Examensarbete, Högskolan på Åland, Utbildningsprogrammet för Informationsteknik

API FÖR AUTENTISERING INOM 3D-SECURE

Jonathan Terry, Emanuel Sanchez

2024:12

Datum för godkännande: 24.05.2024
Handledare: Joakim Isaksson

EXAMENSARBETE
Högskolan på Åland

Utbildningsprogram:	Informationsteknik
Författare:	Jonathan Terry, Emanuel Sanchez
Arbetets namn:	API för autentisering inom 3D-Secure
Handledare:	Joakim Isaksson
Uppdragsgivare:	Extern uppdragsgivare

Abstrakt

Syftet med detta examensarbete är att utveckla ett API som är en del av en 3D-Secure-lösning och möjliggör för en extern part att begära stark autentisering av användare som har initierat en kortbetalning. API:et kommer att erbjuda olika autentiseringsmetoder för att säkerställa och verifiera användarens identitet. Projektet är en fristående applikation som använder de senaste versionerna av aktuella ramverk för att garantera högsta möjliga säkerhetsnivå och långsiktig hållbarhet.

Applikationen är utvecklad i Java och använder ramverket Spring Boot. För att definiera API:et använder vi oss av OpenAPI som vi sedan genererar gränssnitt ifrån.

Nyckelord (sökord)

Spring, OOB, Autentisering, OpenAPI, 3DS

Högskolans serienummer:	ISSN:	Språk:	Sidantal:
2024:12	1458-1531	Svenska	38

Inlämningsdatum:	Presentationsdatum:	Datum för godkännande:
10.05.2024	24.05.2024	24.05.2024

DEGREE THESIS
Åland University of Applied Sciences

Degree program:	Information Technology
Author:	Jonathan Terry, Emanuel Sanchez
Title:	API for authentication within 3D-Secure
Academic Supervisor:	Joakim Isaksson
Commissioned by:	External employer

Abstrakt

The purpose of this thesis is to develop an API that is part of a 3D-Secure solution and enables an external party to request strong authentication of users who have initiated a card payment. The API will offer various authentication methods to ensure and verify the user's identity. The project is a standalone application that uses the latest versions of current frameworks to guarantee the highest possible level of security and long-term sustainability.

The application is developed in Java and uses the Spring Boot framework. To define the API, we use OpenAPI from which we then generate interfaces.

Keywords
Spring, OOB, Authentication, OpenAPI, 3DS

Serial number:	**ISSN:**	**Language:**	**Number of pages:**
2024:12	1458-1531	Swedish	38

Handed in:	**Date of presentation:**	**Approved:**
10.05.2024	24.05.2024	24.05.2024

INNEHÅLLSFÖRTECKNING

1. INTRODUKTION

1.1 Syfte

Syftet med detta examensarbete är att skapa ett API som är en tjänst som gör det möjligt för olika system att kommunicera med varandra för att säkerställa och verifiera identiteten på användare som har initierat en kortbetalning. Applikationen kommer att delas upp i två servrar, en intern som sköter själva autentiseringen, och en extern som hanterar kryptering och signering av JWT (*JSON Web Token*) för att säkra data som överförs samt kommunikation utåt. I detta examensarbete kommer vi att gå igenom utvecklingsprocessen och de verktyg som använts.

När arbetet är färdigställt kommer vi ha en fristående applikation som en tredje part kommer att integrera emot för att autentisera användare som vill utföra en kortbetalning online.

Denna applikation är en del av ett större projekt som innefattar en hel 3D-Secure-lösning för en finansiell aktör som involverar flera olika parter men i detta arbete kommer vi fokusera på autentiseringen och kort förklara innebörden av 3D-Secure samt dess protokoll.

1.2 Metod

Vi kommer att arbeta agilt enligt utvecklingsmetodiken SCRUM och dela upp arbetet i små delar, även kallade “stories”, som i korta segment specificerar vilket arbete som behöver göras. Initialt har vi haft uppstartsmöte med vår handledare samt andra utvecklare där vi kort har gått igenom varje “story”. Vi kommer att bygga upp koden enligt en standard som definieras av beställaren och eftersom vi redan har jobbat en tid som trainees för beställaren är vi bekanta med denna standardisering.

Vartefter vi slutför segment av applikationen kommer vi att lägga upp koden internt för att granskas av andra utvecklare som får möjlighet att ge feedback. Feedbacken kan vi använda för att förbättra applikationen och rätta till eventuella misstag.

1.3 Avgränsningar

Eftersom denna applikation kommer att hantera en stor mängd känslig information kommer vi inte att beskriva utvecklingsmiljön eller eventuella testverktyg som kommer att skapas. Den kod som visas kommer att vara kodexempel eller modifierade versioner av den verkliga koden.

2. RAMVERK OCH VERKTYG

2.1 OpenAPI specification (OAS)

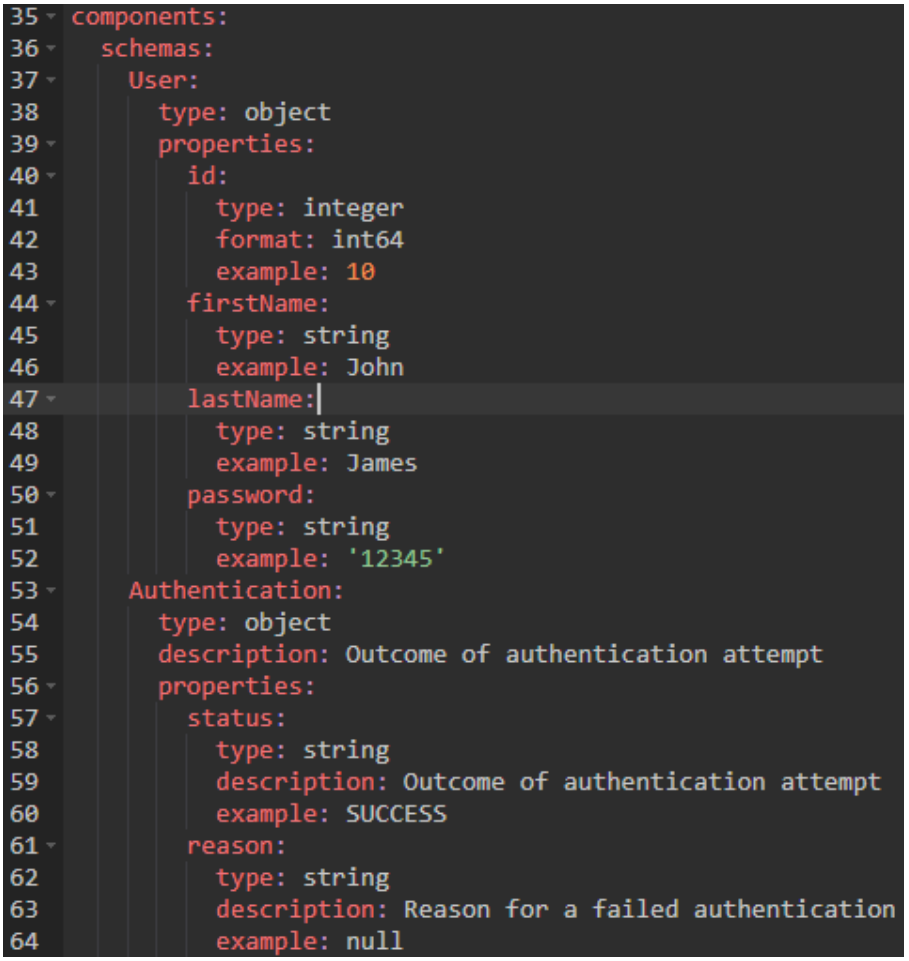
OAS är en standard där ett API definieras i antingen JSON- eller YAML-format. Fördel är att konsumenten och producenten av API:et inte behöver förstå eller ha kunskap om den bakomliggande koden. API:et är oberoende av programmeringsspråk. Eftersom API-specifikationen följer en öppen standard ger den en överblick som underlättar samarbete bland olika utvecklare oavsett deras tekniska bakgrund. (*What Is OpenAPI?*, 2023). Figur 1-3 är ett exempel på en API-specifikation enligt OAS, skriven i YAML.

```yaml
openapi: 3.0.3
info:
  title: Thesis API
  description: 'Example OAS specification for an API.'
  contact:
    email: apiteam@owner.com
  version: 1.0.0
servers:
  - url: http://localhost:8080
tags:
  - name: user
    description: Operations about user
```

Figur 1. Information om OpenAPI-specifikationen.

```yaml
paths:
  /authenticate:
    post:
      tags:
        - user
      summary: Authenticate a user
      operationId: authUser
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/User'
        required: true
      responses:
        '200':
          description: Successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Authentication'
        '400':
          description: Parameter validation failed, bad request
```

Figur 2. Specifikation av anslutningspunkter.



