{ Produktrapport }

Creators = { Daniel, Rasmus, Benjamin }

Titelblad:

Deltagere: Daniel Vuust, Rasmus Thougaard og Benjamin Hoffmeyer
Projektnavn: LockHive
Dato: 2024-04-05
Skole: ZBC Ringsted
Vejleder: ?

Daniel Vuust .

Rasmus Thougaard .

Benjamin Hoffmeyer .

Indholdsfortegnelse

Læsevejledning:

Gennem denne rapport vil du få indsigt i produktdelen af vores app. Hvorfra der vil være en del diagrammer, nogle af disse diagrammer vil have en bestemt måde at læse dem på, hvorfra måden de skal læses er beskrevet lige før diagrammet placering i dette dokument.

Definitioner, akronymer og forkortelser:

- GDPR: General Data Protection Regulation.
- OWASP: The Open Worldwide Application Security Project.
- CIA: Confidentiality, Integrity, Availability.
- NIST: National Institute of Standards and Technology
- DB: Database
- HTTPS: Hypertext transfer protocol secure
- API: Application Programming Interface
- TLS: Transport Layer Security
- UI: User interface
- 2FA: Two factor authentication
- NPS: Net Promoter Score
- REST: Representational State Transfer.
- SSO: Single sign-on
- MVP: Minimal viable product
- Continuous development: Kontinuerlig udvikling er en proces, hvor softwaren bliver udviklet styk for styk og sikre at der altid er et produkt som kan bruges.
- PR: Pull request
- UI: User Interface
- PCI DSS: Payment Card Industry Data Security Standard. En international standard for behandling af betalingskort som bl.a kræver kryptering af kortdata under transmission, adgangskontrol, monitorering og logning af netværk samt andre regulatoriske krav.)

Kravspecifikation:

1. Introduktion:

Denne kravspecifikation beskriver de funktionelle og ikke-funktionelle krav til udviklingen af en password manager.

1.1 Formål:

Formålet er at udvikle en password manager, der effektivt forbedrer brugernes sikkerhed og bekvemmelighed ved at håndtere adgangskoder og minimere risikoen for datalæk og hackerangreb via komplekse, unikke, autogenererede passwords.

1.2 Scope:

Projektets omfang dækker design, udvikling og implementering af password manageren, som inkluderer automatisk generering af sikre adgangskoder, krypteret cloud-baseret opbevaring samt yderligere funktionaliteter beskrevet under funktionelle krav.

2. Lovmæssige og regulatoriske krav

2.1 Datasikkerhed:

- Vi læner os op af OWASP.org guidelines for at sikre den største sikkerhed muligt samt har vi udvalgt specifikke elementer fra NIST (minimum 8 tegn).

  [AFGRÆNSET]
- Til udvikling af betalingskort har vi udvalgt specifikke elementer fra Payment Card Industry Data Security Standard (PCI DSS) hvor vi læner os op af følgende krav:
  - Beskyttelse af kortholder data
  - Implementering af adgangskontrol
  - Sikker transmission af betalingskort data over åbent netværk

  Snitflader:
- Kommunikationen mellem app og api gøres der brug af https og tls.
- Kommunikationen mellem api og DB vil gøre brug af standard kryptering under transport

2.2 Kryptering:

- Brug af industri-standard krypteringsteknologier for beskyttelse af brugerdata.

2.3 Afgrænsninger:

- Projektet er afgrænset fra den lovmæssige krav fra GDPR
- Projektet er afgrænset for sikkerhed under transport udover det beskrevet ovenover i snitflader

3. Funktionelle krav:

Vi har beskrevet alle funktionelle krav for password manager, i en plan a, b og c hvor plan c er den mest basale version med krav, mens plan a og b er ønsker og yderligere funktionalitet.

