# Комуникационен портал на приложението

За да се достъпи приложението от различните интерфейси то трябва всяка заявка да премине през комуникативния портал. Тапи методология е избрана поради множествено положителни показатели. Тези положителни показатели включват: единен вход което води до единни проверки за коректност, превод от rest метода на комуникиране по-бързия разширен протокол за опашка за съобщения( Advanced Message Queuing Protocol- AMQP)

# Защитен сервиз

## Автентикация

Автентикирането на потребители се случва на ниво защитен сервиз и включва потвърждаване на потребителя чрез емайл/потребителски име и парола. След което се създава JWT( Json Web Token) токен. За да се случи това потвърждаване на потребителскота самоличност първо трябва да се приемат неговите данни и това е възможно чрез следния бизнес обект:

public record UserLogIn(String credentials, String password) {}

Който съдържа и поле за идентификация и поле за парола. Което поле за идентификация представлява или емайл или потребителско име и поред вина на полета се приемат два различни начина на идентифициране. При приемате на данните те минават през верифициране на типа на данните, това се отнася за всяка заявка към сървъра. При

OAuth

Криптиране на пароли



Фиг. №: LXL- Изображение на как е реализиран модела за създаване на потребителски токен.

## База от данни отговорна за потребителски сесии

За запазване сесиите на активните потребители в момента се използва база от данни тип библиотека( dictionary) или още позната като ключ със стойност( key value pair). И конкретно тази база от данни тип библиотека е пряко и единствено сервиза със сервиза отговорен за защита(ink-security). Този сервиз както е разгледано в предишните подточки е отговорен за защитата на потребителските данни и общата безопасност на системата от нападения. Това не значи че другите сервизи не спазват традиционните практики за защита от атаки. Напротив те трябва да следват многослойни процедури и техники за избягване на масивни или централизирани атаки, като например DDOS( Denial-of-service attack). Но самите сервизи не трябва да отговарят дали дадената потребителска сесия е активна, това е работа на сервиза за защита който автентификация и авторизация потребителя да има достъп до дадените сервизи. Затова е нужна база която да съхранява времената информация за потребителя, неговата авторизация и до колко е валидна сесията.

Относно базата използвана за реализация на каширане на сесиите на потребителите е използвана дистанционно разгъната библиотечна база от данни Redis. Трябва да се уточни че се използва библиотечната функция на Redis, тъй като Redis поддържа и множествени други специализирани режими на съхраняване на данни. Други функционалности на Redis включват и: база от данни подходяща за търсачи (Search data base), съхраняване в стандарт JSON, граф бази от данни, таблична база от данни, клъстър база от данни и много други които могат да се комбинират и споделят информация помежду си. Но за целта на реализация на каширане на потребителски сесии сме използвали библиотечните функции на платформата. Redis е лидер относно този вид съхраняване на данни и е използван от най-големите софтуерни фирми да забързат свалянето на често използвани данни от традиционни релационни бази от данни. Като пример е социалната мрежа Twitter които използват Redis за да забързан предаване на най-популярните публикации на платформата.



Фиг. XX: Графика показваща цялостна бързина спрямо релационни и не релационни бази от данни.

В java се поддържа свързаност с базата от данни чрез библиотеката наречена ‚ jedis‘ и е създадена от същите създатели на базата от данни. Задава се нова инфомация към базата като след като първо се реализира връзка към базата от данни чрез:

*JedisPool pool = new JedisPool("localhost", 6379);*

След усъществена връзка се преминава към вземане, добавяне, променяне или изтриване на хеш от базата. Пример за вписване на данни в базата:

*jedis.hset("user-session:123", hash);*

# Източници

Работа с база от данни Redis и java: https://redis.io/docs/connect/clients/java/