

SOLUTION PROPOSAL FOR SPELGRÄNS

An e-service where gamblers can set their national deposit limits

Abstract

The Spelgräns system is a solution designed to empower gamblers by allowing them to set their daily, weekly, and monthly deposit limits across all licensed casinos in a country. This system aims to prevent excessive gambling and provides a safer gambling environment for players. In this technical solution proposal, the necessary components were presented to bring this system in production. The proposed solution ensures seamless integration with all licensed casinos and provides a user-friendly interface for gamblers to set and manage their deposit limits. With a robust technology, the Spelgräns system is poised to make a positive impact on society.

Björn Berglund
Huseyin Erdinc
Precio Fishbone AB

1. Revision history

A description of added and/or changed content should be added in the table below

Version	Date	Description	Responsible
1.0	2023-04-06	Initial version is published.	Björn Berglund & Huseyin Erdinc

2. Table of contents

1.	Revision history	1
2.	Table of contents	2
3.	Introduction	7
1.1	Purpose and scope	7
1.2	Limitations.....	7
2	Term list	7
3	Time estimates.....	8
4	Stakeholders	9
4.1	Paf	9
4.2	Spelinspektionen in Sweden	9
4.3	System architects	9
4.4	Developers	9
4.5	Infrastructure managers	9
5	The context of the system	10
5.1	Frontend.....	10
5.2	Player	10
5.3	E-services	10
5.4	Licensed operator	10
5.5	Administrator	10
5.6	Authentication service	10
6	Requirements.....	11
6.1	Functional requirements.....	11
6.1.1	Authentication	11
6.1.2	Functionality player	11
6.1.3	Functionality licensed operator	11
6.1.4	Functionality administrator	11
6.1.4.1	Statistical reports	12
6.1.5	Deposit limit rules	12
6.1.6	User integrity	13
6.1.7	Logging	13
6.1.7.1	Logging for API	13
6.1.7.2	Logging for administrator portal.....	14
6.2	Non-functional requirements	14

6.2.1	Performance and response times	14
6.2.2	Security	14
6.2.3	Design.....	15
7	Architectural views	16
7.1	Scenarios	16
7.2	Logical view	16
7.3	Developer view	16
7.4	Physical view	16
7.5	Process view.....	17
8	Logical view	18
8.1	Class diagram	18
8.2	Database diagram	19
8.3	User interface.....	20
8.3.1	Logging in for the first time.....	20
8.3.2	Setting limits for the first time	21
8.3.3	Seeing current & upcoming limits.....	22
8.3.4	Changing existing limits.....	23
9	Process view.....	24
9.1	State diagrams	24
9.1.1	Player's perspective	24
9.2	Sequence diagrams	25
9.2.1	Player	25
9.2.1.1	Login to the e-service.....	25
9.2.1.2	See their active limit configuration.....	26
9.2.1.2.1	Change their limit configuration	27
9.2.2	Licensed operator	28
9.2.2.1.1	RequestDeposit endpoint	28
9.2.2.1.2	FinalizeDeposit endpoint	29
9.2.3	Administrator	29
9.3	Performance	29
9.4	High availability.....	30
9.4.1	Retry functionality of Frontend	30
10	Development view	31
10.1	Design considerations	31
10.2	Contracts for the public facing API	32
10.3	Versioning policies	32

10.4	Deployment.....	32
10.5	Component diagram	34
10.6	Frontend.....	35
10.6.1	Open website	35
10.6.1.1	Caching.....	35
10.6.2	My pages.....	35
10.6.2.1	Caching.....	35
10.7	Operator API	35
10.7.1	RequestDeposit endpoint	36
10.7.2	FinalizeDeposit endpoint	36
10.7.3	Internal endpoints (zPages)	36
10.7.3.1	IP whitelisting.....	36
10.7.3.2	Health check.....	36
10.7.4	Caching.....	36
10.7.5	Logging	37
10.7.5.1	Extra requirements	37
10.7.5.2	Performance considerations	37
10.8	Internal API	37
10.8.1	GetDepositLimits.....	38
10.8.2	UpdateDepositLimits.....	38
10.8.3	Logging	38
10.8.4	Administrator methods.....	38
10.8.4.1	Reporting.....	38
10.9	Load balancer.....	38
10.10	Business layer.....	39
10.11	Data layer	39
10.12	Hangfire.....	39
10.12.1	Nightly job to delete old data	39
10.12.2	One-time jobs to schedule limit increases.....	40
10.12.3	High availability.....	40
11	Physical view	41
11.1	Software stack.....	41
11.1.1	Operative system	41
11.1.2	Web server.....	41
11.1.3	Database	41
11.1.4	Load balancer.....	41

11.1.5	File share (NAS)	41
11.1.6	SMTP server	41
11.2	Hardware stack	42
11.2.1	CMS servers	42
11.2.2	API servers	42
11.2.3	Database servers	42
11.3	Security	42
11.4	Availability	42
11.5	Performance	42
11.6	Scalability	43
11.7	Monitoring	43
11.7.1	Internal monitoring	43
11.8	Application environments	44
11.8.1	Test environment	45
11.8.2	Staging environment	46
11.8.3	Production environment	46
12	Scenarios / use-cases	47
12.1	Conceptual level	47
12.1.1	Users	47
12.1.2	Player	47
12.1.2.1.1	Logging in	47
12.1.2.2	Setting up initial limits	47
12.1.2.2.1	Viewing existing limits	47
12.1.2.2.2	Changing existing limits	47
12.1.2.2.3	Viewing upcoming limit changes	48
12.1.3	Licensed operator	48
12.1.3.1.1	Deposit request	48
12.1.3.1.2	Finalize deposit	48
12.1.4	Administrator	48
12.1.4.1.1	Logging in	48
12.1.4.1.2	Changing the site content	48
12.1.4.1.3	Changing the system settings	48
12.1.4.1.4	Managing the license holders	48
12.1.4.1.5	Creating aggregated reports	48
13	Appendices	49
13.1	Precio Fishbone	49

13.2	Spelpaus.se	49
14	References	50

3. Introduction

Spelgräns concept is brought to discussion by Paf to prevent vulnerable players from playing on unregulated online casino. It is considered as a preventive measure to allow the players to limit their gambling while remaining within the protection of the regulated environment.

1.1 Purpose and scope

This proposal tries to give a solution to the problem by providing comprehensive information of the necessary components which the system consists of and the information flow among them.

1.2 Limitations

This proposal does not cover the following areas in depth:

- **The architecture of the client application:** That is, how the public web page & My Pages will work. This decision can be made best when the UX and UI designs are ready.
- **The architecture for a cloud-based hosting model:** This proposal does not contain the necessary changes which will be required to adapt the system architecture for cloud if a cloud-based hosting model is chosen.
- **The database design for log tables:** It would be more efficient to design the log tables when requirements about logging is finalized.

2 Term list

Term	Description
Authentication	The process of verifying the identity of a computer system user.
API	Application Programming Interface, a way to allow communication between computers or software.
API key	An authentication key which is sent with the API request.
ActorId	A unique identifier to identify a licensed operator.
Player	A private individual who wants to register a deposit limit.
Licensed operator	A gambling operator that has been granted a licence to offer gambling for money in Sweden or Finland.
Administrator	A private individual who is authorized to manage CMS contents and settings.
Deposit limit	National individual limit for gambling covering all licensed operators.
Accumulated deposits	All deposits reported to Spelgräns for a single player.
Available deposit space	Difference between the deposit limit and the accumulated deposits.
EID	E-identification, a digital ID-document that you can use to authenticate yourself securely on, for example, a website or in an app.
CMS	Content Management System
Umbraco	An open-source CMS written in C#.
Umbraco Backoffice	The editor and administrator user-interface of Umbraco. It is used to change the system settings and to manage the content.
Optimizely	A CMS written in C# which requires a license.
Spelinspektionen	The Swedish Gambling Authority

3 Time estimates

The implementation of the proposal described in this document is found below:

Activity	Estimate (hours)
Adjust architecture to final requirements	100
Infrastructure	450
Project management	650
Development – CMS	800
Development – Administrator portal	550
Development - API	450
QA tests	200
Total	3200

The estimates in the above table are not precise estimations and the time needed for a given activity can increase or decrease when the final requirements are set. **To be on the safe side, a buffer of 18% can be added on top of the total, which results to approximately 3780 hours.**

For effective development it is suggested to have two developers for infrastructure and API development, two developers for CMS and administrator portal development, one project leader for project management, and one or two people for QA tests. QA tests can also be done by the project manager. In total, it is recommended to assign 5 – 6 people to this project to complete it within 5 – 6 months.

4 Stakeholders

The following stakeholders have been identified.

4.1 Paf

Paf has an interest in this project because their final mission is to be beneficial to the society. Preventing players from playing on unregulated online casinos without any deposit limiting function harms the society.

4.2 Spelinspektionen in Sweden

Spelinspektionen has an interest in this project because their primary goal is to regulate the gambling industry by making the market legal, safe, and reliable. Spelinspektionen was recently commissioned to research about alternate ways of excluding oneself from the games. (Ericsson, 2022)

4.3 System architects

System architects have responsibilities for steering system solutions towards the best possible architecture, considering business rules and current conditions. System architects has an interest in this document to assess the system's architectural properties.

4.4 Developers

Developers have an interest in understanding the system's architecture and functionality during implementation and further development.

4.5 Infrastructure managers

Infrastructure managers have responsibilities to manage the system's resources and ensuring that they are always healthy. Infrastructure managers have an interest in this document to understand the different infrastructure components in the system and their communications among each other.

5 The context of the system

5.1 Frontend

The public facing website where a player can get information about the e-service and login using their e-identity to configure her daily, weekly, and monthly limit.

5.2 Player

A player is a natural person who must set her gambling limits from Spelgräns e-services.

5.3 E-services

E-services define a protected area within the Frontend, where players are required to identify themselves to access and use the service according to pre-set rules.

5.4 Licensed operator

A licensed operator is a gambling operator that has been granted a licence to offer gambling for money in Sweden and/or Finland.

5.5 Administrator

An administrator is a natural person who can modify the site content, change the system settings, and generate aggregated statistical reports.

5.6 Authentication service

The authentication service is a third-party entity which provides methods to authenticate a Player by using various supported electronic identity (EID) systems.

6 Requirements

6.1 Functional requirements

6.1.1 Authentication

1. Players must be authenticated with an EID to use the e-service.
2. Only licensed operators are allowed to use the Operator API.
3. Licensed operators must be authenticated with actorId and API-key to use the Operator API.
4. Administrators must be authenticated with two-factor authentication to be admitted in the system.

6.1.2 Functionality player

1. A player can set deposit limits for day, week, and month. A weekly deposit limit cannot be lower than a daily deposit limit. A monthly deposit limit cannot be lower than a weekly deposit limit.
2. A player can change one or more of his/her deposit limits for day, week, and month.
3. Before saving changes of the deposit limits the player can see date and time when the changes will come into effect.
4. A player can view his/her current deposit limits for day, week, and month.
5. A player can view his/her current available deposit space.
6. No historical information or reports regarding events related to the service is available for the player.
7. A player must be able to make an explicit logout from the service.
8. A player must be automatically logged out from the service after 1 hour of inactivity.

6.1.3 Functionality licensed operator

1. A licensed operator can use the Operator API to request a deposit permission if a given amount is smaller or equal to the current available deposit space for a player.
2. A licensed operator can use the Operator API to finalize a deposit request for a player.

6.1.4 Functionality administrator

1. An administrator can configure the maximal deposit limits allowed in the service for day, week, month.
2. An administrator can edit content in the interface for the player service.
3. An administrator can add a licensed operator.
4. An administrator can edit a licensed operator.
5. An administrator can remove, activate, or inactivate a licensed operator.
6. An administrator can add and remove IP-addresses in the operator API access whitelist (see non-functional requirements).
7. When adding a new licensed operator, the user interface must generate two secure authentication keys for license holders. An administrator must be able to edit these keys.
 - a. A secure authentication key consists of 64 characters using upper case letters, lower case letters, and digits.
8. An administrator must be able to make an explicit logout from the administration interface.
9. An administrator must be automatically logged out from the administration interface after 30 minutes of inactivity.
10. An administrator must be able to change system settings such as maximum deposit limits and data retention periods.