Plan C:

- User
  - User authentication
- Generering af password til alle onlinekonti
  - Unikke
  - Minimum 8 karaktere lange og max 128 karaktere

    - Komplekse
        - Minimum et Specialtegn
        - Minimum et Stort bogstav
        - Minimum et lille bogstav
        - Minimum et tal
- Krypteret skybaseret lagring af password

Plan B:

[AFGRÆNSET]
- Opbevaring af kreditkortoplysninger

[AFGRÆNSET]
- Advarsel om potentielt password læk

[AFGRÆNSET]
- Automatisk udfyldning af inputfelter

Plan A:

[AFGRÆNSET]
- 2FA

3.1 Afgrænsninger:

- Projektets implementering af login og styring bliver afgrænset til Firebase authentication

4. Kvalitetsegenskaber:

4.1 Anvendelighed (Usability):

- Udvikle en brugervenlig app, der er intuitiv på både iOS og Android, målt ved en standardiseret skala inden for den planlagte udviklingstid.
- Intuitiv autogenerering af password, der er længere og mere komplekse end normen, indenfor tidsrammen af projektet.

4.2 Pålidelighed (Reliability):

- Skal være pålidelig og tilgængelig med minimum 99,9% oppetid i tidsintervallet 06:00-24:00.

4.3 Ydeevne (Performance):

- Skal opretholde høj performance med hurtig responstid selv under belastning.
- Systemet skal kunne skaleres for at understøtte en voksende brugerbase og datastørrelse uden at gå på kompromis med ydeevne eller sikkerhed.

4.4 Supporterbarhed (Supportability):

- Let at vedligeholde og opdatere med support for fremtidige sikkerhedsstandarder og platforme.

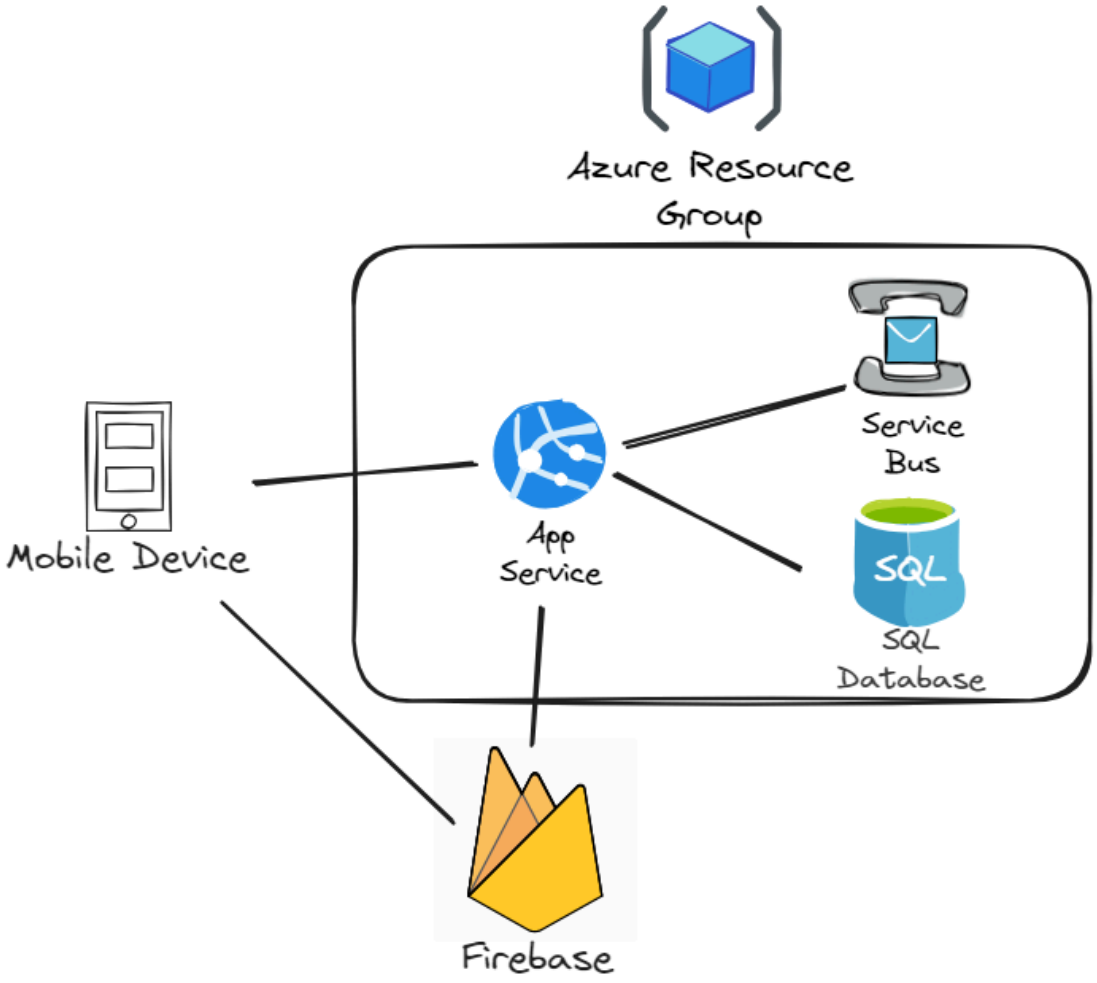
5. Design Begrænsninger:

5.1 Teknisk dokumentation:

- Al teknisk dokumentation, såsom kode kommentarer og diagrammer, skal ske på engelsk.

Teknisk produkt dokumentation:

Overordnet struktur:



[1]

Azure:

Azure er en cloud platform, der har mere end 200 produkter og cloud services, som kan hjælpe en med at skabe en løsning til projekter. Vi har valgt at bruge specifikke ting Azure tilbyder, såsom Azure Service Bus, SQL database og SQL Service. Grunden til valget af Azure er fordi,

[1] https://excalidraw.com/#room=7767be9ffa970b1b3cc7,NFiGWffKoSbmXz-D9QSb5Q

vi ikke ville være afhængig af hvor i verden vi er, og kan altid komme i kontakt med Azure Service Bus, SQL database eller SQL Service, som bliver hostet i Azure.

MsSql:

MS SQL Server er en relationel Database Management System (RDBMS) udviklet af Microsoft. En relational database er bygget på Relational Model architecture, hvor data er organiseret i tabeller og er relateret til hinanden. Vi har valgt at bruge MSSQL på en lidt anden måde end normalt. Vi har lavet tabeller ud fra vores domain modeller. F.eks har vi en user table, der indeholde alt data om en user, derudover har vi en password table, som indeholder alt data om et password, men nede i password tabellen har vi et userId, som holder styr på relationen på hvilken user der ejer et password. Relationen her er ikke en FK