Figur 3. Specifikation av komponenter.

2.1.1 Kodgenerering utgående från en OpenAPI-specifikation

Vår applikation använder sig av OpenAPI Generator vilket är ett plugin till Java och andra programmeringsspråk för att generera kod. Koden som genereras består av olika objekt och klasser som definieras i OAS-specifikationen. Koden som genereras kan ses som ett skelett som vi sedan använder och implementerar i applikationen. Detta utvecklingssätt minimerar behovet av att skapa kod som är onödig och tidskrävande. På samma sätt kan klienter genereras utifrån ett befintligt kontrakt. Koden genereras under byggprocessen av applikationen, så om kontraktet ändras så ändras också koden.

Figur 4 visar komponenten "User" som är genererad utgående från komponentspecfikiationen som visas i figur 3. (*What Is OpenAPI?*, 2023)

```java
@Generated(value = "org.openapitools.codegen.languages.SpringCodegen",
        date = "2024-04-21T22:17:44.116011600+03:00[Europe/Helsinki]")
public class User implements Serializable {
    no usages
    private static final long serialVersionUID = 1L;
    7 usages
    private Long id;
    7 usages
    private String firstName;
    7 usages
    private String lastName;
    7 usages
    private String password;
```

Figur 4. Java-kod för komponenten User.

2.2 Spring

Spring är ett ramverk baserat på Java som erbjuder en heltäckande programmerings- och konfigurationsmodell för Java-baserade applikationer och är utformat för att stödja utvecklare i att skapa högpresterande, lätt testad och återanvändbar kod. Ramverket Spring inkluderar ett brett urval av moduler för uppgifter såsom databasåtkomst, transaktionshantering och webbtjänster. Dessa moduler är utformade för att fungera sömlöst tillsammans och kan kombineras på olika sätt efter specifika behov. (*Spring Framework - Overview*, n.d.)

2.2.1 Spring Boot

Spring Boot är en utvidgning av Spring som förenklar användning av ramverket genom att automatisera repetitiv kod och konfigurationer som vanligtvis krävs. Spring Boot erbjuder funktioner som självkonfigurering, inbyggda webbservrar, samt förkonfigurerade mallar för vanliga användningsområden för att förenkla utvecklingen och implementeringen av applikationer. (*What Is Java Spring Boot?*, n.d.)

2.3 Gradle

Gradle är ett byggverktyg som används för att automatisera och effektivisera processen att bygga, testa, och distribuera applikationer. Det använder ett domänspecifikt språk (DSL) baserat på Groovy eller Kotlin, vilket gör det mycket anpassningsbart och kraftfullt för komplexa byggscenarier. Dessutom använder Gradle en effektiv cache-mekanism som gör att upprepade byggprocesser går snabbare genom att bara göra om de delar som faktiskt har ändrats sedan senast. (Markovic, 2023)

2.4 AWS

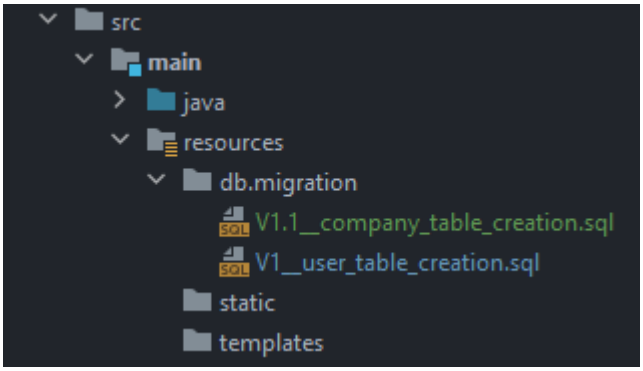
Amazon Web Services (AWS) är en molnplattform som erbjuds av Amazon och tillhandahåller över 200 tjänster från globala datacenter. AWS erbjuder en bred uppsättning infrastrukturtjänster, från grundläggande tjänster som beräkningskraft, lagringsalternativ och nätverk till avancerade tjänster som maskininlärning, artificiell intelligens och databashantering.

En av de största fördelarna med AWS är dess prisstruktur, som tillåter användare att endast betala för de resurser de använder. Detta gör det möjligt för företag av alla storlekar att experimentera och växa utan betydande initiala investeringar i hårdvara eller infrastruktur. (Kumar, R., 2020)

2.5 Flyway

Flyway är ett verktyg baserat på öppen källkod designat för att hantera databasmigreringar, vilket underlättar versionshantering av databasscheman och gör det möjligt att enkelt spåra och samarbeta kring databasändringar. Flyway gör detta genom att använda en serie av SQL-skript som beskriver förändringarna som ska göras i databasen. SQL-skripten körs sedan automatiskt i en specificerad ordning för att säkerställa att databasens schema är konsekvent och uppdaterat i alla miljöer. Egenskapen “fail fast” innebär att Flyway snabbt identifierar problem med migreringsskripten och stoppar processen innan några skadliga ändringar tillämpas på databasen. Detta bidrar till högre säkerhet och stabilitet i databasoperationer, vilket är avgörande för företagskritiska applikationer. (Factor, 2021)

Figur 5 illustrerar två SQL-skript som leder till skapande av två tabeller i databasschemat benämnda enligt Flyways konventioner, vilka är avgörande för att säkerställa korrekt versionshantering av databasschemat. Den strikta namngivningen är kritisk för att Flyway ska kunna identifiera och tillämpa migrationer i en sekventiell och organiserad ordning. Figur 6 visar loggen efter en lyckad migration. Loggen bekräftar hur många migrationer som har exekverats och databasschemats versionsnummer. (Fritchey, 2020)



Figur 5. Två SQL-migrationsskript.

```
: Creating Schema History table "TEST"."flyway_schema_history" ...
: Current version of schema "TEST": << Empty Schema >>
: Migrating schema "TEST" to version "1 - user table creation"
: Migrating schema "TEST" to version "1.1 - company table creation"
: Successfully applied 2 migrations to schema "TEST", now at version v1.1 (execution time 00:00.015s)
```

Figur 6. Loggutskrift från en migration.

2.6 Testcontainers

Testcontainers är ett bibliotek som används för att skapa lättanvända temporära miljöer för integrationstestning med containrar. Det låter utvecklare starta databaser, webbservrar, eller vilken annan mjukvara som kan köras i en Docker-container, direkt i testkoden. Detta ger stora fördelar eftersom det garanterar att testmiljön är ren och konsekvent varje gång testerna körs, vilket eliminerar problemen med att konfigurera och underhålla permanenta testmiljöer (*Testcontainers*, n.d.). I och med Spring Boot 3.1.0 förbättrade Spring sin support för Testcontainers och introducerade möjligheten att använda ramverket även i utvecklingsprocessen för att enkelt starta en databas i en Docker-container (Halbritter, 2023).

2.6.1 Isolering av lokala miljöer

Figur 7 och 8 visar praktisk användning av Testcontainers för att möjliggöra både lokal utveckling och integrationstestning. Figur 7 ger ett exempel som demonstrerar initialiseringen av en Oracle Testcontainer, vilket illustrerar hur enkelt en databas kan konfigureras och startas för utvecklingsändamål. Applikationen startas genom en referens som är placerad i projektets testpaket som ses i figur 8 och inkluderar testkonfigurationen för att undvika att containern inkluderas i den slutliga jar-filen, vilket är kritiskt för att separera de lokala miljöerna från de distribuerade miljöerna.

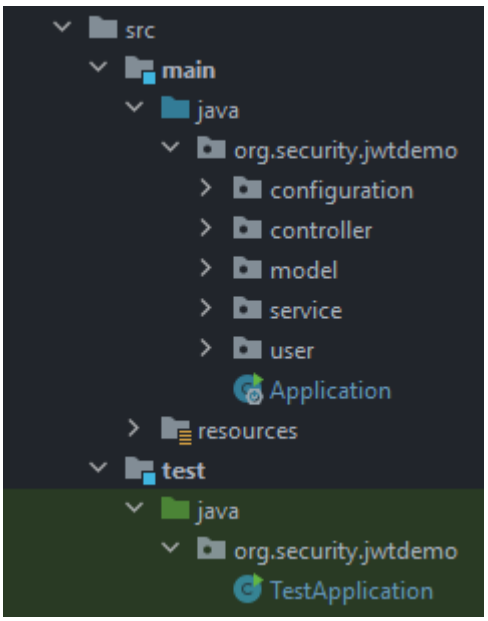
```java
@TestConfiguration(proxyBeanMethods = false)
public class TestApplication {

    @Bean
    @ServiceConnection
    OracleContainer getOracleContainer() {
        List<String> portBindings = new ArrayList<>();
        portBindings.add("56476:1521");
        OracleContainer container =
                new OracleContainer( dockerImageName: "gvenzl/oracle-xe:21-slim-faststart")
                .withDatabaseName("TEST_DB")
                .withReuse( reusable: true);
        container.setPortBindings(portBindings);

        return container;
    }

    public static void main(String[] args) {
        SpringApplication.from(Application::main).with(TestApplication.class).run(args);
    }
}
```

Figur 7. Konfigurationen av containern.