6.1.4.1 Statistical reports

1. An administrator must be able to create an aggregated report showing players' average daily, weekly, and monthly deposit amount grouped by birth year and gender.
2. An administrator must be able to create a report showing license holders' requests for a given personal number. The report can be created by the following filter parameters:
 - Personal number (obligatory)
 - License holder (one, many, or all)
 - Date from & date to
 - API method

The report will show the following data:

- Timestamp of the request
- API method
- Response of the request
- License holder
- Request Id & Response Id

3. An administrator must be able find a specific request by following parameters:
 - Date from & date to
 - Request Id & Response Id

The report will show the following data:

- Timestamp of the request
- License holder
- API method
- Time taken to generate a response
- Request Id & Response Id

4. An administrator must be able to create a report showing the audit log of activities which were done in the administrator portal by following parameters:
 - Administrator name
 - Function
 - Date from & date to

The report will show the following data:

- Timestamp of the activity
- Administrator name
- Function
- Changes made (if any settings were made)
- Search parameters (if any search parameters were used)

6.1.5 Deposit limit rules

1. The maximum deposit limit in the system is SEK 10.000.000 or EUR 1.000.000. The system must be able to handle multiple currencies but only one currency can be active at a time.
2. Possible currencies are initially SEK and EUR, additional currencies must be possible to add.
3. Deposit limits are saved in the system as öre or eurocent depending on selected currency.
4. Deposit periods are Gregorian calendar based:
 - a. a day starts at 00.00 date n and ends when next day starts at 00.00 date n+1

- b. a week is an interval of seven days and begins on Monday 00:00
 - c. a month is a calendar month
- 5. If a player wishes to change one of the set deposit limits, the change shall come into effect immediately if it relates to a decrease.
- 6. If a player wishes to increase one or more of the deposit limits, the change shall come into effect earliest after 72 hours.
- 7. An increased limit may not take effect before the current period (day/week/month) has passed:
 - a. Daily limits come to effect when 72 hours has passed since the daily limit was changed.
 - b. Weekly limits come to effect 00:00 after both of the following conditions are fulfilled:
 - i. 72 hours has passed since the weekly limit was changed.
 - ii. Sunday 23:59:59 has passed since the weekly limit was changed.
 - c. Monthly limits come to effect 00:00 after both of the following conditions are fulfilled:
 - i. 72 hours has passed since the monthly limit was changed.
 - ii. The calendar month has changed since the monthly limit was changed.
- 8. Example:
 - a. Player submits an increase for her daily, weekly, and monthly limits at 15:12 on Tuesday, January 31st
 - b. Daily limit is increased 15:12 4th February (72 hours minimum)
 - c. Weekly limit is increased 00:00 6th February (Monday following week - if at least 72 hours away)
 - d. Monthly limit is increased 00:00 February 4th (since it must be at least 72 hours since request)
- 9. In all cases the sum of all deposits is still counted from the start of the period – regardless of the time the limit was changed

6.1.6 User integrity

- 1. No one except the player is allowed to view the deposit limits for an individual player.
- 2. Deposit events are saved for 31 days.
- 3. All player data will be erased from the service if the player has not logged in and no license operator has sent any requests about the player for a period equal to the length of the retention period set in the system settings. This retention period cannot be lower than 31 days.

6.1.7 Logging

The following requirements are suggestions and are prone to change when the requirements are finalized.

6.1.7.1 Logging for API

The following will be logged for Operator API:

- Timestamp of the request
- Timestamp of the response
- ActorId
- RequestId & Responseld
- Method

- Deposit amount requested
- Authorization code if exists
- Error if exists

6.1.7.2 *Logging for administrator portal*

For every action an administration does in the administrator portal the following will be logged:

- Timestamp
- Action
- Administrator name
- Changed values if anything has been changed
- Search parameters if exists

6.2 Non-functional requirements

6.2.1 Performance and response times

1. Player interface:
 - a. The service must initially handle at least 5 million registered players.
 - b. The service must initially handle at least 250 concurrent logged in players.
2. Response times for operator APIs
 - a. API response time refers to the duration between the moment when an API receives a request and the moment when it returns a response. It is the total time it takes for the application to process the incoming request, retrieve the necessary data, perform any necessary calculations or operations, and construct the response. It does not take network delays in consideration.
 - b. The API must initially handle at least 100 calls per second in peaks.
 - c. Average response time for the API must not exceed 500 milliseconds.
 - d. The response time of individual calls to the API must not exceed 1000 milliseconds.
3. There should be support for regular automated testing of the functionality of the APIs by querying the APIs and verifying that:
 - a. An answer is received.
 - b. The answer is correct.
 - c. The answer is within the stipulated time limits.

6.2.2 Security

1. All information related to individual players, held, or transmitted, must be encrypted by using both in-transit (SSL) and at-rest encryption methods.
2. Access to the administration interface and the operator APIs must be protected using whitelisting on IP address or subnet level, where only whitelisted IP addresses or subnets must be allowed through.
3. The system must be redundant, meaning that several instances of the System must be running at the same time to minimize any downtime. The various instances and/or their host operating systems must be able to be maintained with, for example, updates independently of each other.
4. The operator APIs must be GEO redundant, meaning that they must be in at least two (2) different physical locations.
5. All storage of social security numbers must be done by storing a checksum. Social security number must be combined with a salt before a checksum is produced. No social security numbers may be stored in plain text.
6. Security and access logs must be protected against manipulation and unauthorized access.

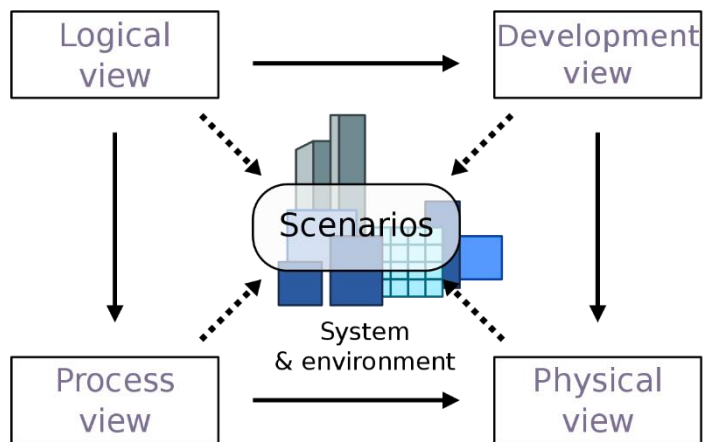
7. When handling personal or system data, the System must allow traceability of who processed the information, when, what information was processed and what the processing consisted of. Examples of processing are reading, writing, changing, and deleting data.
8. The only exception to storing social security numbers using checksum is the logs that must be saved for searches in the Administration interface. In these logs, social security numbers must be stored encrypted.

6.2.3 Design

1. All graphical interfaces must fulfil the standard EN 301 549 “Accessibility requirements for ICT products and services”.
2. All graphical interfaces must be built responsively to fit different formats on screens.
3. The system must have a flexible structure so that the number of possible players can be increased and decreased based on changes if needed.
4. The system must be scalable so that it can handle increased numbers of requests and increased number of players.

7 Architectural views

This proposal uses 4 + 1 architectural view model:



7.1 Scenarios

Also known as	Use case view
Purpose	Illustrates the use cases of the system.
Diagram types	Use case diagrams
Stakeholders	End users

7.2 Logical view

Also known as	Structural view
Purpose	Illustrates the functional requirements the system provides to end-users.
Diagram types	Object diagrams, class diagrams, composite structure diagrams
Stakeholders	Analysts, designers

7.3 Developer view

Also known as	Implementation view
Purpose	Illustrates the system from a developer's perspective.
Diagram types	Package diagrams, component diagrams
Stakeholders	Software management

7.4 Physical view

Also known as	Deployment view
Purpose	Illustrates the system from a system engineer's perspective.
Diagram types	Deployment diagrams, network topology
Stakeholders	Software architects, infrastructure engineers

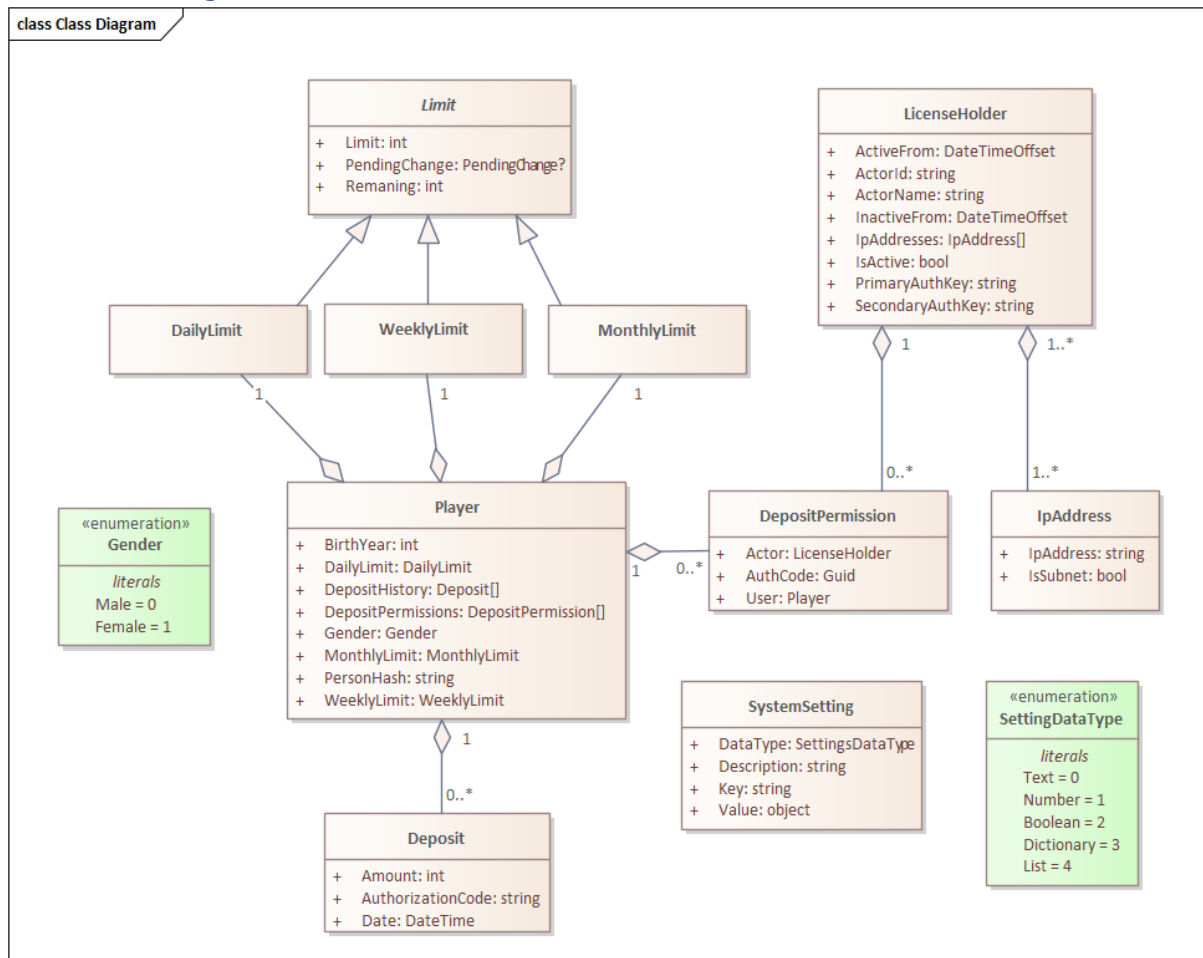
7.5 Process view

Also known as	Behaviour view
Purpose	Illustrates the system processes and how they communicate.
Diagram types	Sequence diagram, communication diagram, activity diagram
Stakeholders	System integrators

8 Logical view

This chapter describes how the functional requirements are filled.

8.1 Class diagram



Spelgräns system will have three main classes which defines players, license holders, and system settings.

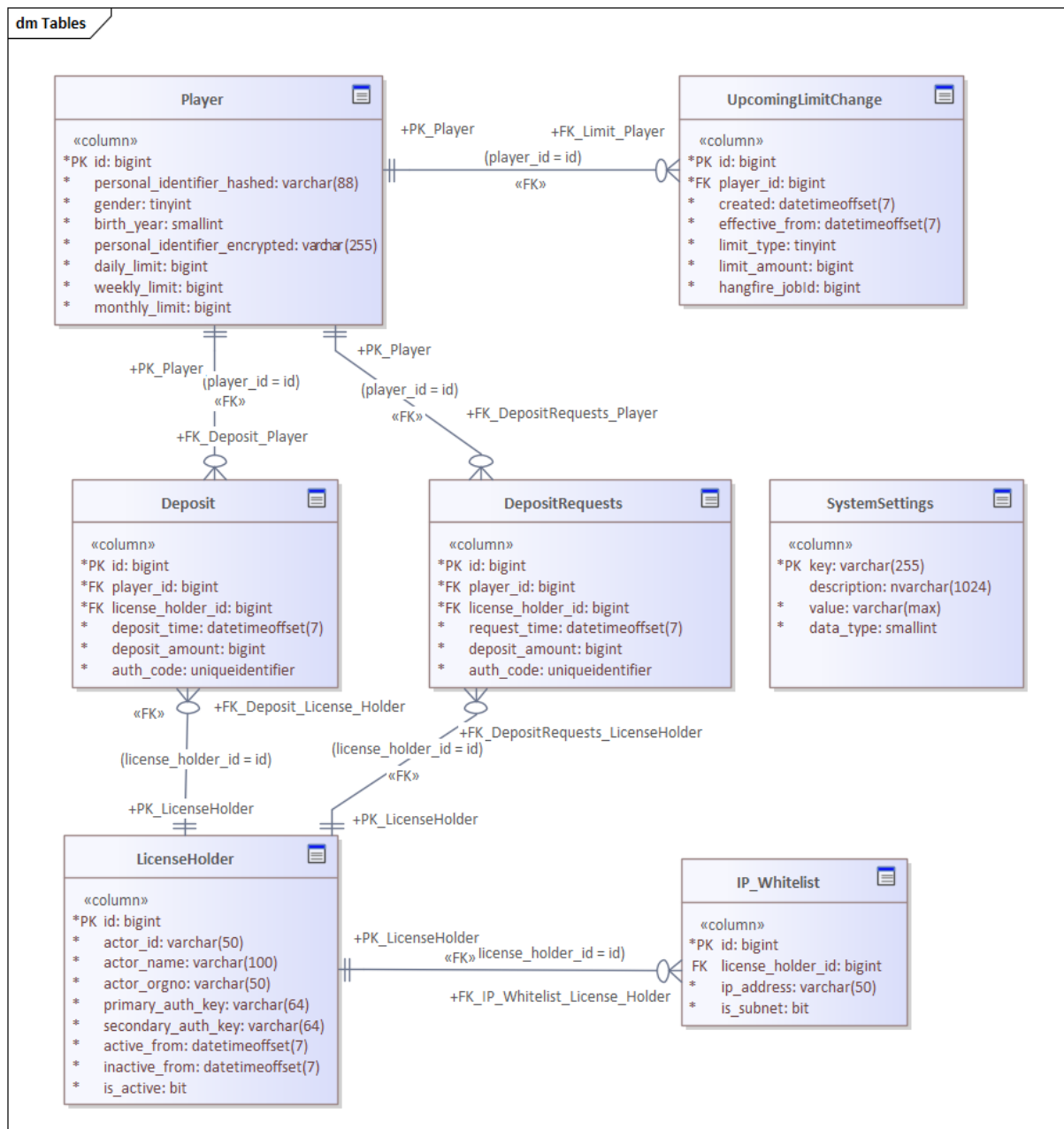
The player class encapsulates the relevant data of player's current situation to show the relevant information on Frontend. This information will be player's deposit limits and remaining deposit space. Information regarding player's age and birth year will only be used to generate aggregated statistical data.