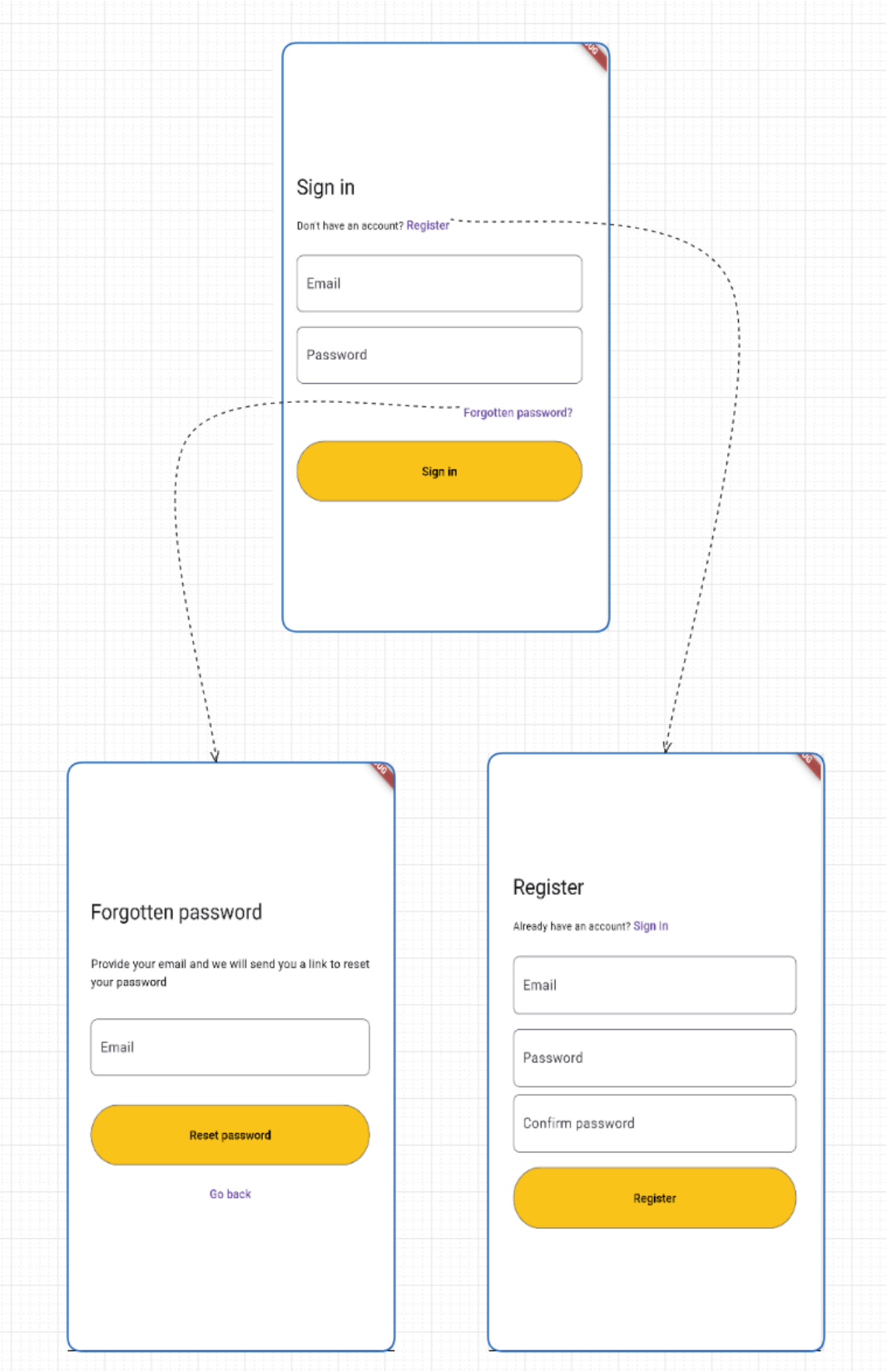
ServiceBus:

Vi gør brug af Azure Service Bus, som er vores message broker, der skal transportere beskeder fra en kø til en modtager. Eksempelvis gør vi brug af Service Bus, når vi skal oprette et password i user service. Her sender vi en command på Service Bus køen for User service. I User service har vi en handler, der modtager denne besked og begynder et stykke arbejde.

Firebase:

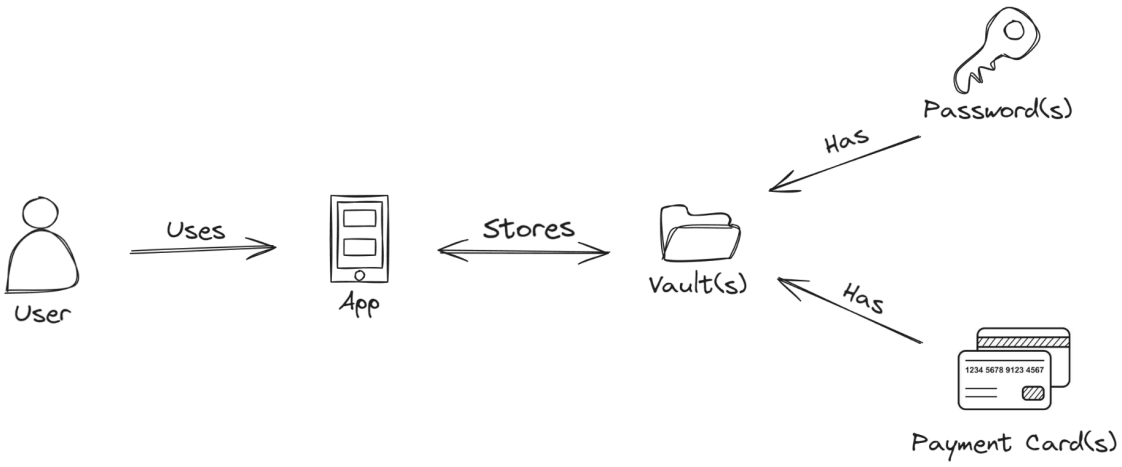
Vi har brugt Firebase til at holde styr på authentication i appen. Firebase gemmer alle login-legitimationsoplysninger for os og begrænser os derfor for de lovmæssige og sikkerhedsmæssige valg. Dog bruger vi et unikt FirebaseId som vi gemmer i vores databasen for at finde hvilken bruger vi er logget ind som og relatere til sin vault med passwords. Se Database Struktur for mere.
I appen gør brug af firebase_ui_auth[2] pakken, lavet af Google, til at håndtere UI til at logge ind, sign up, resette passwords og mere. Dette gør det også