Figur 8. Ett projekts paketstruktur.

2.7 JSON Object Signing and Encryption(JOSE)

JSON Object Signing and Encryption (JOSE) är en samling av standarder för att säkra JSON-data genom signering och kryptering. JOSE inkluderar flera specifikationer:

- **JSON Web Signature (JWS):** Används för att signera JSON-objekt digitalt. Detta ger äkthetsbevis och säkerställer att innehållet inte har ändrats.
- **JSON Web Encryption (JWE):** Ger mekanismer för att kryptera JSON-objekt, vilket skyddar data från obehörig åtkomst.
- **JSON Web Key (JWK):** Definierar formatet för nycklar som används för signering och kryptering, vilket möjliggör hantering och distribution av kryptografiska nycklar.
- **JSON Web Algorithms (JWA):** Specificerar vilka kryptografiska algoritmer som är kompatibla med JWS och JWE.
- **JSON Web Token(JWT):** Ett kompakt och standardiserat sätt att säkert överföra information mellan parter med hjälp av JSON-datastrukturer.

Dessa standarder arbetar tillsammans för att möjliggöra säkert utbyte av data mellan olika parter genom att använda JSON-formatet som är enkelt att läsa, skriva och dela mellan system. (“JOSE - JSON Object Signing and Encryption,” 2021)

2.7.1 Asymmetrisk signering

Signering av JWT med en asymmetrisk algoritm är en process för att säkerställa integriteten och äktheten av datan i en token. Processen går ut på att data i en JWT signeras med den privata nyckeln. Detta innebär att man använder den privata nyckeln tillsammans med en signaturalgoritm (som RSA eller ECDSA) för att generera en digital signatur baserad på innehållet i en token.

När en token tas emot, kan den som mottar den använda den publika nyckeln från producenten av den signerade JWT-token för att verifiera signaturen. Om signaturen är korrekt bekräftar det att datan inte har manipulerats sedan den signerades och att den som skapade signaturen besitter den privata nyckeln.

2.7.2 Asymmetrisk kryptering

Asymmetrisk kryptering används för att säkerställa att endast innehavaren av den privata nyckeln kan läsa innehållet i en token. På samma sätt som asymmetrisk signering används ett nyckelpar i form av en publik och privat nyckel men själva krypteringen sker med mottagarens publika nyckel vilket säkerställer att endast mottagaren kan läsa innehållet även om någon obehörig har tillgång till denna token.

2.8 JSON Web Token (JWT)

JSON Web Tokens (JWT) är en öppen standard (Jones et al., 2015) för säkert utbyte av information mellan parter, vanligtvis mellan en server och en klient. Information i en JWT kodas och valfritt krypteras i form av JSON-objekt och den resulterande strängen används för säker överföring eller autentisering. JWT kodas i form av Base64URL, vilket innebär att de kan användas i en webbläsares adressfält. En sådan JWT visas i figur 9.

Figur 9. En Base64URL kodad JWT.

Figur 10 visar hur en JWT är uppdelad i tre delar:

- ***Header***: Innehåller metadata om token, såsom vilken typ av token det är (t.ex. JWT, Access token, Bearer token, eller ID token) och vilken algoritm (t.ex. HMAC, ECDH-ES, ECDSA eller RSA) som används för att signera eller kryptera token.
- ***Payload***: Innehåller den faktiska datan. Detta kan inkludera användaruppgifter, giltighetstider för token, utfärdare och andra attribut. Viktigt är att datan i payload är läsbar för vem som helst om den inte krypteras.
- ***Signature***: För att säkerställa att en token inte har manipulerats skapas en signatur med hjälp av headern, payloaden och en hemlig nyckel eller ett offentligt/privat nyckelpar. Denna signatur hjälper mottagaren att verifiera att token kommer från den avsändare som förväntas och inte har förändrats efter att den skapades.

Figur 10. Strukturen för en JWT i JSON-format.

2.9 JSON Web Keys (JWK)

En JWK (JSON Web Key) är en JSON-representation av en kryptografisk nyckel, som definieras i RFC 7517 (Jones, 2015). Den används främst i samband med JSON Web Tokens (JWT) för att signera eller kryptera tokens. JWK-formatet gör det möjligt att på ett enkelt och standardiserat sätt utbyta kryptografiska nycklar mellan olika system över webben. Figur 11 visar två uppsättningar av JWK-nycklar i ett JSON Web Keys Set (JWKS) som används för signering och kryptering.

Figur 11. JSON Web Key Set.

Tabell 1 representerar komponenterna i en JWK som används för att beskriva en kryptografisk nyckel baserad på en elliptisk kurva. Varje parameter i en JWK ger specifik information som tillsammans definierar hur nyckeln ska användas, vilken algoritm den är associerad med, och vilken typ av kryptografiska operationer den stödjer. Det är värt att notera att strukturen på en JWK kan variera beroende på nyckeltyp; till exempel, en RSA-nyckel skulle inkludera andra parametrar som n (modulus) och e (exponent), vilket inte är relevant för en EC-nyckel.

Tabell 1. Strukturen för en JWK som är av typ EC.

Parameter	Beskrivning
kty	Nyckeltypen; här anger "EC" att det är en kryptonyckel baserad på en elliptisk kurva.
d	Den privata nyckeldelen, som är en del av den privata nyckelinformationen för en elliptisk kurva.
use	Nyckelns användningsområde; “sig” för signature och “enc” för encryption.
crv	Namnet på den elliptiska kurvan som används.
kid	Nyckelidentifierare; ett värde som används för att matcha en specifik nyckel.
x	X-koordinaten för nyckelpunkten på den elliptiska kurvan; en del av den publika delen av nyckeln.
y	Y-koordinaten för nyckelpunkten på den elliptiska kurvan; en del av den publika delen av nyckeln.
alg	Algoritmen som ska användas med nyckeln.

3. TEORETISK BAKGRUND

3.1 Autentisering

Autentisering är ett brett område som inrymmer mer än enbart inloggning. Det gäller exempelvis också signeringar eller andra moment där identiteten av en entitet behöver styrkas.

Inom bankvärlden och EU är autentisering hårt reglerat. En autentisering av en kortbetalning måste följa “Payment Service Directive” (PSD2) och dess bestämmelser för “Strong Customer Authentication" (SCA). Om kraven i bestämmelserna inte uppfylls måste företaget som erbjuder banktjänsten betala höga böter. “3 Domain Secure” (3DS) är en företagsstandard när det gäller kortlösa kortbetalningar inom bankvärlden. 3DS 2.0 följer EU:s krav på SCA (*3-D Secure*, 2024). Genom att använda protokollet säkerställs även att kraven efterföljs.

En viktig parameter i både PSD2 och 3DS är kravet på multifaktorautentisering av kortbetalningar och hur dessa får genomföras. Genom att ställa höga krav på multifaktorautentisering försöker EU och alla inblandade parter minska på bedrägerier och hackerattacker vid betalningar som sker över internet.

3.1.1 Multifaktorautentisering (MFA)

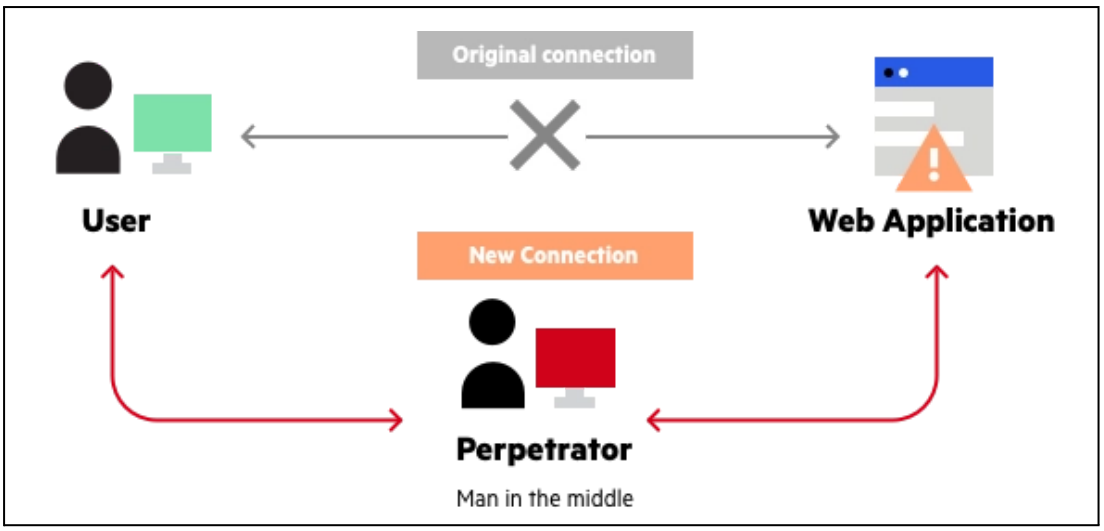
Multifaktorautentisering är en autentisering där två eller flera autentiseringar baserat på olika faktorer ingår. Det finns flera olika former för enskilda autentiseringar, och dessa kan delas upp i fyra olika kategorier. Kunskapsbaserad autentisering är någonting som användaren vet, exempelvis ett lösenord. Tillgångsbaserad autentisering är till exempel en applikation för inloggning eller en hårdvarunyckel. Fingeravtryck och röstigenkänning är exempel på biometrisk autentisering. Kontextbaserad autentisering handlar om plats eller sammanhang, exempelvis ett slutet nätverk (*MFA vs. 2FA: What’s the Difference?*, 2022).