The license holder class holds the information to authenticate a license holder to determine if it has permissions to query the Operator API.

The system setting class denotes a single system setting which holds information about a particular setting of the system. These settings can define configurable parts of the system from maximum deposit limits per period to retention periods of log data.

8.2 Database diagram

Below is the preliminary database diagram for Spelgräns. Note that log related tables are not available in the diagram as it has been stated in 1.2 Limitations section.



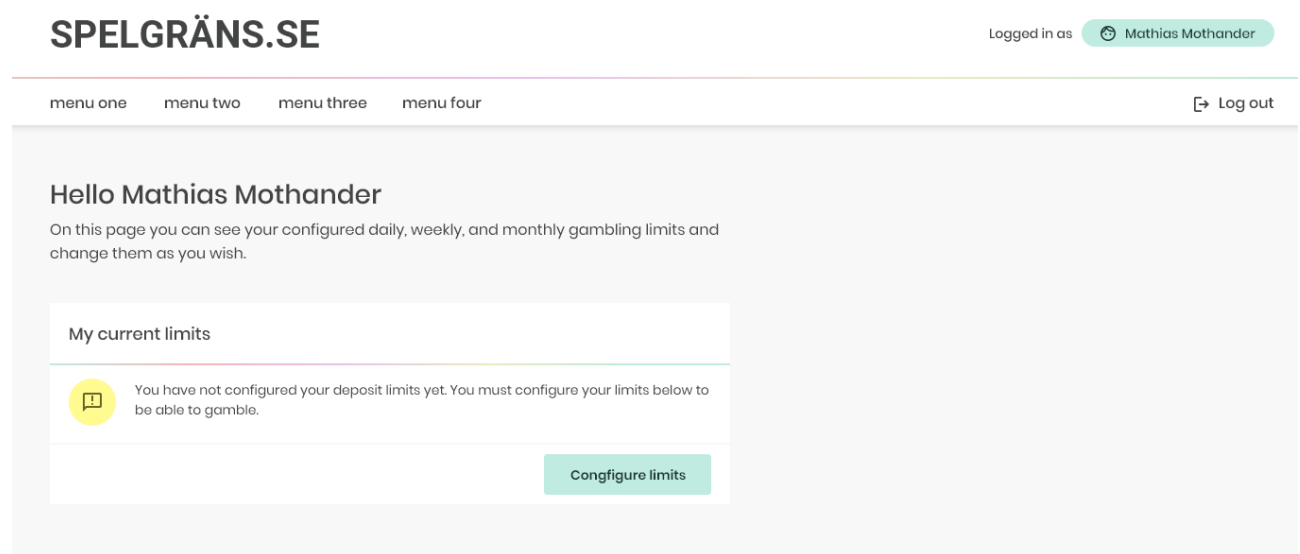
8.3 User interface

Below can be found user interface drafts of the 'My pages' section of Spelgräns. These materials should be considered as reference material which shows the information available on 'My pages'. They are not to be seen as a final design of the user interface and interactions.

Please note that the following user interface designs are cropped but the high-resolution versions of them will be delivered as a separate file. Alternatively, the design can be found from the following URL: <https://xd.adobe.com/view/3942b1e7-c18e-4c20-bba0-4b2901e694c8-6c11/>

8.3.1 Logging in for the first time

When a user logs in to 'My pages', a message will appear stating that the user has not yet configured their deposit limits and must do to continue gambling.



8.3.2 Setting limits for the first time

Clicking on 'Change limits' button will replace the view with the following component. Each period can be configured separately, with the default value being zero. Textboxes will only accept digits as input and the maximum accepted value per textbox will be 1,000,000 EUR or 10,000,000 SEK.

SPELGRÄNS.SE

Logged in as  Mathias Mothander

menu one menu two menu three menu four Log out

Hello Mathias Mothander

On this page you can see your configured daily, weekly, and monthly gambling limits and change them as you wish.

Configure my limits

Daily limit (SEK)

Current	New limit
Not set	500

Weekly limit (SEK)

Current	New limit
Not set	2000

Monthly limit (SEK)

Current	New limit
Not set	5000

Cancel Save changes

8.3.3 Seeing current & upcoming limits

If a user has already defined deposit limits, they will be shown along with the used limit. A progress bar showing the percentual equivalence of the used amount will be available as a visual aid. Users will have an option to change these limits at any time by clicking on 'Change limits' button.

A decrease in limit will be effective immediately, therefore it's not possible to list the action under upcoming changes. An increase in limit, however, will be scheduled to a future point in time and the user will be able to see the scheduled changes. It will not be possible for the user to cancel this pending change.

SPELGRÄNS.SE

Logged in as  Mathias Mothander

menu one menu two menu three menu four

Log out

Hello Mathias Mothander

On this page you can see your configured daily, weekly, and monthly gambling limits and change them as you wish.

 Your limits have successfully changed

My current limits

Daily limit

120 / 500 (SEK)

Weekly limit

890 / 2000 (SEK)

Monthly limit

890 / 4000 (SEK)

Configure limits

Upcoming limit changes

Daily limit (SEK)

Current	New limit
Not set	500

 Your new daily limit will be enabled 2023-04-01 08:00

8.3.4 Changing existing limits

Changing existing limits will give a similar experience as setting up the limits for the first time. However, the users will be able to see their current limit and deposit status next to the text inputs. It is not possible to increase any limits if there are pending changes. Decreasing limits is always possible.

Depending on the action the user does, an information text will be shown under the textboxes:

- Decreasing the limit will inform that the change will be effective immediately.
- Increasing the limit will inform the exact date and time the change will take place.

If the user chooses to decrease a particular limit while there is a pending limit increase, the pending limit change will be deleted.

Clicking on 'Save changes' will save the changes and redirect the user to the default view.

SPELGRÄNS.SE

Logged in as Mathias Mothander

menu one menu two menu three menu four [Log out](#)

Hello Mathias Mothander

On this page you can see your configured daily, weekly, and monthly gambling limits and change them as you wish.

Configure my limits

Daily limit (SEK)

Current
120 / 500

New limit
750

Your daily limit will be set to 750 SEK after 72 hours.

Weekly limit (SEK)

Current
890 / 2000

New limit
2000

Monthly limit (SEK)

Current
890 / 5000

New limit
4000

Your monthly limit will be set as 4000 SEK immediately.

Cancel

Save changes

9 Process view

This chapter describes how the non-functional requirements are filled within the software itself.

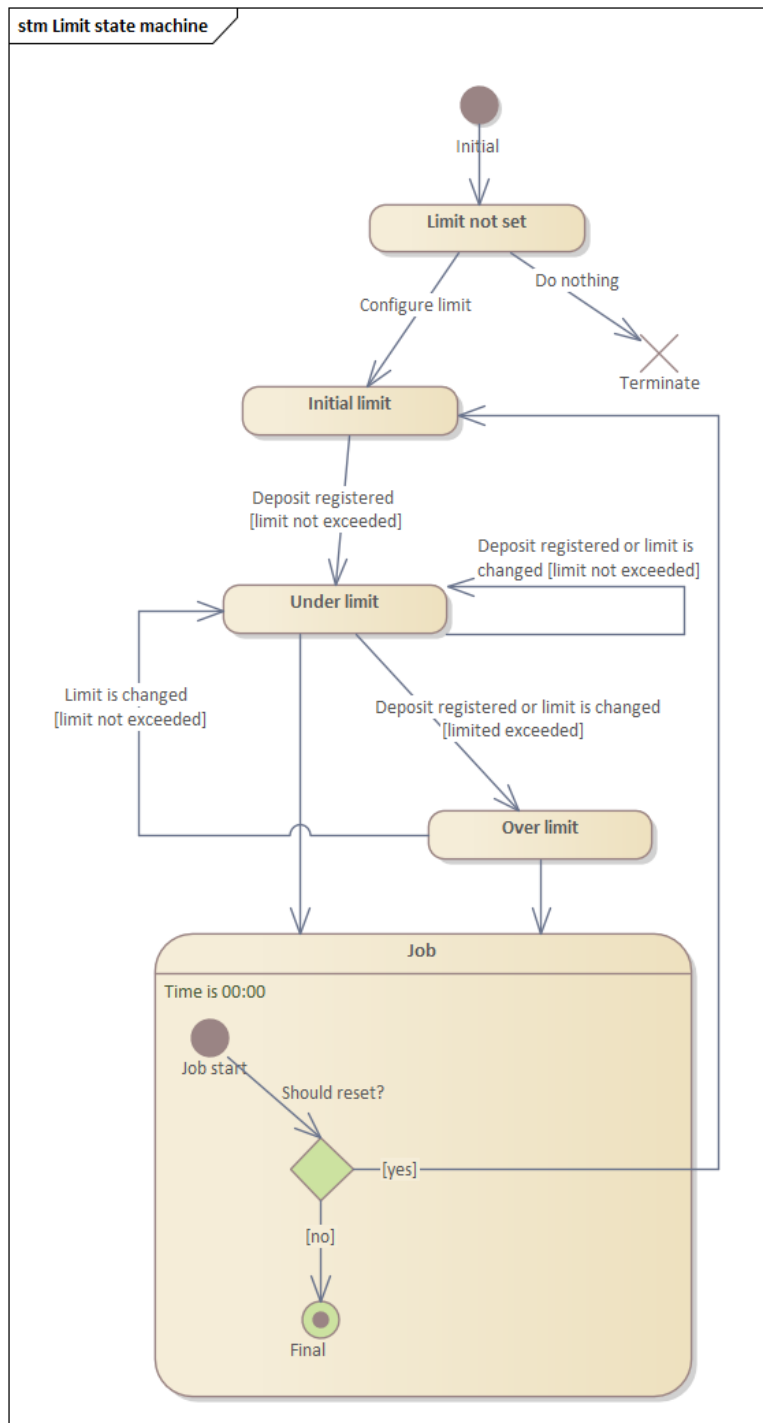
9.1 State diagrams

The state diagrams of the important objects in the system can be found below.

9.1.1 Player's perspective

The following state diagram shows how the player's period limit is changed with different events.

The diagram does not focus on representing all possible period limits but rather focuses on a single period limit since they act the same.



