Es können keine Karten anderer Personen gelöscht werden.

### Beispiel:

```
> curl -X DELETE -H "userKey: xxxxx" https://vs.uniks.de/api/map/5e14741269dcf01cb837600e
```

### Antwort:

```
{
  "status": "success",
  "message": "Map deleted",
  "data": {}
}
```

## POST /game

### Beschreibung:

Erzeugt ein neues Spiel.

### Body:

- name: String
- neededPlayer: Integer
- map: String (Karten ID)

### Beispiel:

```
> curl -X POST -H "userKey: xxxxx" https://vs.uniks.de/api/game \
-d '{ "name": "TestGame", "neededPlayer": 4, "map": "5e14741269dcf01cb837600e" }'
```

### Antwort:

```
{
  "status": "success",
  "message": "",
  "data": {
    "gameId": "5e2ffbd8770dd077d03df505"
  }
}
```



## GET /game

### Beschreibung:

Liefert eine Liste aller Spiele zurück.

### Beispiel:

```
> curl -X GET -H "userKey: xxxxx" https://vs.uniks.de/api/game
```

### Antwort:

```
{
  "status": "success",
  "message": "",
  "data": [
    {
      "id": "5e2ffbd8770dd077d03df505",
      "name": "TestGame",
      "neededPlayer": 2,
      "joinedPlayer": 0,
      "map": "5e14741269dcf01cb837600e",
      "connectedPlayers": ["Alice", "Bob"]
    }
  ]
}
```

## GET /game/:id

### Beschreibung:

Tritt dem Spiel mit der gegebenen ID bei.

### Parameter:

- id: String

### Beispiel:

```
> curl -X GET -H "userKey: xxxxx" https://vs.uniks.de/api/game/5e2ffbd8770dd077d03df505
```

### Antwort:

```
{
  "status": "success",
  "message": "Game joined, you will be disconnected from the chat and the system socket.
Please connect to /ws/game?gameId=GAME_ID",
  "data": {}
}
```

## DELETE /game/:id

### Beschreibung:

Löscht das Spiel mit der gegebenen ID.

### Parameter:

- id: String

### Beispiel:

```
> curl -X DELETE -H "userKey: xxxxx" https://vs.uniks.de/api/game/5e2ffbd8770dd077d03df505
```

### Antwort:

```
{  
  "status": "success",  
  "message": "Game deleted",  
  "data": {}  
}
```

# Asynchron WS (<wss://vs.uniks.de/ws>)

Die Beispiele für die Asynchrone Kommunikation bestehen lediglich aus beispielhaften JSON Nachrichten und der dazugehörigen URL.

## **/system**

### **Beschreibung:**

Auf diesem Channel werden System Nachrichten versendet. Zum Beispiel, ein Spieler hat die Lobby betreten / verlassen oder ein Spiel wurde erzeugt / gelöscht.

Auf diesen Channel können keine Nachrichten gesendet werden.

**Beispiel:** `wss://vs.uniks.de/ws/system`

### **Nachrichtenaufbau (Antworten):**

```
{  
  "action": "<action>",  
  "data": {  
    ...  
  }  
}
```

Aktionen: "userJoined", "userLeft", "gameCreated", "playerJoinedGame", "playerLeftGame", "gameDeleted"

Das Daten Feld variiert zwischen den Aktionen.

## **/chat?user=MyUserName**

### **Beschreibung:**

Auf diesem Channel werden Chat Nachrichten gesendet und verteilt. Es gibt Öffentliche und Private Chats.

### **Query Parameter:**

- user: String

### **Befehle:**

- Nachricht an alle
  - channel: String (all)
  - message: String
- Nachricht an einen bestimmten Spieler
  - channel: String (private)
  - to: String
  - message: String

**Beispiel:** `wss://vs.uniks.de/ws/chat?user=TestUser`

```
{
  "channel": "all",
  "message": "Hallo Welt an alle!"
}
{
  "channel": "private",
  "to": "Bob",
  "message": "Hallo Welt nur an dich!"
}
```

### **Nachrichtenaufbau (Antworten):**

```
{
  "channel": "all" / "private",
  "message": "Hello World!",
  // Nur private
  "from": "Karli",
  "to": "Bob",
}
```

## **/game?gameId=123456789**

### **Beschreibung:**

Auf diesem Channel werden Spiel relevante Nachrichten gesendet. Aktuell ist das der Hinweis darauf, dass die Startsituation gesendet wird. Zur Anmeldung wird die Spiel-ID benötigt.

Es kann aktuell nur der Befehl zum Verlassen des Spieles gesendet werden.

### **Query Parameter:**

- gameId: String

### **Befehle:**

- Chat
  - messageType: String (chat)
    - Nachricht an alle
      - channel: String (all)
      - message: String
    - Nachricht an einen bestimmten Spieler
      - channel: String (private)
      - to: String
      - message: String
- Kommandos
  - messageType: String (command)
    - Spiel verlassen
      - action: String (leaveGame)
    - Spieler ist bereit
      - action: String (readyToPlay)
    - Spiel starten
      - action: String (startGame)

- Turm bauen
  - action: String (buildTower)
  - data: Object
    - fieldId: String
    - towerType: String
- Turm upgraden
  - action: String (upgradeTower)
  - data: Object
    - towerId: String
- Turm verkaufen
  - action: String (sellTower)
  - data: Object
    - towerId: String
- Turmstrategie ändern
  - action: String (changeTowerStrategy)
  - data: Object
    - towerId: String
    - strategy: String
- Vampir beschwören
  - action: String (summonVampire)
  - data: Object
    - vampireType: String

**Beispiel:** wss://vs.uniks.de/ws/game?gameId=5ewrj3kg45345j

```
{
  "messageType": "command",
  "action": "leaveGame" || "readyToPlay" || "startGame"
}
{
  "messageType": "command",
  "action": "buildTower",
  "data": {
    "fieldId": "xxxxxwe342",
    "towerType": "small",
  }
}
{
  "messageType": "chat",
  "channel": "all",
  "message": "Hallo Welt an alle!"
}
{
  "messageType": "chat",
  "channel": "private",
  "to": "Bob",
  "message": "Hallo Welt nur an dich!"
}
```

### Nachrichtenaufbau (Antworten):

Als Ergebnis von Kommandos sendet der Server Datenmodellveränderungen in Form von Feld-Update Nachrichten. Bei einem Spielbeitritt werden neben den erzeugten Objekten, Statusnachrichten gesendet. Der Ablauf ist dabei immer der gleiche *Initialize game -> List of Objects -> Initialize finished*. Die Reihenfolge der Nachrichten richtet sich immer nach diesem Ablauf.

Beispiel:

```
{ "action": "info", "data": { "message": "Initialize game, sending start situation..." } },
{ "action": "gameInitObject", { "id": "Game@234", "name": "MyTestGame", ... } },
{ "action": "gameInitObject", { "id": "Player@567", "currentGame": "Game@234", ... } },
...,
{ "action": "gameInitFinished" }
```