3.1.2 “One Time Password” (OTP)

Namnet ger en antydan om innebörden av OTP. Detta är ett genererat lösenord som är giltigt endast en gång. Syftet med lösenordet är att minska olika attackytor. Den uppenbara fördelen är att koden enbart är brukbar vid ett tillfälle. Varje signering eller ny inloggning kräver en ny kod. Även om koden hamnar i fel händer är koden förbrukad efter en användning. Det här minimerar drastiskt attackmöjligheterna för en potentiell hackare och minskar konsekvenserna av en förlorad kod. OTP är fortfarande utsatt för olika typer av attacker men minimerar risken. Därför används ofta OTP i kombination med andra metoder som pushnotiser för att exempelvis meddela användaren att någon försöker genomföra en betalning. En annan vanlig metod för att öka säkerheten är att OTP skickas “out-of-band” till användaren (*One-Time Password*, 2024).

3.1.3 “Out-Of-Band authentication” (OOB)

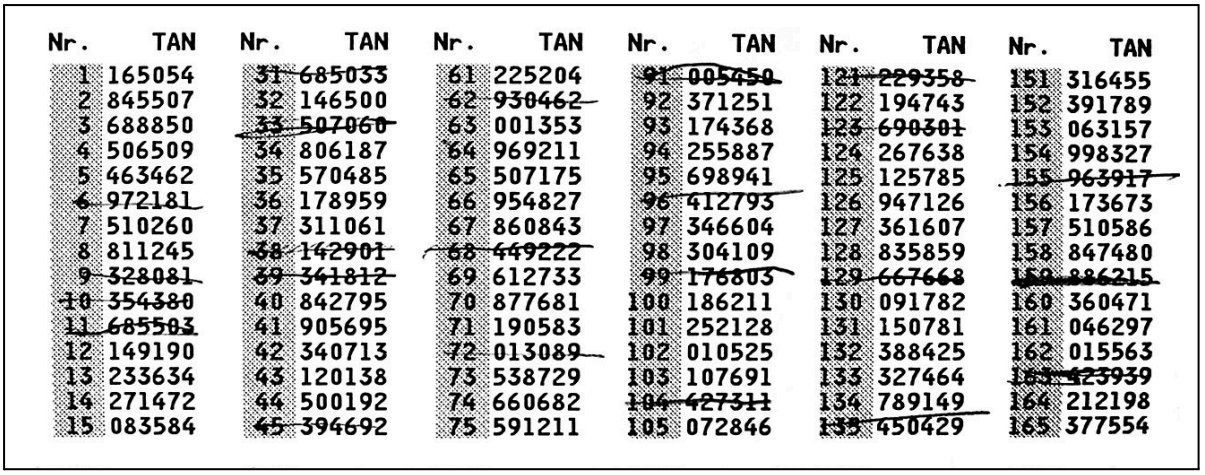
“Out-Of-Band Authentication” är en form av multifaktorautentisering. Viktigt är att det i en OOB-autentisering sker två verifikationer över två olika kommunikationskanaler. Det här är vanligt förekommande inom system som kräver hög verifiering av användarna. Ett enkelt exempel är att användaren loggar in via datorn. En notis skickas till användarens telefon och användaren loggar in i en applikation för att verifiera inloggningen. Syftet med att använda sig av OOB-autentisering är att förhindra eller begränsa eventuella “man-in-the-middle-attacker” (MITM). En MITM är när en hackare får tag i information mellan sändaren och mottagaren och använder den för att få åtkomst till ett system utan att användaren vet om att det pågår (Identity Management Institute, 2022). Figur 12 visar en illustration av en “man-in-the-middle-attack”.



Figur 12. Illustration av en "man-in-the-middle-attack" (Masas et al., 2024)

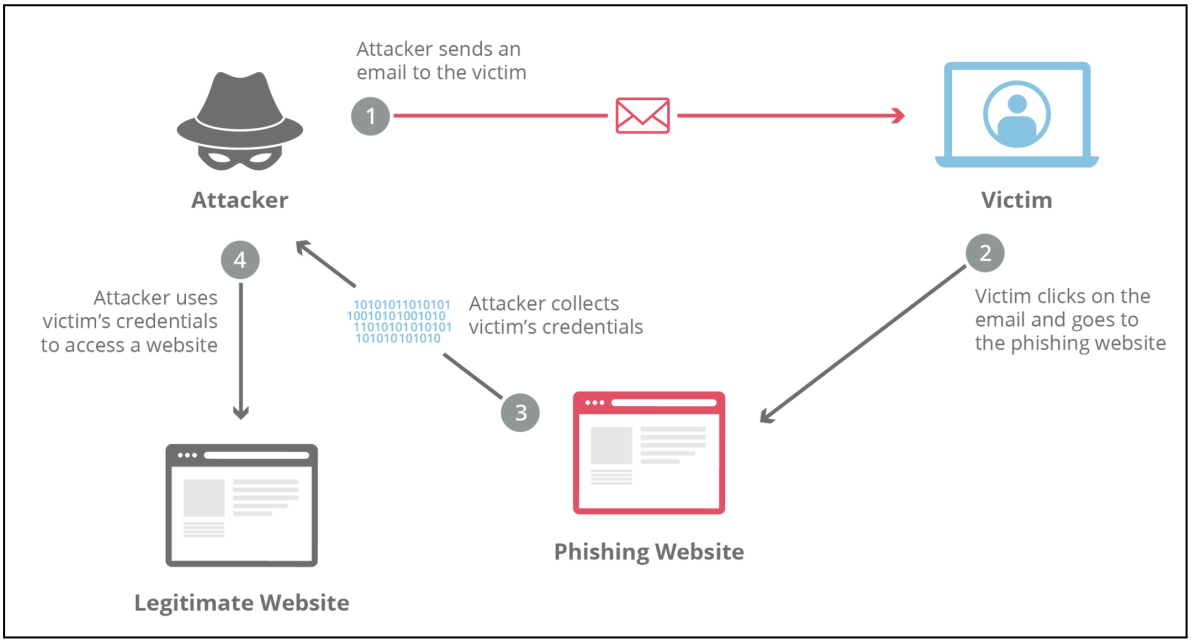
3.1.4 "Personal Identification Number" / "Transaction Authentication Number"

"Personal Identification Number" (PIN) och "Transaction Authentication Number" (TAN)-koder används ofta tillsammans. PIN är en personlig kod som fås separat från TAN-koderna. TAN-koderna fås vanligen utskickade via vanlig post eller alternativt hämtas de ut från banken. Figur 13 är ett exempel på hur en TAN-kodtabell ser ut i verkligheten. PIN/TAN-systemet är en multifaktorautentisering som består av två typer av autentiseringformer. PIN-delen är kunskapsbaserad och TAN-delen är baserad på ägande. TAN-koder används enbart en gång och är inte brukbara vid flera tillfällen. Systemet är sårbart för så kallat nätfiske ("phishing attacks") och "man in the middle"-attacker. I figur 14 finns ett diagram som visar en nätfiskeattack En version av TAN är indexerade TAN-koder (iTAN), vilket minskar riskerna något. iTAN betyder i praktiken att det alltid är en specifik kod från tabellen som efterfrågas. Vid varje autentiseringsförsök specificeras vilken TAN-kod som bör användas. Indexerade TAN-koder skyddar inte helt från dessa attacker men det är svårare att utföra en sådan attack om iTAN används. (*Transaction Authentication Number*, 2023).