9.2 Sequence diagrams

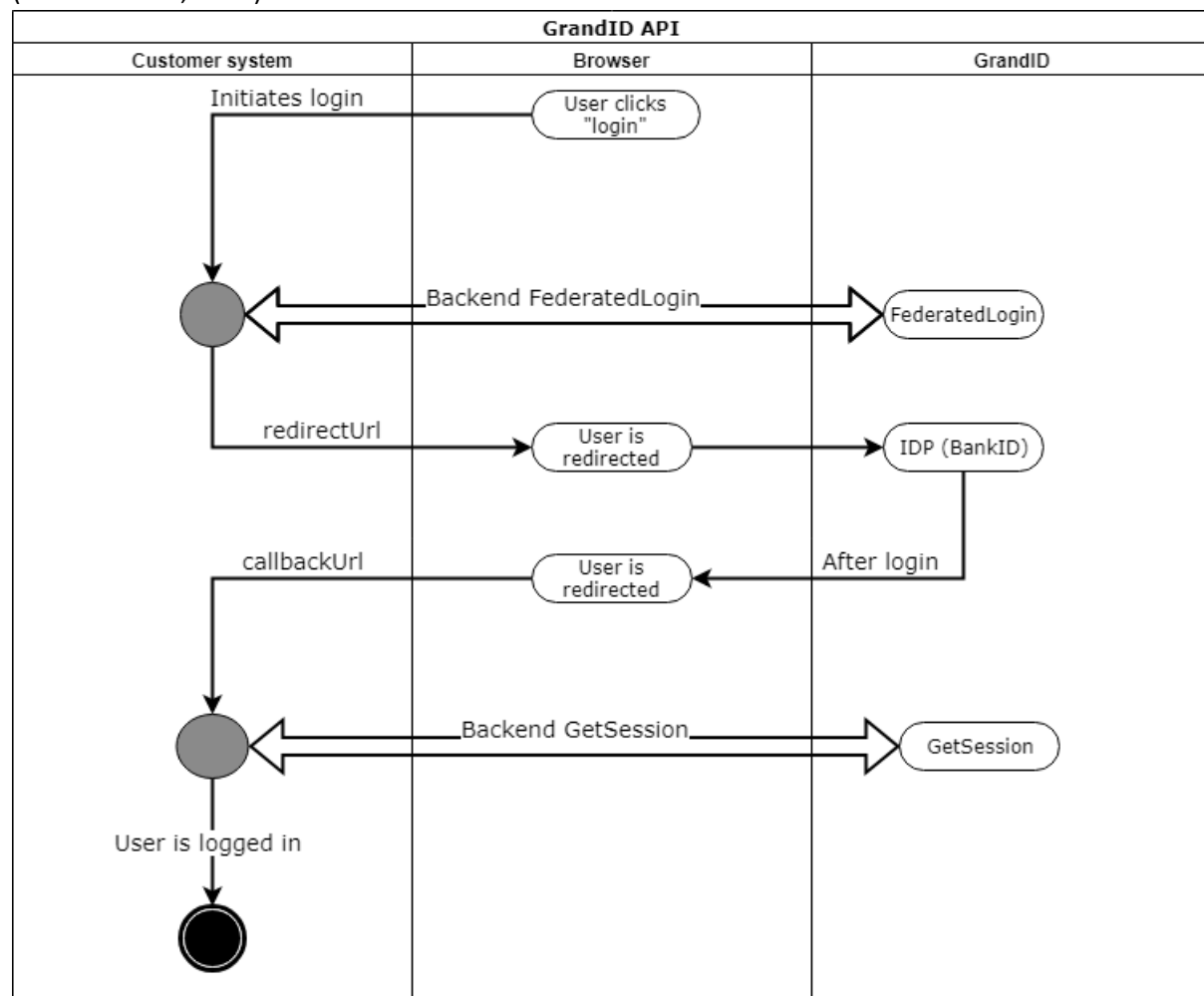
9.2.1 Player

Sequence diagrams which describe players' use-cases are listed below.

9.2.1.1 Login to the e-service

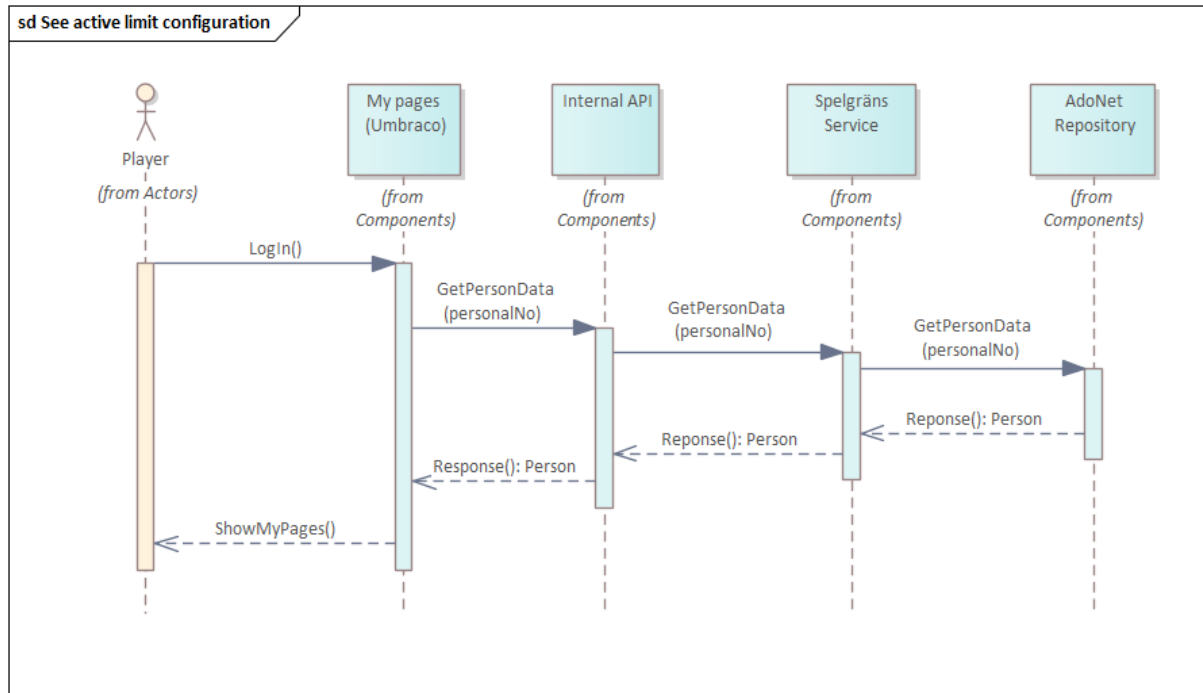
Logging to the e-service requires an electronic identity. The following diagram, taken from GrandId API documentation, shows the sequence diagram of the authentication.

(GRANDID API, 2023)



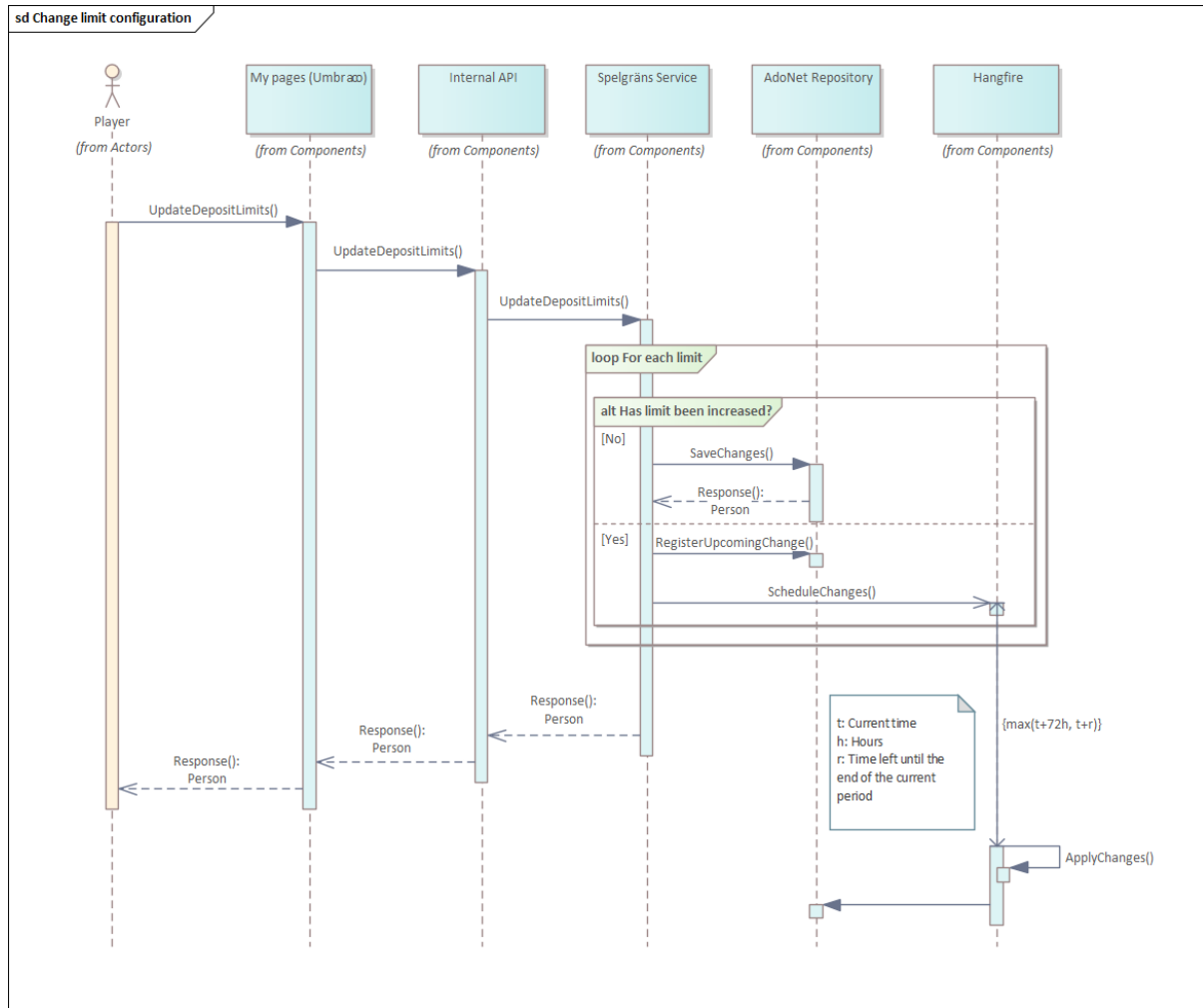
After authenticating a user, a session will be created for the user. The session will have a unique identification string as well as user related data such as name and surname, encrypted personal number, and other information which may be displayed on Frontend. This session will be saved to a distributed cache to keep the application stateless. The session identification string will be then encrypted and saved as a cookie.

9.2.1.2 See their active limit configuration



When a user logs in to My pages, Frontend will contact to Internal API by HTTP and ask for the user data. The call chain will end up in the repository class which in turn will get the user data from the database and return it to the calling system. This operation must not alter any user data in the database.

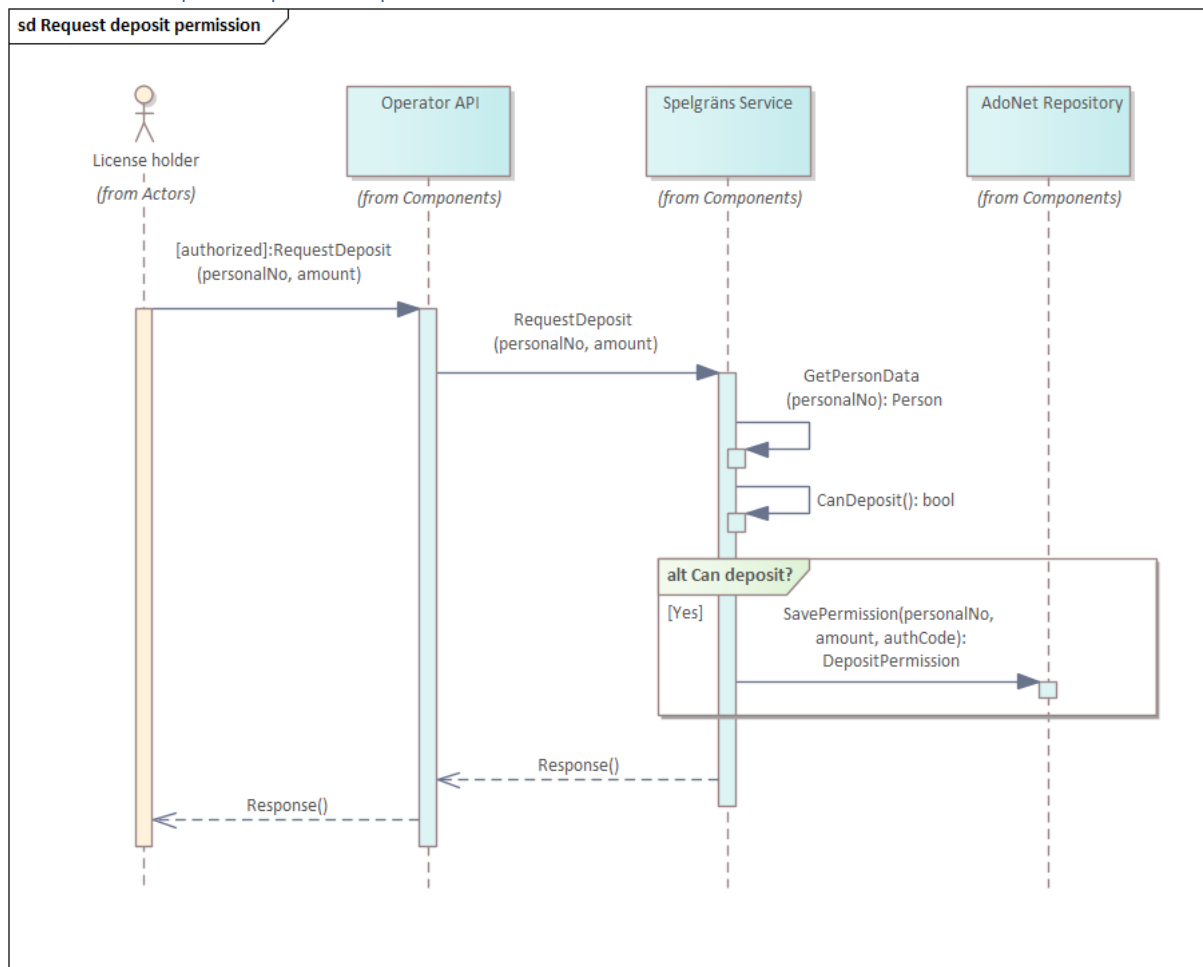
9.2.1.2.1 Change their limit configuration



When a user wants to modify his/her existing deposit limits, a request including new deposit limits will be sent to Internal API via HTTP. Internal API will call the relevant method in the service class to initiate the update operation. It will be service class's responsibility to check whether any deposit limit has been increased. Any limits which were decreased will be saved to the database directly. For those limits which were increased, the service class will calculate the date and time the new limit can be applied according to the business rules and schedule the changes with Hangfire.