Nr.	TAN	Nr.	TAN	Nr.	TAN	Nr.	TAN	Nr.	TAN	Nr.	TAN
1	165054	31	685033	61	225204	91	005450	121	229358	151	316455
2	845507	32	146500	62	930462	92	371251	122	194743	152	391789
3	688850	33	507060	63	001353	93	174368	123	690301	153	063157
4	506509	34	806187	64	969211	94	255887	124	267638	154	998327
5	463462	35	570485	65	507175	95	698941	125	125785	155	963917
6	972181	36	178959	66	954827	96	412793	126	947126	156	173673
7	510260	37	311061	67	860843	97	346604	127	361607	157	510586
8	811245	38	142901	68	449222	98	304109	128	835859	158	847480
9	328081	39	341812	69	612733	99	176803	129	667668	159	886215
10	354380	40	842795	70	877681	100	186211	130	091782	160	360471
11	685583	41	905695	71	190583	101	252128	131	150781	161	046297
12	149190	42	340713	72	013089	102	010525	132	388425	162	015563
13	233634	43	120138	73	538729	103	107691	133	327464	163	423939
14	271472	44	500192	74	660682	104	427311	134	789149	164	212198
15	083584	45	394692	75	591211	105	072846	135	450429	165	377554

Figur 13. Äldre TAN-kodtabell (Transaktionsnummer, 2023).



Figur 14. Illustration av en "phishing attack" (What Is a Phishing Attack?, n.d.).

3.1.5 "Payment Service Directive" (PSD)

"Payment Service Directive" PSD är ett EU-direktiv för digitala betalningstjänster som ersattes av PSD2. PSD2 trädde i kraft 2018 och direktiven gäller för hur betalningstjänster får hantera exempelvis autentiseringar (*Payment Services Directive*, 2024). Ett av direktiven är att varje inloggning med en betaltjänst som visar kontoinformation bör ha SCA "Strong Customer Authentication". Det finns omständigheter som kan minska kraven men genom

användning av till exempel OOB-autentisering säkerställs SCA (Commission Delegated Regulation, 2018). 2023 lades även PSD3 fram som förslag och väntas träda i kraft runt år 2026.

"Strong customer authentication" (SCA) som specificeras inom PSD-direktivet ställer relativt höga krav på autentiseringar. Kraven gäller inte enbart på själva autentiseringen utan även loggning och vilken data som måste sparas för varje typ av autentisering. Om direktivet inte efterföljs medför detta höga straffavgifter, upp till 4 % av företagets årliga vinst. Enligt PSD2 måste en autentisering bestå av minst två olika typer av autentiseringar (Tomych, 2024). En viktig poäng att nämna är att PSD2-direktiven specificerar minimikraven och att företag kan välja att implementera högre krav på autentisering utifrån egna behov och bedömningar.

3.2 "Three Domain Secure" (3DS)

3DS är ett protokoll för att identifiera och autentisera kortanvändare vid köp över internet. VisaInc skapade protokollet 3DS 1.0 redan år 2001. Syftet med protokollet är att minska mängden bedrägeriförsök vid "card not present" CNP-betalningar. Exempel på sådana transaktioner är näthandel eller återkommande betalningar då kunden redan uppgett sina kontouppgifter. Eftersom kortinformation är statisk och på så vis sårbar för attacker läggs en extra autentisering till vid kortköp. Alla betalningar på nätbutiker omdirigeras till betalningens kortutgivare för att utföra en autentisering. När identifieringen är klar kan betalningen hanteras med vetskap om att det är kortinnehavaren som utför betalningen. Protokollet har fått ta emot en hel del kritik eftersom det gör betalningar onödigt komplicerade för kunder och 2016 kom 3DS 2.0 för att förenkla processen (Ali et al., 2020).

3DS 2.0, också känt som EMV 3DS, liknar 3DS 1.0 men med några uppdateringar för att exempelvis underlätta flödet i vissa situationer då stark autentisering inte är nödvändig. Syftet med EMV 3DS är att verifiera kortinnehavarens identitet och samtidigt validera att kontoinformationen är korrekt (*EMV® 3-D Secure Protocol and Core Functions Specification*, 2023).

Som namnet antyder är 3DS-systemet byggt på tre domäner: ”Issuer Domain”,“Interoperability Domain” och “Acquirer Domain”. Dessa tre domäner separeras med blå bakgrund i figur 15. De olika domänerna kan mycket förenklat förklaras som köpare, transaktionshanterare och försäljare.

3.2.1 “Issuer domain”

“Issuer domain” består av fyra komponenter: kortinnehavare, konsumerande enhet, “Access Control Server” (ACS) och “Issuer”(utfärdare). ACS kontrolleras av "Issuer", och den tillhandahåller alla regler för autentisering. I den rollen ingår att kontrollera exempelvis kortnummer och dess legitimitet, samt att verifiera konsumentens identitet. ACS kan vara uppdelad i många olika servrar för att separera funktionalitet. “Issuer” är en finansiell institution som erbjuder korttjänster till sina kunder (*EMV® 3-D Secure Protocol and Core Functions Specification*, 2023). I figur 15 återfinns “Issuer domain” längst till vänster. Det är förenklat sett kunden som utför en betalning och banken som godkänner sin kunds betalning.

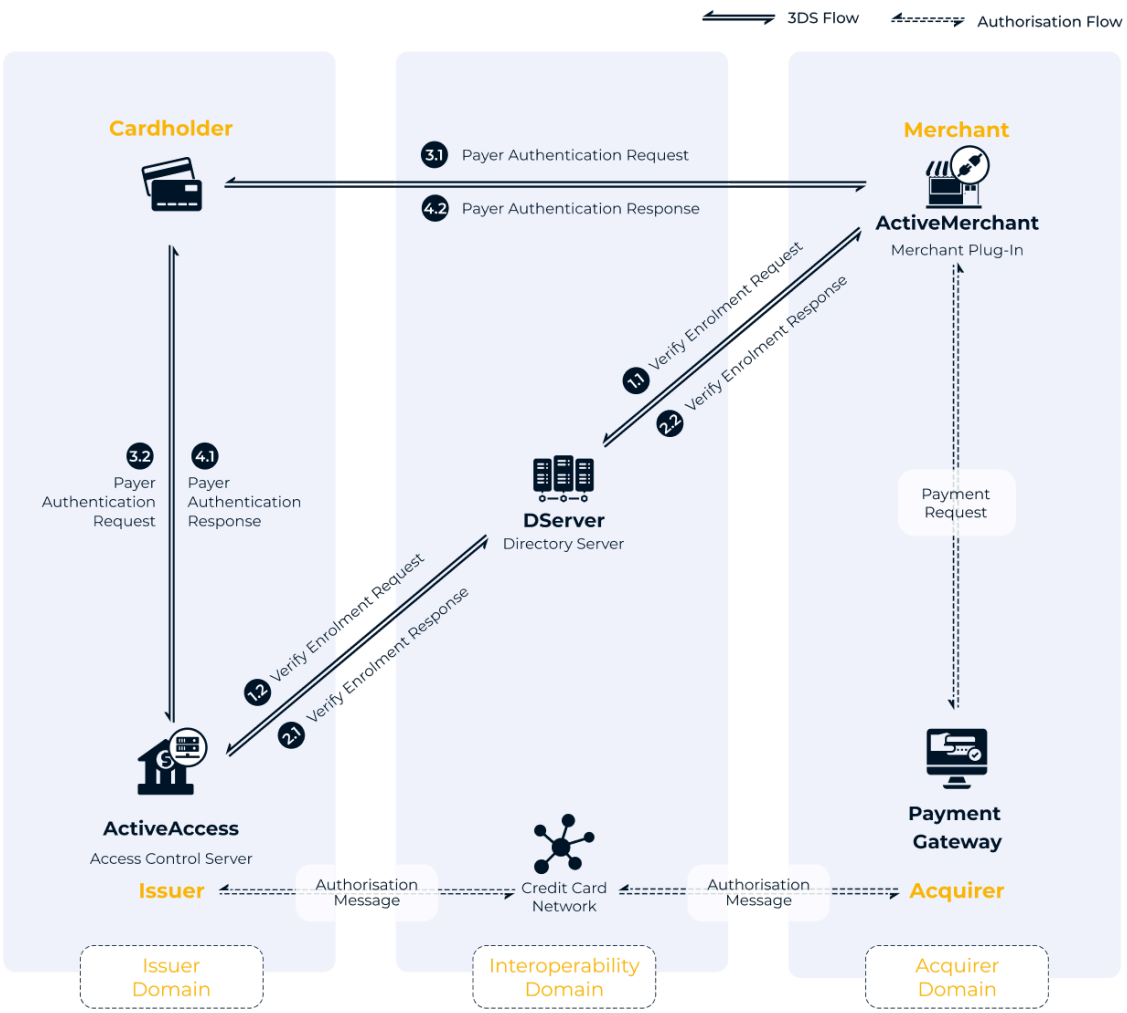
3.2.2 “Interoperability domain”

“Interoperability domain” är uppbyggt av “Directory Server” (DS) , “Directory Server Certificate Authority” (DS CA) och ”Authorization System". DS huvudsakliga syfte är att skicka information mellan 3DS-applikationer och ACS. Med hjälp av DS CA och “Authorization System” ser systemet till att all trafik är legitim och korrekt (*EMV® 3-D Secure Protocol and Core Functions Specification*, 2023). Det här visualiseras i mittsektionen i figur 15. Det är infrastrukturen som hanterar kommunikationen mellan betalare och betalningsmottagare. Det är korttillverkare som ansvarar för denna del av 3DS-protokollet.

3.2.3 “Acquirer domain”

“Acquirer domain” består i huvudsak av tre delar: “3DS Requestor Environment”, “3DS Integrator” och “Acquirer”. “3DS Requestor Environment” kan förenklat ses som flera små delar eller applikationer som integrerats med till exempel en hemsida där 3DS-betalningar erbjuds. “3DS Integratorn” underlättar integrationen mellan exempelvis hemsidan och de olika 3DS-applikationerna. ”Acquirer” är banken som har en relation till försäljaren som har

hemsidan. Den här sektionen är mottagaren av betalningen och ses till höger i figur 15 (*EMV® 3-D Secure Protocol and Core Functions Specification*, 2023).



Figur 15. Flödesdiagram av 3DS 2.0-protokollet (3D Secure Authentication, n.d.).

4. UTVECKLINGSPROCESS

4.1 Förberedelser

Innan utvecklingen av själva applikationen kunde börja hade vi ett start up-möte där vi gick igenom kravspecifikationer och den tilltänkta strukturen. Planeringen runt detta projekt var till stor del redan genomförd och Jira-stories, som är enskilda arbetsuppgifter eller mål i projektet, var redan skapade när vi fick detta projekt. Jira är ett verktyg för projektledning och spårning av fel. Det tillåter användare att skapa projekt, tilldela arbetsuppgifter, sätta tidsfrister, spåra framsteg och samarbeta inom ett team.

Arbetets huvuddelar bestod av att skapa en OpenAPI-specifikation för att initialt specifiera sökvägar och operationer samt datastrukturen för API:et, implementera den interna servern och alla tillgängliga autentiseringsmöjligheter och implementera den externa servern med tillhörande JWT-signering och -kryptering.

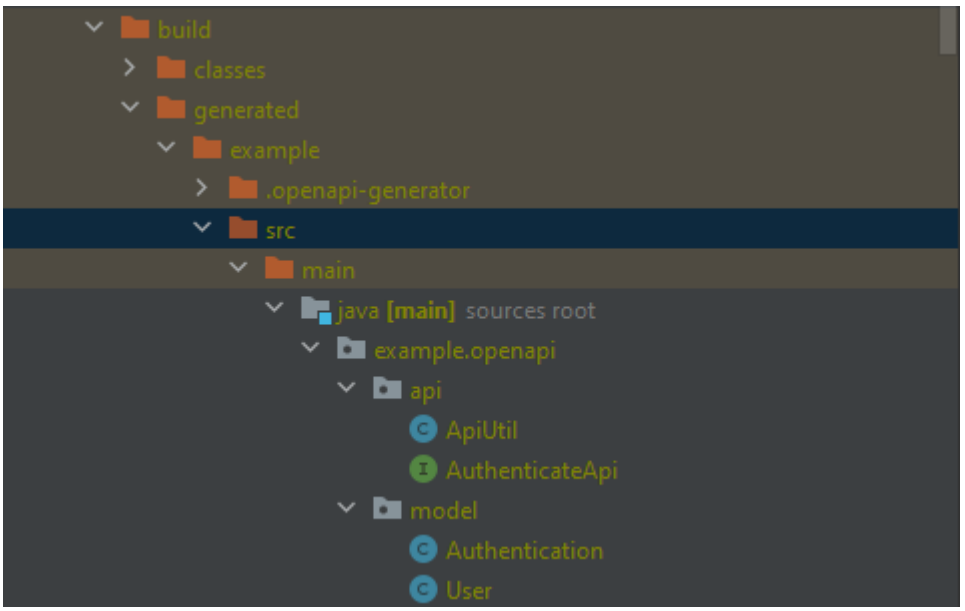
4.2 Applikationens OpenAPI-specifikation

Den initiala fasen av applikationsutvecklingen fokuserade på framtagandet av en OpenAPI-specifikation. Denna specifikation skapades genom att analysera och integrera exempel på förfrågningar och svar som tillhandahållits av leverantören av "Access Control Server" (ACS). Den slutliga OpenAPI-specifikationen utgjorde sedan grunden för genereringen av serverkod för applikationens interna system samt klientkod för användning i externa system. Denna generering möjliggjordes via konfiguration av en kodgenerator som specificerades i projektets byggkonfigurationsfil, vilket illustreras i Figur 16. Denna metodik möjliggjorde en standardiserad och effektiv utvecklingsprocess och säkerställde enhetlighet mellan systemkomponenterna.

Figur 16. Konfigurerad generator i byggverktyget Gradle skriven i Groovy. Tabell 2 beskriver konfigurationsparametrarna.

Tabell 2. Beskrivning av parametrar som ses i figur 16.

Parameter	Beskrivning
generatorName	Anger vilken kodgenerator som ska användas. I detta fall används "spring" för att generera serverkod. För att generera en klient i Java anges i stället "java" som generator. (*Generators List*, 2024)
inputSpec	Sökvägen till OpenAPI-specifikationsfilen som används för att generera koden.
outputDir	Sökvägen där den genererade koden ska sparas.
apiPackage	Det paketnamn som används för sådana API-relaterade klasser som ses i figur 2.
modelPackage	Det paketnamn som används för sådana modellklasser som ses i figur 3.
configOptions	En lista över konfigurationsalternativ som styr hur koden genereras.



Figur 17. Genererade API-gränssnitt och komponenter.

4.3 Intern autentiseringsserver

Autentiseringsservern är utformad för att hantera och säkra autentiseringsprocessen och kravet på stark kundautentisering specificerat i PSD2-regelverket.

4.3.1 Databas

Trots att den interna applikationens arkitektur är utformad för att vara tillståndslös, kräver efterlevnad av relevanta regelverk att viss data lagras temporärt under användarens autentiseringsprocess. Denna lagring säkerställer att obligatoriska data behålls genom hela autentiseringsflödet, för att sedan raderas antingen när processen är fullbordad eller avbruten. För detta implementeras en relationsdatabas som placeras i AWS.

Användningen av Flyway som verktyg för databasversionering och migration medförde ett behov av att upprätta en lokal containerbaserad databasmiljö. Denna lösning valdes framför användningen av den instansierade databasen i testmiljön på AWS. Huvudanledningen till detta beslut var att undvika potentiella konflikter och störningar i testmiljön orsakade av automatiska schemaändringar och datamigrationer. Genom att använda en lokaliserad containerbaserad databas i utvecklingsmiljön kunde vi isolera strukturella förändringar och förenkla processen för återställning till tidigare databasscheman utan att vara beroende av manuella ingrepp.