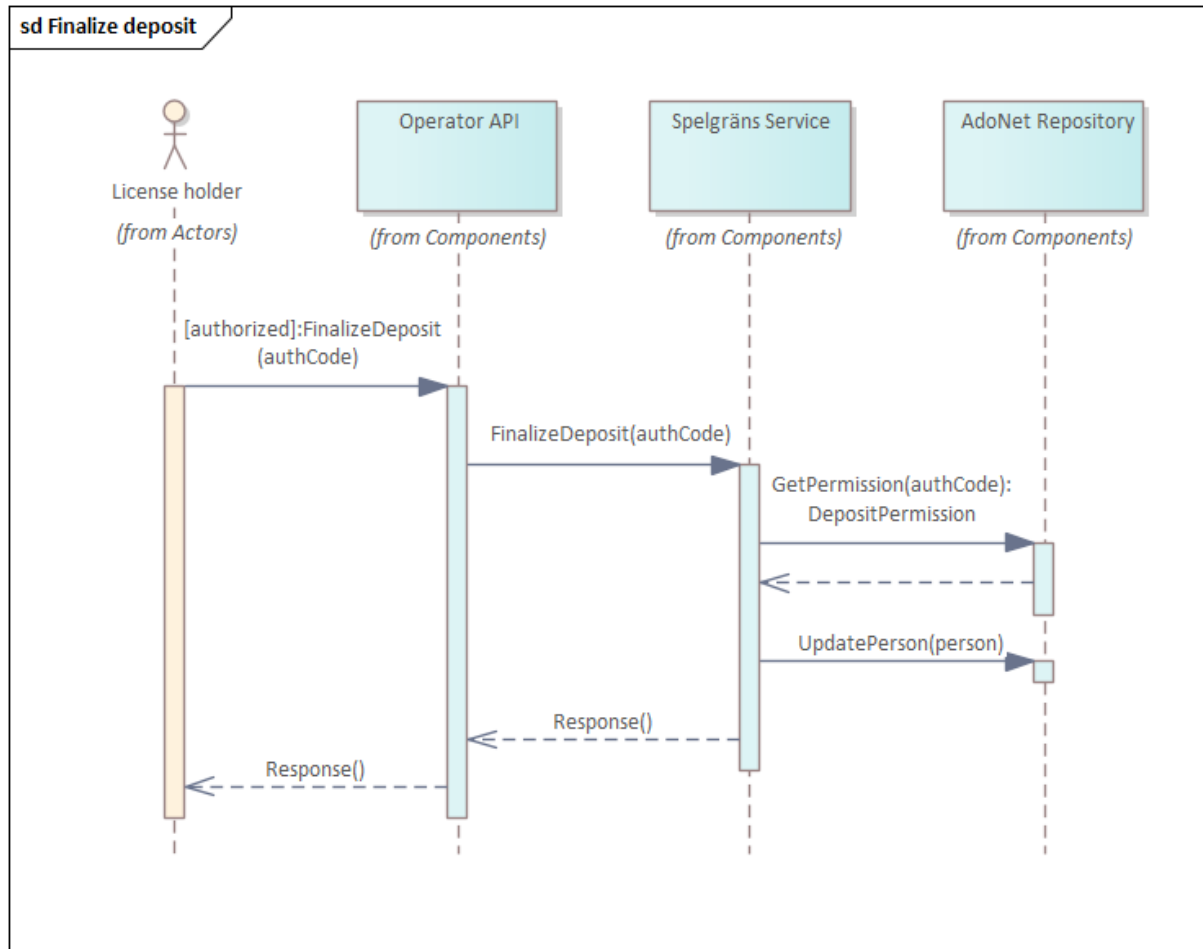
9.2.2 Licensed operator

9.2.2.1.1 RequestDeposit endpoint



When a user initiates a deposit operation on a system which is owned by an authorized license holder, the license holder in question must get a deposit permission by calling the RequestDeposit endpoint and sending the personal number of the user who wants to deposit money into her account as well as the deposit amount. The deposit amount must be either in eurocents or öre. The Operator API will receive this call and forward it to the service class. The service class will ensure that the person has enough deposit space left which will cover the requested amount. If the user is allowed to deposit money, the request will be saved in the database along with an authorization code. The authorization code will then be sent as a response to the license holder. The license holder must only withdraw the amount from the user's bank account if the authorization code exists in the response.

9.2.2.1.2 FinalizeDeposit endpoint



Once the license holder has verified the deposit is successful, it must register the deposit by calling the **FinalizeDeposit** endpoint. The only parameter needs to be sent is the deposit authorization code. The system will then find the relevant deposit request which is tied to the deposit authorization code and register the deposit to the player's account.

9.2.3 Administrator

The sequence diagrams for administrators are omitted in this proposal.

9.3 Performance

The Operator API is expected to handle initially 100 concurrent calls per second at peak. By not using sticky session and configuring the load balancer to distribute incoming requests according to the least connections algorithm, it is expected that each API server is able to handle 50 concurrent calls per second. Since the API servers will connect to database only for handling the business logic and nothing else, they are expected to manage this load without any problems.

The Frontend is expected to handle 250 concurrent logged in users per second. Assuming most of the online users will only initiate read operations from the database to check their current deposit status, it won't generate too much load on the database.

Consider using cluster mode as it's described in 11.1.3 Database if there's a bottle neck at the database communication.

9.4 High availability

The system will be hosted in two different geographically redundant zones, minimizing the risk of down time should any unforeseen event causes downtime in one of the zones. This implies that every service must be installed in both zones and must be stateless.

9.4.1 Retry functionality of Frontend

In certain conditions such as doing a release for the Internal API on a server, the API will be offline for a moment and the Frontend won't be able to connect to it. To always ensure high availability, Umbraco must retry sending the request to the other Internal API instance should the instance installed on the same server be unresponsive.

10 Development view

This chapter focuses on general information about the software, how the code packages are version controlled, and the software packages the solution has as well as how each subsystem communicates with each other.

10.1 Design considerations

Spelgräns system will be developed with the latest non-beta .NET version available at the time of the development work. This will ensure that the system will get the latest performance optimizations as well as security updates.

The system design was deliberately kept as simple as possible to make future changes as easy as possible. Below can be found the design alternatives that were considered but not chosen for implementation:

- **Hosting model:** Hosting model was one of the first decision that was taken. An evaluation of cloud solutions and non-cloud solutions (such as traditional virtual private servers) was made, and it has been decided that although the cloud solution would be much easier to host and deploy, the nature of this project won't allow cloud solutions due to GDPR rules and the sensitivity of the data. Therefore, it has been decided that the virtual servers from Finland or Sweden must be used. However, it is known that the European Union and the United States of America, where Microsoft Azure and Amazon AWS are based on, are working on a deal, namely EU – U.S. Data Privacy Framework, which will provide for binding safeguards that limit access to data by US intelligence authorities to what is necessary and proportionate to protect national security (European Commission, 2022). If this deal is signed, Spelgräns project can be hosted on a cloud-based solution. However, if it is chosen to use a cloud-provider, it is advised to restructure the architectural design of the project to make it more cloud-native. It should be noted that the hosting cost of the project may be higher in the long run compared to non-cloud solutions and there is a risk of vendor lock-in if cloud specific products are to be used. Such vendor lock-ins may make it harder and more expensive to move out of the chosen cloud provider.
- **Caching:** Different caching solutions for Operator API were evaluated such as ASP.NET in memory cache and Redis. Each one of them had their own disadvantages:
 - **Memory cache:** Although this approach is very simple to implement, it would not be easy to invalidate cache of other instances, which in turn would create inconsistent responses depending on which server the request is handled on.
 - **Redis:** Redis would solve the cache invalidation problem but hosting the application in two different zones would mean that minimum of three Redis instances were required to have them in cluster mode. This would possibly increase the hosting cost by increasing the number of virtual servers needed.
- **Actor model:** Microsoft Orleans¹ is a virtual actor system which is used for distributed systems. It was considered to act as the backbone of the system due to what it is capable of:
 - **Concurrency:** It abstracts away the concurrency problems the application may have had such as registering multiple deposits for a user at the same time, exceeding the deposit limit. However, the requirements do not state that such case must be handled.
 - **Caching:** Due to the life cycle of an actor, the application would have had an automatically managed memory cache for the active players (each player could be

¹ <https://learn.microsoft.com/en-us/dotnet/orleans/>

represented as an actor). The duration which defines the activeness of an actor is adjustable, therefore it would provide an increase in the response times.

- **Distributed:** After configuring the system, each virtual actor can be created in whichever virtual server, which is available, transparent to the developer and to the users of the system. The communication between virtual servers is done by remote procedure calls.

The actor model was not chosen because the concept is not very beginner friendly, and the goal was to keep the system design as simple as possible. Moreover, it would fit perfectly for a cloud hosting model where the number of application instances could be increased and decreased dynamically depending on the load. Using it for on premise hosting model would limit its distributed computing capabilities to two servers.

- **Queue system:** Adding a queue system between the CMS and the Internal API was considered to make the system more resilient to database failures. This would, as in Redis consideration, require at least three instances of a queue instance to create a cluster; increasing the overall hosting & maintain costs.

10.2 Contracts for the public facing API

The public facing API of Spelgräns will have two endpoints. These endpoints are responsible to initiate a deposit request and to finalize an already initiated deposit request. Called 'RequestDeposit' and 'FinalizeDeposit' respectively, a license holder must call both methods during their workflow of depositing money from player's account to their own account.

The first method, 'RequestDeposit', is used to check whether a given user has enough deposit space left for all three periods. If and only if the user has deposit space left for all the periods, the endpoint will return a true response along with a so called 'deposit authorization code'. If the endpoint returns false, license holders must not continue the depositing process.

The second method, 'FinalizeDeposit', is used to finalize the deposit request with its authorization code. This endpoint will decrease the user's balance for all periods. If this endpoint gives an error after a successful deposit operation, the license holder must retry finalizing the deposit at some later time.

The Swagger definition of the public facing API will be delivered as a separate file.

10.3 Versioning policies

The development code will be available in a Git repository and Atlassian's the "Git Feature Branch Workflow" strategy² will be adopted.

10.4 Deployment

According to the requirements, the system must be online all the time, even during the deployments. This will be possible by following a well-tested deployment routine for every application. This requires automated build & release pipelines in some external system such as Azure DevOps. Every deployment step must be automated. That is, a deploy must be able to be initiated from a user interface without connecting to a Remote Desktop. This requires an agent to be installed on servers. One such routine may look as follows:

1. The application is built with a build pipeline.
2. A release is initiated for the Test environment.

² <https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow>

3. The configuration is transformed for the environment.
4. The application is deployed on IIS by the agent.

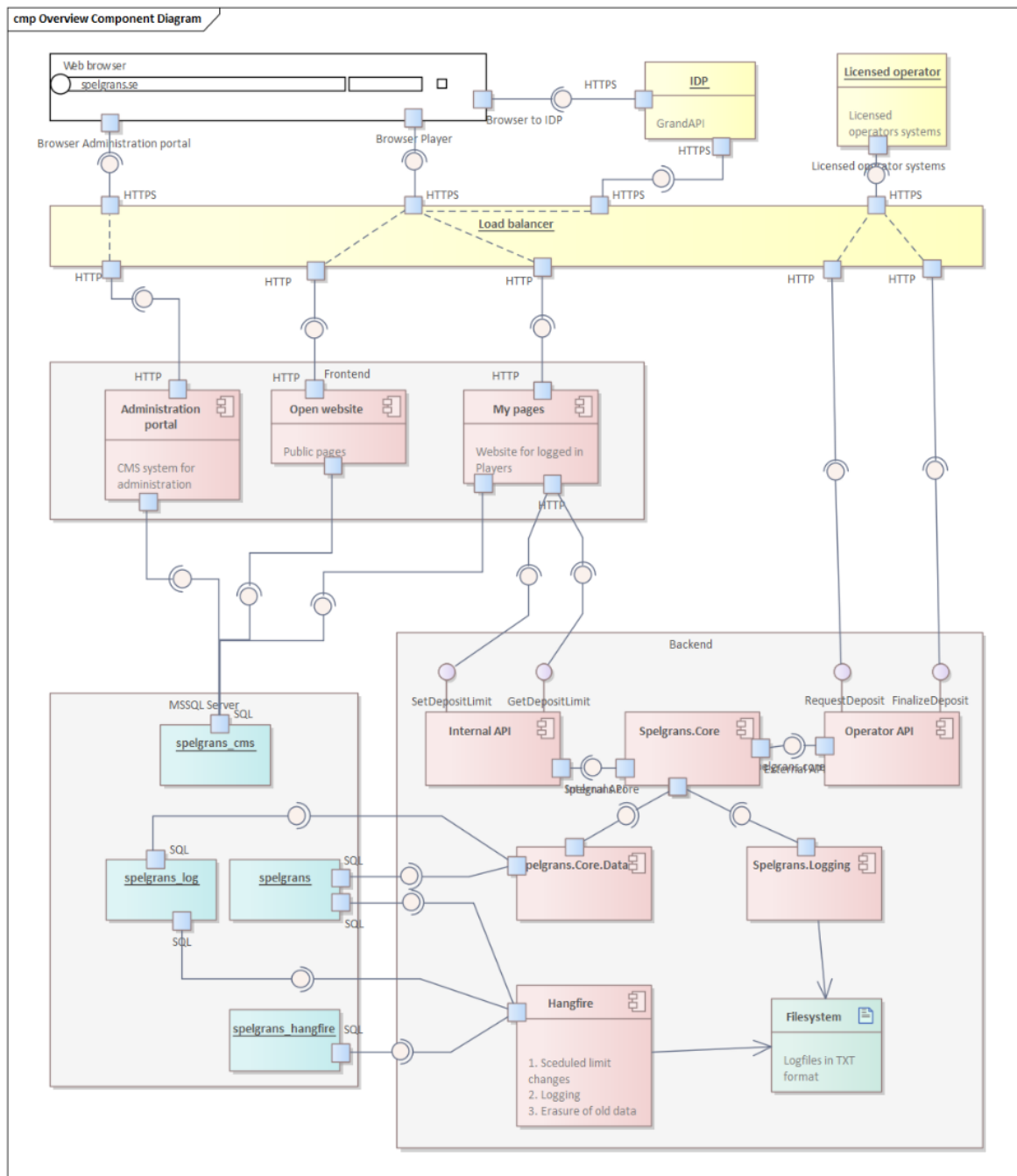
This process should take shorter than a minute if done correctly. For staging and production environments, different steps must be taken to mark the application as offline so that the load balancer won't send any requests to it during the deployment process:

1. Mark the application as offline by changing the content of the text file which /healthz endpoint checks.
2. Wait until the load balancer marks the application as offline. The max duration of the waiting time will be as follows:

$$d_{Max} = (\text{load balancer's health check interval} + \text{load balancer's health check timeout period}) * \text{error count before marking the application offline}.$$
3. The configuration is transformed for the environment.
4. The application is deployed on IIS by the agent.
5. There is no need to mark the application as online since a new deployment will overwrite the contents of the text file used by the /healthz endpoint.
6. Repeat the same procedure for the other instance.

10.5 Component diagram

The components which the system includes and their relations with each other can be seen below.



10.6 Frontend

The frontend will be a .NET based content management system (CMS) which the visitors of the webpage will land on to get information about the system (Open website) and to set and see their deposit limits (My pages). In this chapter, Frontend will be term used to define a CMS system with its different areas; namely open website, user area or 'My pages', and lastly the administration area.

There are two different products that are identified which can be used as Frontend. The first one is Optimizely³ and the second one is Umbraco⁴.

Optimizely is a powerful .NET based CMS which offers an extensive customizability and an easy editor interface to modify the website's content. It requires a license to use, and new licenses are only available for their so-called Digital Experience Platform which is hosted on Microsoft Azure. However, existing on-prem licences continue to work. If there is an existing Optimizely license, it can be used to develop the Frontend with Optimizely.

Umbraco is an open-source .NET based CMS which will not incur any additional costs to the project. Although it has similar capabilities to Optimizely's, Umbraco's editor experience is not as good.

The rest of the proposal will name Umbraco as the chosen CMS, but it must be noted that everything is also applicable to Optimizely, if Optimizely is chosen as the CMS.

10.6.1 Open website

Open website will be the public facing part of the Spelgräns website, where the visitors can get any information deemed as necessary by the providers of the service. The content of open site will be modifiable freely by CMS administrators.

10.6.1.1 Caching

Response caching will be enabled for the open website to ensure fast delivery of content while using minimum server resources.

10.6.2 My pages

My pages will be a part of the open website, but will be available to those who identify themselves with an electronic ID. The authentication will be done using GrandID service and the logged in users will be able to see and configure their limits, see the sum of their deposits for a given period, and see their upcoming limit changes if they have increased limits recently.

Deposit limit information and changes in deposit limits will be handled by an internal API, which the Frontend will connect using HTTP protocol. The frontend must have a retry functionality which must try to send the HTTP request to the other zone if the Internal API instance in the same zone is down of some reason. See Internal API chapter for more info.

10.6.2.1 Caching

My pages area will not have any response caching because to data shown is individual for each user.

10.7 Operator API

The Operator API will be developed as ASP.NET Core WebApi and it will be responsible of processing requests coming from the license holders. The API must validate that the license holder is authorized to send a request to the system by checking that:

³ <https://www.optimizely.com/>

⁴ <https://umbraco.com/>

- The license holder is active.
- The time of the request is between license holder's ValidFrom and InvalidFrom dates.
- The authorization key is correct.
- The request contains the actorId path parameter and X-Request-Id header parameter.

Only after validating that the incoming request is authenticated, the API server can process the request.

The API, in the test environment for license holders, expose a Swagger definition. The definition can be found in the delivery folder.

10.7.1 RequestDeposit endpoint

Requesting a deposit implies that the user has initiated a depositing process in an online casino. The endpoint will get the sum of daily, weekly, and monthly deposits of the user from the application database and compare the values with the user's respective limits minus the requested amount. The request will be allowed only if there's deposit limit left in all the limit types. If the deposit is approved, an authorization code will be created for the request, and it will be saved in the application database.

10.7.2 FinalizeDeposit endpoint

Finalizing a deposit implies that the licence holder has successfully processed the user's deposit request and the amount can now be registered in the application database. From the incoming authorization code, the application will find the relevant request in and add the deposit to the user's balance.

10.7.3 Internal endpoints (zPages)

The API server will expose so called internal endpoints to a specific port which won't be accessible from the internet, but only from the local area network. These endpoints will be used by the load balancer for health check and IP whitelisting purposes.

10.7.3.1 IP whitelisting

The endpoint GET /ipwhitelistz will get the X-Forwarded-For header and decide whether client IP is allowed. This endpoint will be used by the load balancer to decide whether the request can be let through or must be stopped.

10.7.3.2 Health check

The endpoint GET /healthz will decide whether the application can respond to the incoming requests or is in some bad state. The endpoint will return a positive response if and only if the following are true:

- Application has database access
- Application is not marked as offline by an external party such as a developer, a release pipeline etc.

The load balancer must call this endpoint every 10 seconds to determine the health status of the nodes. A node which is unhealthy for 30 seconds must be taken down by the load balancer from the instance pool until the endpoint returns an HTTP 200 status code.

10.7.4 Caching

The API server will cache license holder information such as their ID, authorization keys, enabled status & validity periods, and their IP addresses to increase the throughput of the system. This

information will be cached for a minute to ensure that the changes that may be made by an administrator will be applied in a relatively fast manner.

10.7.5 Logging

Logging should be implemented according to the needs of the system owner since it is not possible at this stage to know the minimum level of logging that is required considering everything from GDPR rules to legal necessities. The public facing API will be able to log the following if it is chosen to do so:

- **License holder information:** ActorId, IP address
- **Request information:** Timestamps for request & response, unique request & response Id, endpoint, request body, response body, HTTP status code, time taken to generate the response
- **Person information:** Hashed personal number (Personal numbers in clear text must not be logged)
- **Environment information:** Machine & environment name

10.7.5.1 Extra requirements

Logging huge volumes of data must be planned accordingly when it comes to application performance and the disk space needed for the data. The logs can be saved to the database by a separate service to not to degrade the API performance. The required disk space can be calculated by referring the SQL Server documentation⁵. Enabling page compression for the table may result in less space needed to save the data and is highly recommended.

10.7.5.2 Performance considerations

Saving a lot of data to the database must be done in a performant manner. In case of extensive logging, our recommendation is to save the log data to the local disk of the API servers and have another job which imports them to the database in bulk. This will decrease the number of open database connections.

Another consideration is to partition the log table into multiple files, where each partition represents a day. This will help in four ways:

1. Reading log data for report generation in Umbraco Backoffice will put a WHERE clause on the generated report query and only the relevant file groups will be read.
2. Writing log data will be quicker when the underlying file is smaller.
3. Backup logic of the database can be adjusted so that only the latest log file can be backed up. Other files won't need to be backed up again since logging is always append only and the data never changes.
4. Deleting the old logs due to data retention policies will be much quicker since instead of running regular DELETE statements which executes a delete operation on table rows by locking the table, a TRUNCATE TABLE statement can be run for the relevant partition to remove the file from the disk, in a much faster way.

10.8 Internal API

The internal API will be hosted on IIS along with the Frontend. This API will be the only way for the Frontend to communicate with the database and other functions of the system. Its existence has two main reasons:

⁵ Data types (Transact-SQL): <https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16>

- **Security:** Not exposing the whole system functionality to the internet mitigates risks. The Internal API will use a separate connection string for the application database. The connection string for the Frontend will only have access to Umbraco database.
- **Deployment independency:** Since the internal API will have the core business logic, it will be possible to deploy new versions of the API without affecting the Frontend.

The internal API will contain several methods which will be used on 'My pages' as well as administration view for Umbraco.

10.8.1 GetDepositLimits

This method will return the information presented on 'My pages' section. It will include the deposit limits, current deposit sums, and upcoming changes if such is available.

10.8.2 UpdateDepositLimits

This method will save the new limits of all periods.

10.8.3 Logging

Like the public facing API, the amount of logging should be considered carefully. The following data will be available for logging:

- **User information:** Hashed personal number, user IP address
- **User actions:** User actions such as setting limits, increasing limits, and decreasing limits in along with their timestamps; before and after values
- **E-identification method:** If the system supports more than one e-identification method, the type of the method

10.8.4 Administrator methods

The Internal API will have relevant endpoints to manage the system settings and to generate system reports.

10.8.4.1 Reporting

Each report type defined in the final requirements will have its own endpoint. These endpoints can be standardised to a degree by returning the same object no matter the report type. Such an example of return value can be a regular CSV string which then can be parsed on the client for presentation.

Report generation strategy must be decided on when the final requirements are set, and a performance test must be done to ensure that the selected strategy is optimal in terms of storage costs and time costs. If fast report generation is wished, a premade lookup table specific to the report can be maintained and updated when it is needed. This can increase the complexity of the program logic, but reports will be available all the time without any delay. If a report is not very time sensitive to have, it can be generated on the fly, although it will take longer time to process the data.

10.9 Load balancer

The load balancer will fill several important roles in the system:

- **Ensuring balanced loads in zones:** The load balancer will send the incoming requests to the zone with least active connections.
- **Protecting Umbraco Backoffice:** Umbraco Backoffice will have public facing link as default. There are different ways to protect it such as IP whitelisting, but it will be safer to disable the access to /backoffice link totally when accessing it from the internet. This decision impacts

how the Umbraco administrators will access to Backoffice; they must connect to a VPN to be able access Backoffice.

- **Performing health checks:** The application instances will expose an internal health check endpoints which the load balancer can use to determine whether a given application instance is health or not. If the instance stays unhealth for a pre-determined period, the load balancer will take the instance offline. This will mean that the unhealthy instance will be covered by the other zone while the instance is recovering.
- **Exposing the client IP address:** Since the system will only accept connections from the load balancer, the load balancer must send the original client IP on the header X-Forwarded-For. If the incoming request already includes that specific header, by whatever the reason is, the load balancer must overwrite it with the correct original client IP.
- **IP whitelisting:** The load balancer will allow incoming connections from known IP addresses and subnets. All other connections will be answered with HTTP 401 Unauthorized response code. This will ensure that the API server won't be busy filtering out unauthorized responses. The load balancer must be able to cache whether an IP address is whitelisted or not. Our experience shows that setting the caching duration to 15 minutes gives a good balance between performance and reflecting any changes in a reasonable time.

10.10 Business layer

The business layer will be separate assembly which will contain the entire application logic. Both the public facing API and the internal API will call the methods available in this assembly to perform their operations. The main responsibilities of this layer are but not limited to:

- To calculate the time an increase in limit can be applied and to schedule a job accordingly.
- To initiate a limit decrease update.
- To handle request from license holders and generate the response which will be sent back.

10.11 Data layer

The data layer will be responsible to read to and write from the database based on the requests from the business layer. It is advised that the data layer to use ADO.NET client to get the fastest performance possible. Since this is a small application, using ADO.NET will not incur too much extra development time compared to using an ORM such as Entity Framework.

10.12 Hangfire

Hangfire is a task scheduler that runs as a Windows Service. That means that the application will run when Windows is started and continue running until it is explicitly shut down either by the user, by an external script, or by Windows itself. This makes Hangfire a good candidate for handling time sensitive jobs, whether they are recurring jobs or one-time jobs. The system will use Hangfire both for recurring jobs and for one-time jobs.

10.12.1 Nightly job to delete old data

The deposit data which will be generated by license holders must be kept in the database for at least a month to calculate the remaining monthly deposit limit for any given user. Any data older than a month becomes thus subject to removal. A nightly job which will take care of this operation will be ran by Hangfire.

If the final requirements state that the API requests must be logged, this job will clean those logs as well.

10.12.2 One-time jobs to schedule limit increases

Limit increases will come into effect earliest after 72 hours but not before the current period has ended. This requires the system to create scheduled tasks for each increase a user initiates. It means a user who increases multiple period limits at the same time will trigger different tasks to be scheduled.

10.12.3 High availability

Processing time sensitive operations, Hangfire must be always running including times when releasing new versions as well as OS updates. Therefore, there will be two or more separate instances of Hangfire which runs in a clustered manner to ensure that jobs are ran when they are due.

11 Physical view

The physical view describes how the non-functional requirements are filled by setting up virtual servers in such way to ensure availability, performance, and scalability.

11.1 Software stack

11.1.1 Operative system

Windows Server 2022 (or newer if exists) will be used as operative system.

11.1.2 Web server

Spelgräns will be hosted as an IIS application as an in-process model since it results in improved performance compared to out-of-process according to Microsoft's documentation⁶.

The SSL termination will not be IIS' responsibility since it can be handled at a layer 7 load balancer.

11.1.3 Database

Microsoft SQL Server Standard Edition will be used as database. It has some limits compared to Enterprise Edition, but our experience shows that the Enterprise edition is rarely needed for projects of this size. If the load tests show that the database is a bottle neck for the performance, it can be considered to create a database cluster from Microsoft SQL Server Enterprise Edition and set the primary replica to ReadWrite and the secondary replica to Read to distribute the database load.

11.1.4 Load balancer

It is up to infrastructure provider to choose and configure the load balancer according to application's needs. The selected load balancer must be able to do the following:

- Support the modern TLS versions such as TLS 1.3 & 1.2 for increased security.
- Do health checks for an instance using a dedicated endpoint the instance exposes and take the instance offline should the health check fails.
- Optionally retry the failed HTTP requests automatically by sending the failed request to another application instance.
- Query an endpoint to determine whether the incoming request is allowed based on the client IP address.

Based on the requirements above, it is recommended that Citrix ADC (NetScaler)⁷ to be used as the load balancer of the system.

11.1.5 File share (NAS)

The servers need to have access to a file share to write and read common application data. The file share system must have a configurable backup system to prevent data loss.

One such product is Hitachi NAS Platform⁸.

11.1.6 SMTP server

The application will need to send emails to CMS Administrators in certain cases. An SMTP server will be required to send these outgoing emails. The SMTP server can either be inside the infrastructure or outside of it as a third-party service.

⁶ In-process hosting with IIS and ASP.NET Core: <https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/in-process-hosting?view=aspnetcore-7.0>

⁷ <https://www.citrix.com/products/citrix-adc/>

⁸ https://knowledge.hitachivantara.com/Documents/Storage/NAS_Platform

11.2 Hardware stack

11.2.1 CMS servers

We recommend that the CMS servers to have 4 vCPU & 8 GB RAM. Each server must have a minimum of 50 GB disk space for the system, and 50 GB disk space for the data. The disks don't need to be SSD.

11.2.2 API servers

We recommend that the API servers to have 6 vCPU & 8 GB RAM. Each server must have a minimum of 50 GB disk space for the system, and 50 GB disk space for the data. The disks don't need to be SSD.

11.2.3 Database servers

We recommend that the database servers to have 4 vCPU & 32 GB RAM. Each server must have a minimum of 60 GB disk space for the system and the SQL Server, and 3 TB disk space for the data. It is advised to have faster disk for the database servers, if available, since the system will be read and write heavy.

11.3 Security

To ensure the security of the virtual servers, they will be placed behind a firewall which can defend against cyber-attacks, including DDOS attacks.

To ensure the security of the systems and applications running on virtual servers, an antivirus software will be installed which can prevent harmful applications.

Access to servers via Windows Remote Desktop must always be done through a virtual private network (VPN) which must be provided by the infrastructure provider. Logging in to the VPN must require two factor authentication for increased security.

The infrastructure provider must have well established routines to update the servers' operative systems to get the latest patches from Microsoft.

All databases must be encrypted with Microsoft SQL Server Transparent Data Encryption.

11.4 Availability

Both are applications are subject to high availability constraints. To ensure this, the application will be hosted in two different and geographically redundant zones. The incoming request will be handled and redirected by the load balancer to the zone with least active connections to ensure a balanced load for both zones.

The firewall and the load balancer must also be redundant. That is, should it go offline for whatever the reason is a backup firewall or load balancer must take over the failed one's job. This failover process should not take longer than a couple of seconds.

The infrastructure provider must ensure that the production system will not be affected during operative system updates. Moreover, if an application has specific routines to ensure availability before operative system updates, the infrastructure provider must be able to perform the routines for each application during the patching process.

11.5 Performance

The virtual servers must be connected to each other by high-speed local area network. This can be satisfied by using gigabit ethernet within the local area network. The servers must also be connected to the internet by at least 1 Gbps (gigabit per second) speed.

11.6 Scalability

As it has been mentioned in 10.1 Design considerations chapter, the application will not be able to scale itself automatically depending on the load. The health level of the application must be watched continuously by automated systems and alarms must be configured to receive notifications should the system resources need to be adjusted and/or if new application servers need to be added.

11.7 Monitoring

The servers and applications will have internal & external monitoring which will continuously measure the following but not limited to:

- CPU, memory, and disk usage of a server.
- Health of the application by calling an application specific API-endpoint.
- Availability of the CMS and API by calling the public facing link of applications and ensuring that the response is correct.

11.7.1 Internal monitoring

One recommendation for monitoring the virtual servers is Monit⁹. Monit is a pull-based system which is installed on some third-party monitoring server within the infrastructure (often managed by the IT infrastructure provider) and it can be used for monitoring the server status as well as for the application health checks within the internal network.

Another option to use instead, or alongside, of Monit is a more detailed setup such as Prometheus together with Grafana. Prometheus is a real time statistics exporter and exposes server & application related statistics on a given endpoint. The exported statistics can include but not limited to:

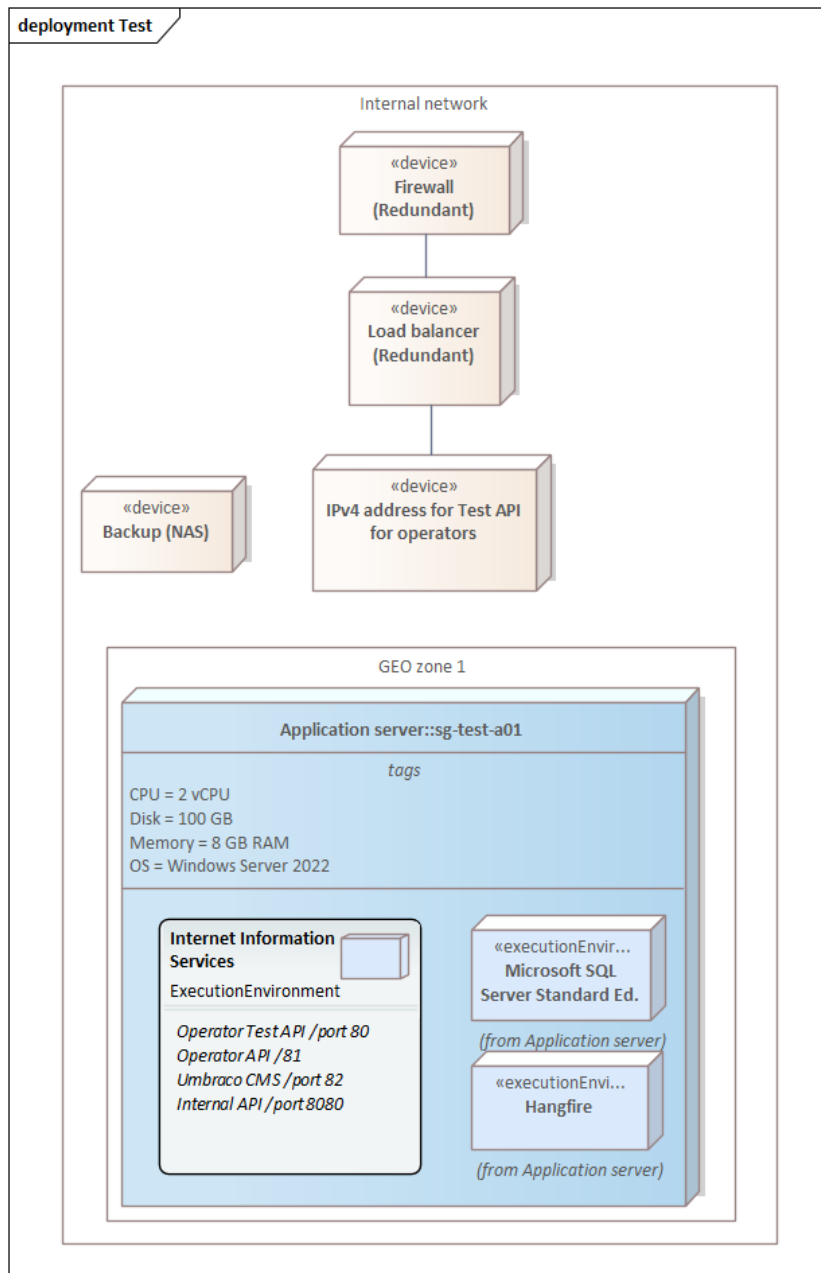
- CPU, RAM, & disk utilisation of the server
- Player statistics such as
 - o Player count
 - o Average deposit
- License holder statistics such as
 - o Current active requests per endpoint
 - o Average response time
 - o Most active license holders
- Application information such as
 - o Error count

The above list is just a sample of what can be achieved with a detailed monitoring tool. It is also possible to setup specific rules to create alerts if something is out of ordinary. Such an alert can be configured to be sent as an e-mail and be helpful to catch environment & application problems early on. Note that such a setup will require an extra virtual server to install Grafana and its dependencies on as it is not advised to install Grafana on application servers.

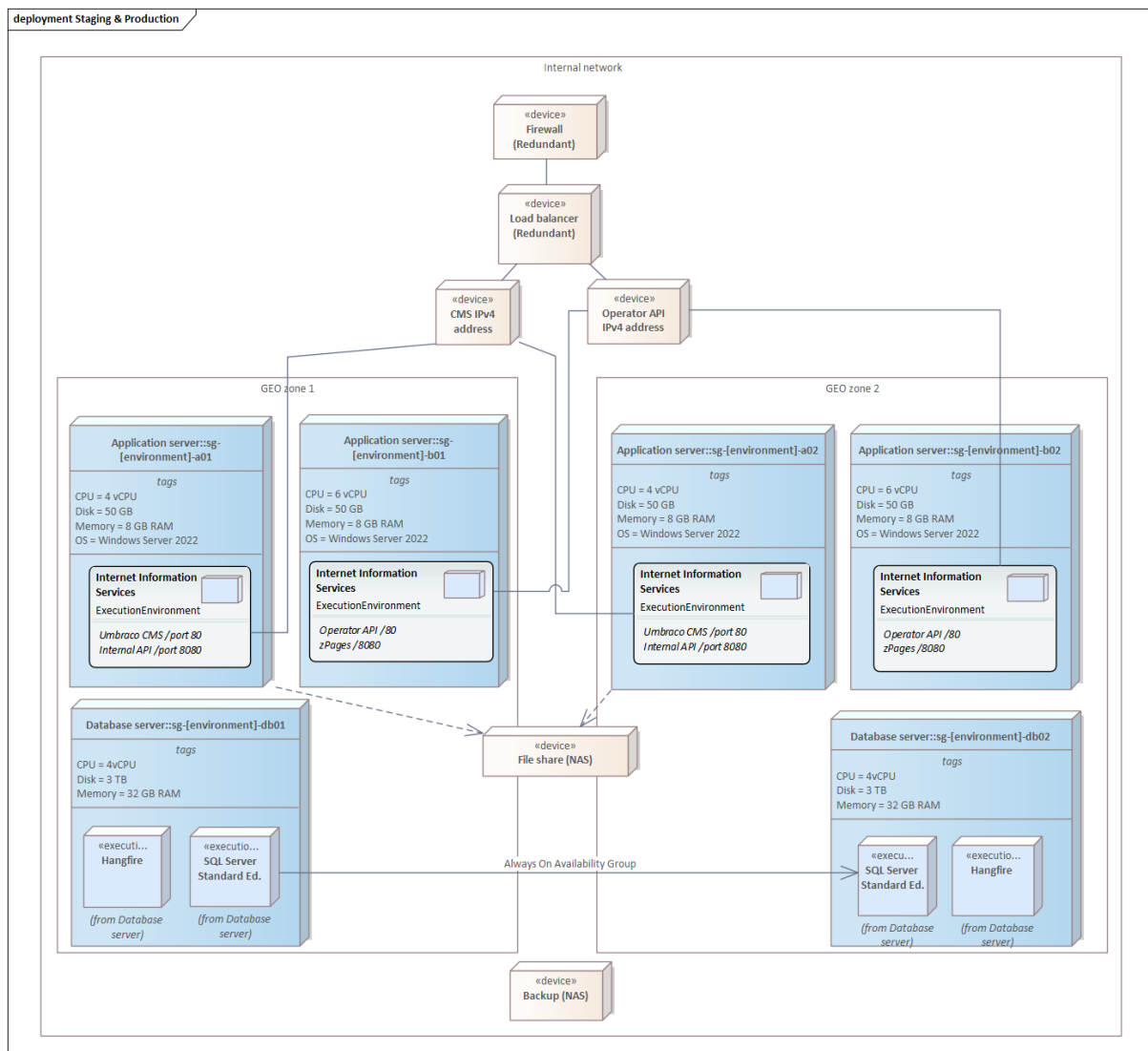
⁹ <https://mmonit.com/monit/>

11.8 Application environments

Below can be seen the deployment diagram of the test environment.