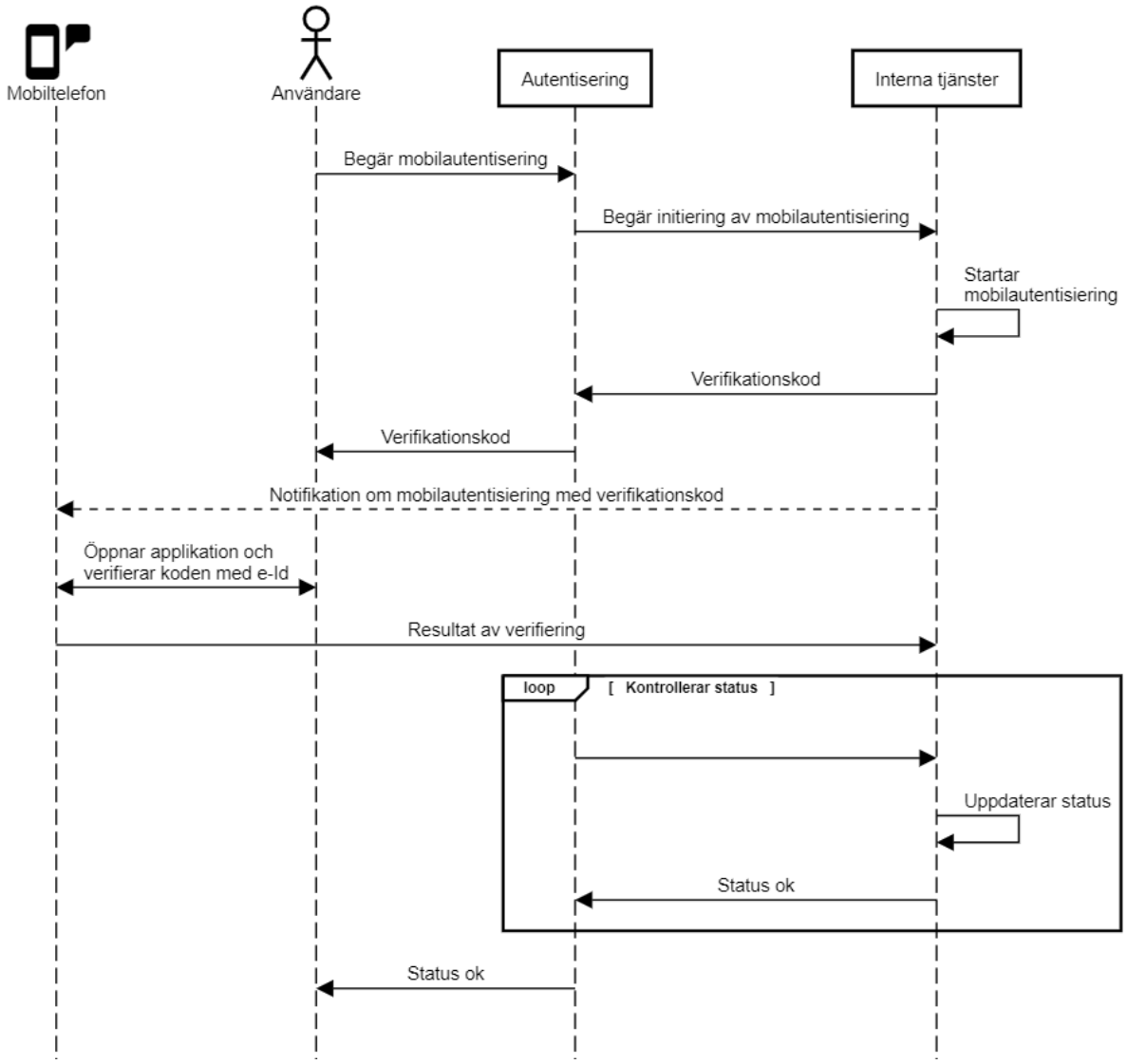
För att förverkliga detta valde vi att implementera Testcontainers. Med Testcontainers kunde vi dynamiskt konfigurera och starta en containerbaserad databas som en del av utvecklingsmiljön. Denna databas kunde konfigureras för att antingen återanvändas eller startas på nytt med varje omstart av applikationen, vilket tillhandahöll en flexibel och kontrollerbar utvecklingsmiljö som emulerar den instansierade databasen i testmiljön.

4.3.2 Integration mot autentiseringstjänster

Samtliga autentiseringsmetoder som används i applikationen finns redan implementerade, vilket innebar att vi endast behövde utföra en integration av existerande klientbibliotek för att skapa två enhetliga autentiseringsmetoder. Dessa system är utformade för att möjliggöra multifaktorautentisering, baserat på kombinationen av något som användaren känner till och något som användaren besitter. Systemen uppfyller i dagsläget kraven för SCA inom PSD2 eftersom de båda faller under kategorin multifaktorautentisering bestående av två olika former för autentiseringen.

4.3.2.1 Mobilautentisering

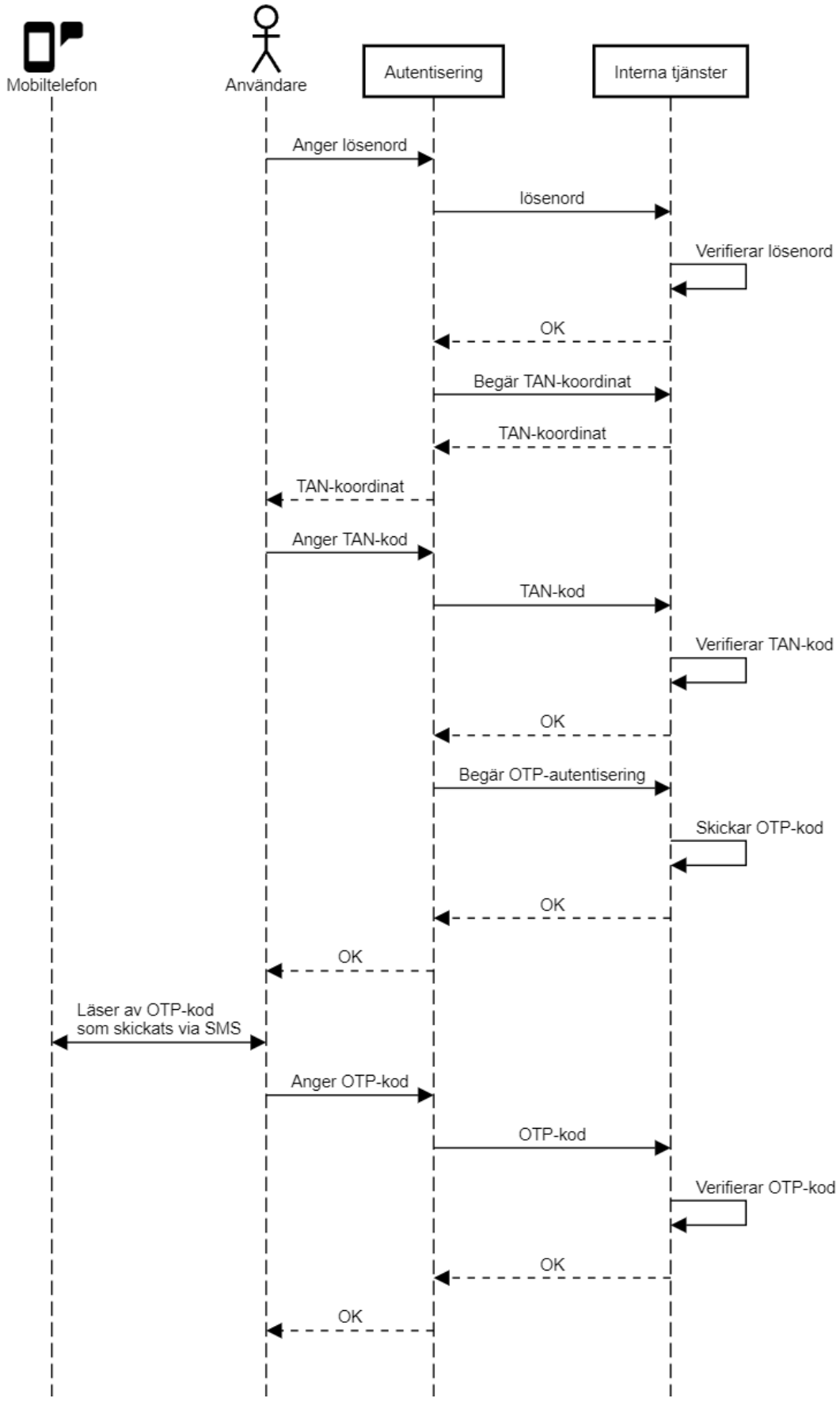
Den interna tjänsten för mobilautentiseringen är specificerad enligt OAS, vilket förenklade integration mot denna tjänst. En Java-klient genererades från OAS. Mobilautentisering är endast möjlig om användaren har registrerat sin mobiltelefon i systemet. Processen fungerar så att när en mobilautentisering begärs, skickas en förfrågan ut till användarens mobil. Användaren identifierar sig i sin mobilapplikation. Mobilapplikationen skickar statusen för identifieringen till våra interna system. Vår applikation skickar kontinuerligt förfrågningar till den interna tjänsten för att se om autentiseringen lyckades. Nedan i Figur 18 finns ett sekvensdiagram som visar en förenklad version av hur mobilautentiseringsprocessen går till.



Figur 18. Sekvensdiagram över mobilautentisering.

4.3.2.2 Integration av PIN, TAN och OTP

Integrationen av PIN, TAN och OTP i vår autentiseringsserver baseras på befintliga klientbibliotek. Under integrationsprocessen uppstod dock flera tekniska utmaningar, främst versionkonflikter i ramverket Spring mellan server och klientbiblioteken. För att hantera dessa problem utvecklade vi anpassade klienter med manuell konfiguration. När dessa utmaningar var lösta kunde vi framgångsrikt integrera de tre autentiseringsmetoderna till en enhetlig lösning för multifaktorautentisering (MFA). En förenklad illustration ses i figur 19.