Below can be seen the application environments for the staging and production.



The virtual servers must not be exposed directly to the internet. Incoming HTTP requests must always be handled by the firewall and the load balancer. The virtual servers must be accessible by Windows Remote Desktop for users who have access rights, but only after connecting to a VPN which must be provided by the infrastructure provider.

11.8.1 Test environment

Test environment will be used to test deploying code packages automatically and to test the system in a virtual environment which is alike its production environment. Resources of the test environment can be much lower than staging & production environments. The test environment will also include a publicly available Operator API for license holders to test their implementations against. This so called 'test environment for license holders' must be behind the load balancer staging and production environments will have due have the exact same SSL termination settings. Logging to database, if such logging exists, and health checks for test environment can be turned off since they won't create any value to developers.

Systems in the test environment will not be accessible from the internet except the so called "Test environment for license holders".

11.8.2 Staging environment

The staging environment will be used to test any possible changes that are made to the system, including but not limited to code packages, hardware specifications, and other integrations the system may have. It will therefore mirror the software and hardware specifications of the production environment as well as the integrations the production system has, such as health checks etc. Databases from the production environment will not be mirrored actively, but it will be possible to move data between environments, albeit manually, if such need arises.

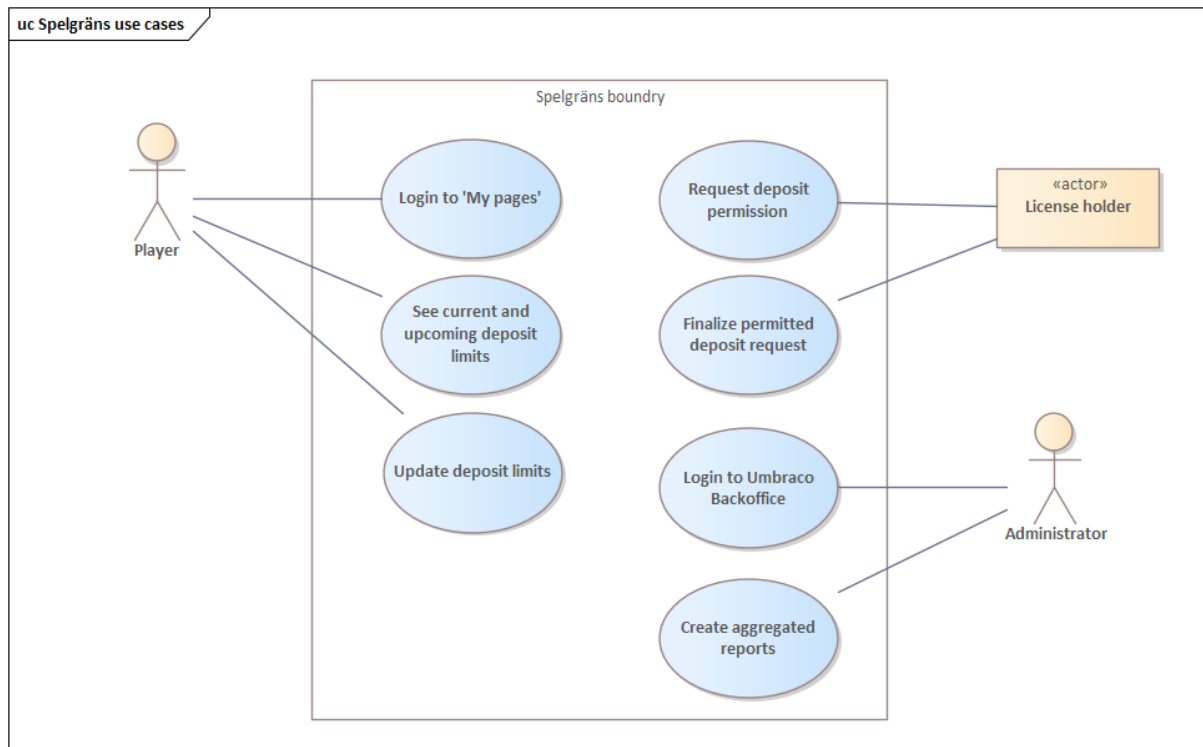
Systems in the staging environment will not be accessible from the internet.

11.8.3 Production environment

The production environment will be used to expose the functionality to the external world where license holders and players will have access from the internet. Administrators will not be able to access to Umbraco Backoffice from the internet due to security considerations. They must instead connect to the VPN provided by the infrastructure provider and use a separate link to access to Backoffice.

12 Scenarios / use-cases

This chapter describes use cases of different stakeholders that will interact with the system.



12.1 Conceptual level

The conceptual level focuses on the stakeholders which were identified above and how they will interact with the system.

12.1.1 Users

Two users and one actor have been identified for the use cases of Spelgräns. The users are players and administrators, whereas the actors are license holders.

12.1.2 Player

A player is a private person who wants to gamble in registered online casinos in Sweden or Finland who must configure her daily, weekly, and monthly national deposit limit. Player and user will be used interchangeably in this section.

12.1.2.1.1 Logging in

The user must log in to 'My pages' with a supported electronic identification method.

12.1.2.2 Setting up initial limits

After logging in to 'My pages' for the first time, the user will see a warning text stating that the deposit limits must be configured to deposit money to an online casino.

12.1.2.2.1 Viewing existing limits

Once the limits are configured by the user, it will be possible to see the existing limits on 'My pages' as well as how much deposit was made within the respective time periods.

12.1.2.2.2 Changing existing limits

The user can change its limits at any given time from 'My pages'. The limit adjustments are subject to special rules as stated in the section *Deposit limit rules*.

12.1.2.2.3 Viewing upcoming limit changes

Limit increases for any period will be applied earliest after 72 hours. During this waiting period, the user will see the date and time of the upcoming limit changes on 'My pages'.

12.1.3 Licensed operator

A licensed operator is a corporate body that has been granted a licence to offer gambling for money in Sweden or Finland.

12.1.3.1.1 Deposit request

The licensed operator will query whether a particular user has enough deposit space left for the given amount. Depending on the answer the API will return, the licensed operator will allow or deny the deposit.

12.1.3.1.2 Finalize deposit

The licensed operator, only after ensuring that the user has enough deposit space left, will be permitted to finalize a deposit. This action will reflect on the user's balance immediately.

12.1.4 Administrator

An administrator is a private person who is authorized by Spelinspektionen in Sweden to manage the content of the webpage, configure CMS settings, and create aggregated reports from the administrator view of Umbraco.

12.1.4.1.1 Logging in

The administrator must log in to Umbraco Backoffice, the management section of the Umbraco CMS, to perform administrative actions. Logging in to Umbraco will require a VPN connection to increase the security.

12.1.4.1.2 Changing the site content

The administrator can create, modify, and delete content from the public site freely. The administrator will have the option to preview the changes as they would have been seen on the public site before publishing the changes.

12.1.4.1.3 Changing the system settings

The administrator will have a dedicated user interface in Umbraco Backoffice to edit the Spelgräns system settings.

12.1.4.1.4 Managing the license holders

The administrator will be able to add, modify, and delete license holders from a dedicated user interface.

12.1.4.1.5 Creating aggregated reports

The administrator will have a dedicated user interface in Umbraco Backoffice to create predefined aggregated reports from the data Spelgräns has.

13 Appendices

13.1 Precio Fishbone

Precio Fishbone is a leading IT solutions company that specializes in Microsoft-based solutions. Their team of approximately 250 specialists has expertise in .NET platforms and frameworks, allowing them to develop customized software solutions that meet the specific needs of their clients. They have worked on several successful projects and have a deep understanding of virtually every type of organization, industry, and line of business. Precio Fishbone has Gold and Silver Partnerships in six different areas of technical competence with Microsoft and is a beta and launch partner in every new release of SharePoint since 2001.

One of the successful projects that Precio Fishbone has delivered is Spelpaus.se, the Swedish national self-exclusion system from online casinos, betting, bingo, and lotteries. The Spelpaus system has been a success, with more than 91,600 self-excluded individuals as of 5 April 2023. The system has received an average of 34,7 million queries every day to control if a player is on the national self-exclusion list, and on average, 32 players log in per second to licensed gambling sites. This project demonstrates Precio Fishbone's ability to develop solutions that address complex and sensitive issues, while also highlighting their technical capabilities in delivering successful projects to government institutions.

In addition to their technical capabilities, Precio Fishbone is also ISO 14001 and ISO 27001 certified, demonstrating their commitment to environmental management and information security, respectively. These certifications help to ensure that Precio Fishbone follows industry best practices and operates in a responsible, secure, and sustainable manner.

13.2 Spelpaus.se

Spelpaus.se is the Swedish national self-exclusion system from online casinos, betting, bingo and lotteries, token-gambling machines, and similar which have operation licenses to run money gambling businesses in Sweden. The players can, with an electronic identification, self-exclude themselves within seconds from all gambling companies who hold a Swedish license. A self-exclusion can last 1 months, 3 months, 6 months, or indefinitely (at least 12 months). For 1-, 3-, and 6-months periods the exclusion ends automatically once the period has come to an end. For indefinite exclusion, the players can choose to end the exclusion earliest after 12 months has passed since the exclusion date. The players have also the possibility to extend their ongoing exclusion with any of the time periods. An extension becomes active automatically when the current exclusion period ends. License holders must cross-check via the *license holder API* whether a player is self-excluding him or herself each time the player tries to log in and when a new player wants to register. License holders must not target the excluded players with direct advertisement either. (Spelpaus, 2023a)

According to the statistics Spelinspektionen publishes, on average of 32 players logs in per second to licensed gambling sites (Spelinspektionen, 2022a) and 34,7 millions queries every day to control if a player is on the national self-exclusion list (Spelinspektionen, 2022b). Within the first week of activating the site, about 4,000 people have self-excluded themselves from the system (Naess, 2019). Spelpaus.se has more than 91,600 self-excluded individuals as of 5 April 2023 (Spelpaus, 2023b).

14 References

- 4+1 architectural view model. (2023, March 30). Retrieved from Wikipedia:
https://en.wikipedia.org/w/index.php?title=4%2B1_architectural_view_model&oldid=1139933856
- Ericsson, V. (2022, December 22). *Spelinspektionen får i uppdrag att öka kunskapen om varför spelare väljer att stänga av sig från spel och undersöka om de fortsätter att spela hos spelbolag som saknar licens*. Retrieved from Regeringskansliet:
<https://www.regeringen.se/pressmeddelanden/2022/12/spelinspektionen-far-i-uppdrag-att-oka-kunskapen-om-varfor-spelare-valjer-att-stanga-av-sig-fran-spel/>
- GRANDID API. (2023, March 30). Retrieved from GrandID API Documentation:
<https://docs.grandid.com/>
- Naess, J. (2019, January 7). *4000 har registrerat sig för att slippa spelreklam – på under en vecka*. Retrieved from 4000 har registrerat sig för att slippa spelreklam – på under en vecka - Dagens Media: <https://www.dagensmedia.se/marknadsforing/marknadsforing/4000-har-registrerat-sig-for-att-slippa-spelreklam-pa-under-en-vecka/>
- Spelinspektionen. (2022, October 3-a). *32 inloggningar i sekunden*. Retrieved from 32 inloggningar i sekunden- Spelinspektionen: <https://www.spelinspektionen.se/om-oss/statistik/statistiknytt/32-inloggningar-i-sekunden/>
- Spelinspektionen. (2022, June 13-b). *34,7 miljoner kontroller per dag mot Spelpaus.se - Spelinspektionen*. Retrieved from 34,7 miljoner kontroller per dag mot Spelpaus.se: <https://www.spelinspektionen.se/om-oss/statistik/statistiknytt/347-miljoner-kontroller-per-dag-mot-spelpaus.se/>
- Spelpaus. (2023, April 5-a). *Stäng av dig från spel!* Retrieved from Välkommen till Spelpaus.se: <http://web.archive.org/web/20230405082050/https://www.spelpaus.se/>
- Spelpaus. (2023, April 5-b). *This is how it works*. Retrieved from Spelpaus: <https://www.spelpaus.se/en/how-it-works/>