Figur 19. PIN/TAN/OTP-sekvensdiagram.

4.3.3 Integrationstester

Integrationstester är en viktig del av programvarutestning där enskilda enheter, moduler eller komponenter kombineras och testas som en grupp. Syftet med dessa tester är att identifiera problem i samspelet mellan sammanlänkade komponenter inom ett programvarusystem. Genom integrationstester kan utvecklare upptäcka fel som inte nödvändigtvis upptäcks vid enhetstestning, vilket oftast fokuserar på isolerade delar av programmet.

För den interna autentiseringsservern utformades ett omfattande integrationstest för att simulera kommunikationen mellan “Access Control Server” (ACS) och autentiseringsservern. För att uppnå detta utvecklade vi en dedikerad testklient som utförde anrop till servern. Detta säkerställer att vi kan verifiera och validera interaktioner och datatransaktioner mellan de två systemen.

Med tanke på databasens centrala roll i serverns förmåga att operera tillståndslöst och dess uppdateringar och versionshantering genom Flyway, beslutade vi att inkludera en containerbaserad databas i våra integrationstester. För detta ändamål utnyttjade vi den befintliga konfigurationen för Testcontainers som används för utvecklingsmiljön.

4.4 Extern server

Inledningsvis skapades en OpenAPI-specifikation (OAS) för att generera serverdelen, medan specifikationen för den interna servern används för att generera en klient. Den externa servern är öppen för trafik från adresser utanför det skyddade interna nätverket och hanterar kommunikationen mellan Access Control Server (ACS) och den interna autentiseringstjänsten. För att säkerställa säkra dataöverföringar och upprätthålla informationens integritet används JOSE-objekt som är asymmetriskt krypterade och signerade genom att både vår server och “Access Control Server” (ACS) exponerar publika nycklar genom en JWKS “end point”.

5. DISKUSSION

I detta kapitel diskuterar vi de utmaningar och lärdomar som har framkommit under utvecklingen av applikationen. Vi redogör för våra slutsatser om projektets framgångar och de svårigheter vi stött på, samt reflekterar över våra personliga erfarenheter och tankar kring de val vi gjort och hur de påverkade slutresultatet.

5.1 Slutsats

Skapandet av en applikation för autentisering av kortbetalningar i en 3DS-lösning är utmanande och komplext, inte minst utgående från ett säkerhetsperspektiv. Det finns många olika ramverk som måste synkroniseras och arbeta tillsammans. Det är avgörande att hålla sig uppdaterad på protokoll och lagar. Dessa lagar och protokoll behöver läsas noggrant, tolkas, förstås, och förklaras. Lagarna och protokollen uppdateras relativt ofta vilket ställer krav på fortsatt utveckling för att efterleva dessa faktorer. För att lyckas med ett så här omfattande projekt är det många personer och olika team som behöver koordineras. Ytterligare aspekter som behöver tas i beaktande är t.ex. utmaningar som att integrera äldre kod med nyare system och ramverk samt att applikationens struktur bör underlätta framtida utveckling

Projektet är lyckat och vi är nöjda med resultatet. Vi har producerat en bra applikation som möter förväntningarna från kunden och oss själva. Detta hade varit svårt att uppnå utan all expertis och hjälp som funnits tillgänglig runt omkring oss i vårt team.

5.2 Personliga tankar

Valet att dela upp applikationen i en extern och en intern del baserade sig på tanken att det skulle underlätta testning av applikationen. Den grundtanken var kanske något felaktig och det hade gått lika bra att testa applikationen med en enhetlig applikation istället. Att använda de senaste versionerna av olika ramverk skapade en hel del problem. Att lösa de problemen tog mer tid än väntat. Det finns en balans mellan att skapa en applikation som använder de senaste versionerna och tiden det tar att integrera äldre versioner. Samtidigt ger dock användning av det senaste en förhoppning om hållbarhet i ett längre perspektiv. I efterhand

sett borde vi istället för att anpassa vår applikation satt mer tid på de äldre bakomliggande tjänsterna och gjort dem kompatibla med nyare versioner av till exempel Spring Boot.

I utvecklingen av applikationen har ingått många olika element, såsom generering av klienter, skapandet av kontrakt, kryptering, felhantering, koordination av olika personer från olika team och testning. Det har varit otroligt lärorikt att bygga denna applikation från grunden och få se helheten med hur många olika delar som ska samspela.

KÄLLFÖRTECKNING

3-D Secure. (2024, April 12). Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=3-D_Secure&oldid=1218489910

3D Secure authentication. (n.d.). GPayments. Retrieved April 13, 2024, from https://www.gpayments.com/about/3d-secure/

Ali, M. A., Groß, T., & van Moorsel, A. (2020). Investigation of 3-D Secure's Model for Fraud Detection. In *arXiv [cs.CR]*. arXiv. http://arxiv.org/abs/2009.12390

Commission Delegated Regulation, (2018/389). The European Commission. (2018). https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32018R0389

EMV® 3-D secure protocol and core functions specification. (2023, May 30). EMVCo. https://www.emvco.com/specifications/emv-3-d-secure-protocol-and-core-functions-specification-6/

Factor, P. (2021, January 22). *Managing database changes using Flyway: an Overview*. Redgate. https://www.red-gate.com/hub/product-learning/flyway/managing-database-changes-using-flyway-an-overview

Fritchey, G. (2020, July 30). *Flyway: Naming patterns matter*. Redgate. https://www.red-gate.com/blog/database-devops/flyway-naming-patterns-matter

Generators List. (2024, May 7). OpenAPI Generator. https://openapi-generator.tech/docs/generators/

Halbritter, M. (2023, June 23). *Improved testcontainers support in Spring boot 3.1*. Spring Blog. https://spring.io/blog/2023/06/23/improved-testcontainers-support-in-spring-boot-3-1

Identity Management Institute. (2022, May 24). *Out-of-band authentication (OOBA)*. Identity Management Institute. https://identitymanagementinstitute.org/out-of-band-authentication-ooba/

Jones, M. B. (2015, May). *RFC 7517: JSON Web Key (JWK)*. IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc7517

Jones, M. B., Bradley, J., & Sakimura, N. (2015, May). *RFC 7519: JSON Web Token (JWT)*. IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc7519

JOSE - JSON object signing and encryption. (2021, September 22). *Red Hat*. https://www.redhat.com/en/blog/jose-json-object-signing-and-encryption

Kumar, R. (2020, May 9). *Introduction to Amazon web services*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-amazon-web-services/

Markovic, B. (2023, January 16). *What is Gradle and why do we use it as Android developers?* Medium. https://medium.com/@banmarkovic/what-is-gradle-and-why-do-we-use-it-as-android-developers-572a07b3675d

Masas, R., Yohann, Omri, Levy, M., Sofia Naer, Johnston, D., Hasson, E., Gabi Stapel, & Cohen, O. (2024). *What is MITM (Man in the Middle) Attack*. Learning Center; Imperva Inc. https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/

MFA vs. 2FA: what's the difference? (2022, September 20). NordLayer. https://nordlayer.com/blog/mfa-vs-2fa-whats-the-difference/?gad_source=1&gclid=Cj0KCQjwlN6wBhCcARIsAKZvD5iIypX80ekjPLY7ulchay3L9TMty8rXYbriZLBc81DuK0vgFO015ogaAgzYEALw_wcB

One-time password. (2024, February 27). Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=One-time_password&oldid=1210546794

Payment Services Directive. (2024, March 27). Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Payment_Services_Directive&oldid=1215859620

Spring Framework - Overview. (n.d.). Tutorialspoint. Retrieved April 10, 2024, from https://www.tutorialspoint.com/spring/spring_overview.htm

Testcontainers. (n.d.). Testcontainers. Retrieved April 14, 2024, from https://testcontainers.com/

Tomych, I. (2024, January 21). *PSD3 vs PSD2: What it means for the payment sector*. DashDevs. https://dashdevs.com/blog/psd3-vs-psd2-regulations-what-new-eu-directive-means-for-payment-

sector/

Transaction authentication number. (2023, December 17). Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Transaction_authentication_number&oldid=1190431740

Transaktionsnummer. (2023, October 14). Wikipedia, The Free Encyclopedia. https://de.wikipedia.org/w/index.php?title=Transaktionsnummer&oldid=238143325

What is a phishing attack? (n.d.). Cloudflare. Retrieved April 14, 2024, from https://www.cloudflare.com/learning/access-management/phishing-attack/

What is java Spring Boot? (n.d.). IBM. Retrieved April 10, 2024, from https://www.ibm.com/topics/java-spring-boot

What is OpenAPI? (2023, March 31). OpenAPI Initiative. https://www.openapis.org/what-is-openapi