muligt at tilføje diverse forskellige SSO'er, som Firebase understøtter, uden af vi behøver at lave nogle ændringer i UI'en.
Her kan vi se den generelle Firebase UI som styr login, sign up, osv

[2] https://pub.dev/packages/firebase_ui_auth


Sign in
Don't have an account? Register
Email
Password
Forgotten password?
Sign in
Forgotten password
Provide your email and we will send you a link to reset your password
Email
Reset password
Go back
Register
Already have an account? Sign in
Email
Password
Confirm password
Register

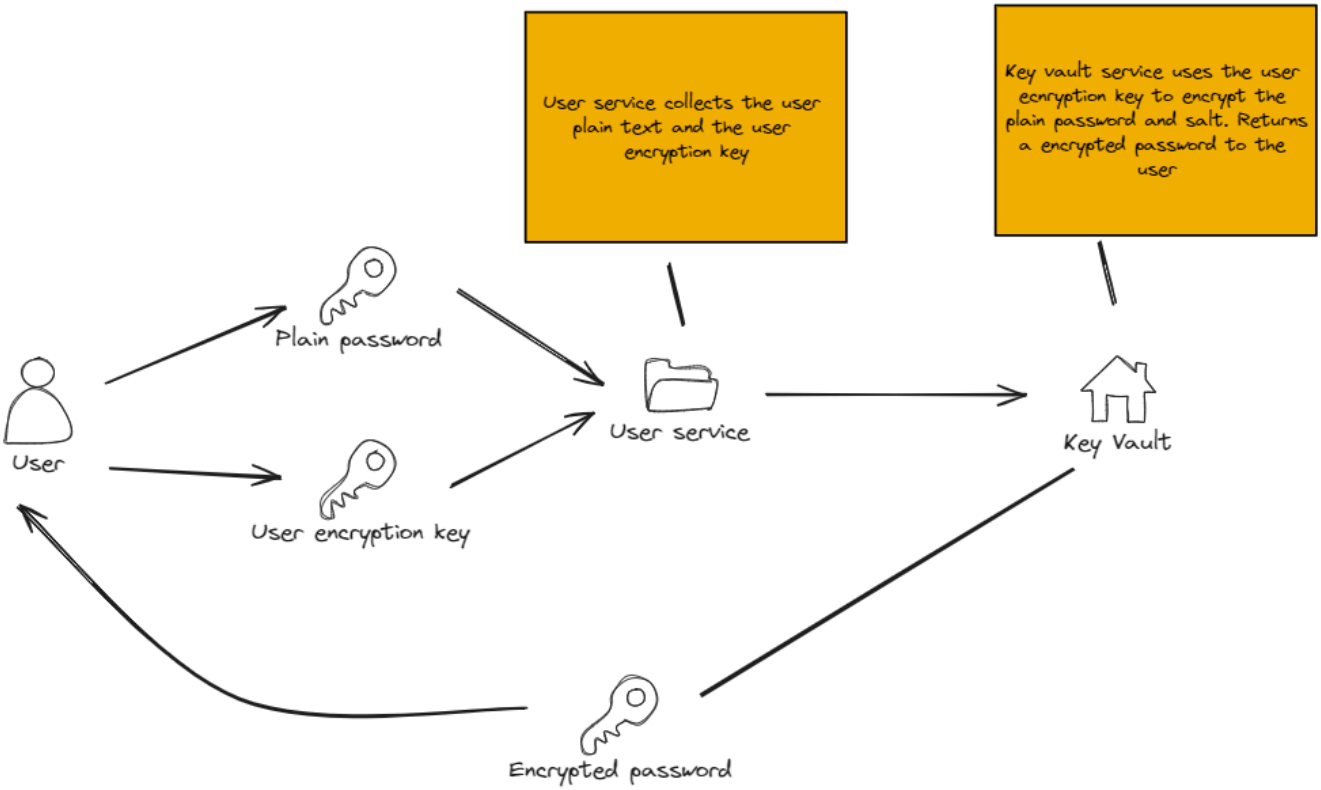
Rigt billed:

Vi har lavet et rigt billede for at have det samme billede af løsningen på problemet.

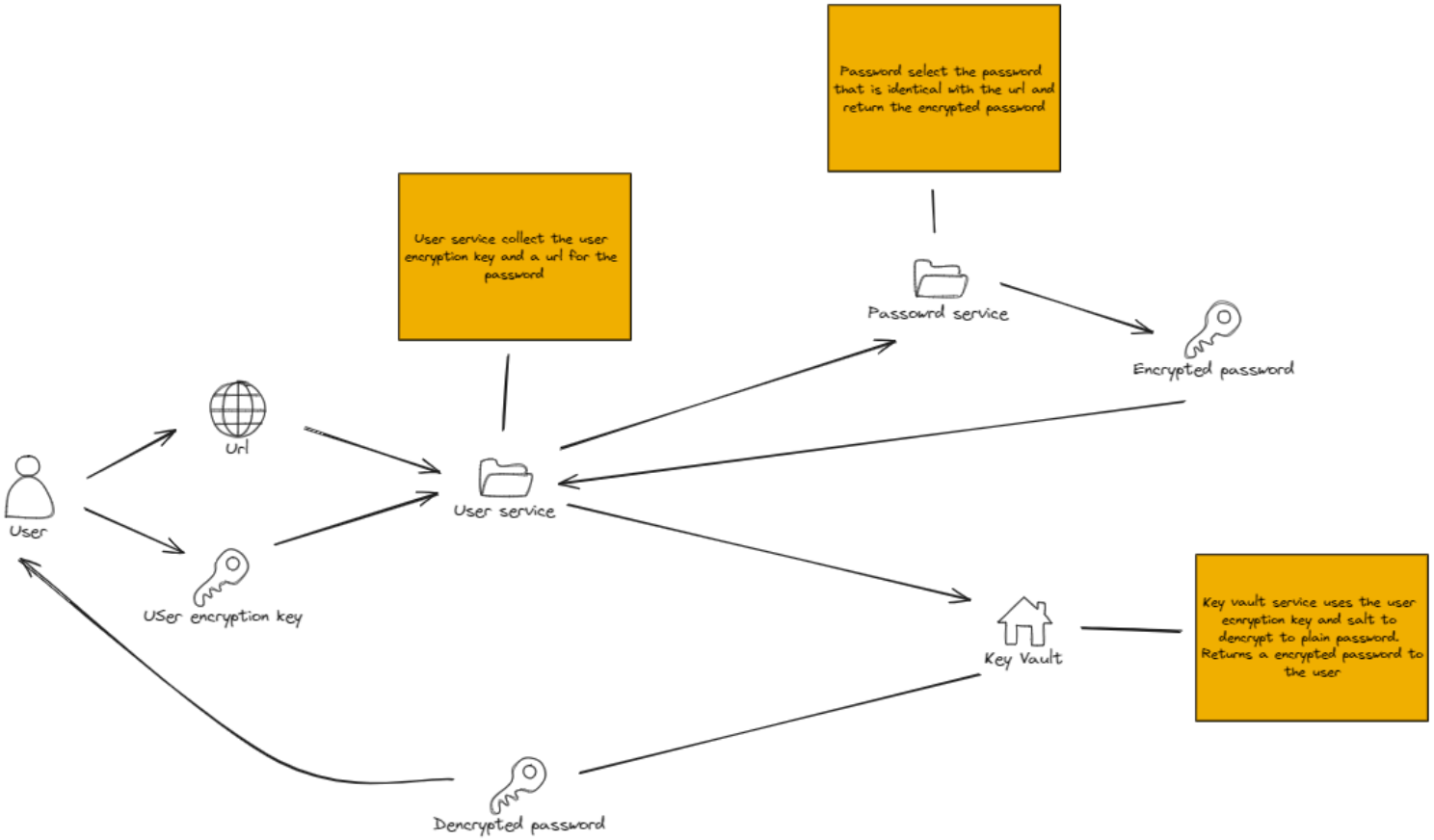


Flow password:

Oprettelse af password:

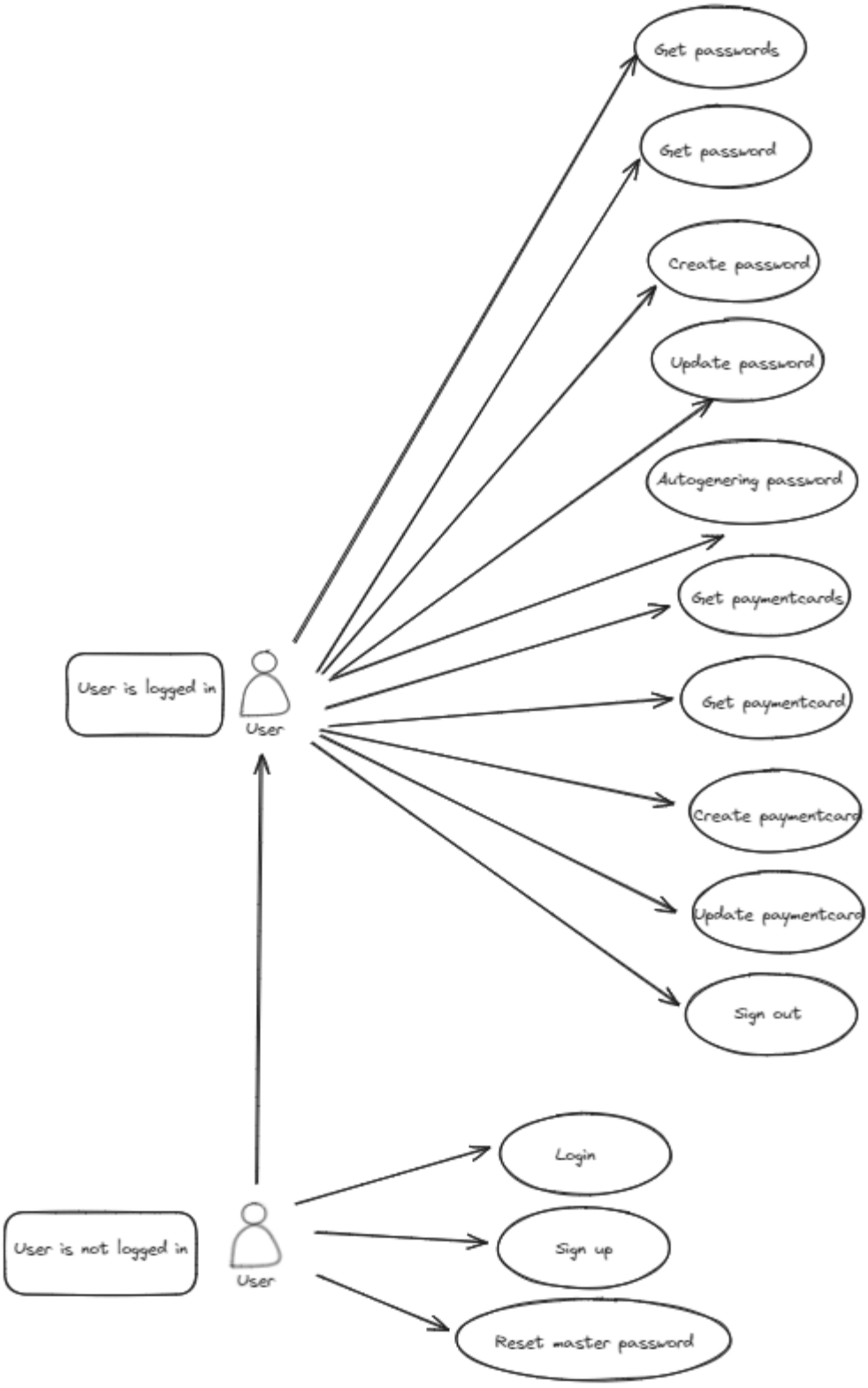


Hente password(s):



Use cases

Her ses alle uses cases for en user ved brugen af vores password manager



https://excalidraw.com/#json=1Pj0UwL1-UDGyGqZiej_I,LklxqqJ7lQiH_ZaMKVfmCw

- **Password**

Update password

Use Case navn	**Update password**
Id	1
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren
Prekonditioner	Brugen har en konto med et eller flere password
Postkonditioner	Brugerens password er opdateret
Kort beskrivelse	Aktøren ønsker at opdater et password. Aktøren indtaster de opdaterede værdier ind.
Normalt forløb	1. Brugeren åbner app 1.1. Bruger finder det ønsket password, der skal opdateres 1.1.1. Aktøren indtaster de værdier, som ønskes at opdateres 1.1.2. Aktøren trykker på gem knappen. 1.2. Systemet gemmer det opdateret password
Alternative forløb	1. Systemet giver en tilbagemelding om fejl 2. Passwordet eksisterer ikke.
Udvidelsesmuligheder	Kontrol af brugeren password overholder vores sikkerheds principper
Ikke funktionelle krav	
Åbne spørgsmål	Hvad gøres der hvis opdatering af brugerens password fejler?

Generate password

Use Case navn	**Automatisk generering af password**
Id	2
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren
Prekonditioner	Brugeren har konto på app'en
Postkonditioner	Brugeren har fået automatisk genereret password
Kort beskrivelse	Aktør forespørger om automatisk genereret password. Aktør indtaster længden på minimum 8 karakterer efterfølgende bliver der autogenereret et unikt og komplekst password
Normalt forløb	1. Brugeren åbner app 1.1. Aktøren ønsker at autogenerere password 1.1.1. Aktøren indsætter, hvor langt password skal være. Min 8 lang og maks 128

	1.1.1.1. Et unikt og komplekst autogenereret password er blevet genereret.
Alternative forløb	1. Systemet giver tilbagemelding om fejl 2. Det autogenereret password overholder ikke kriterierne
Udvidelsesmuligheder	Det genererede password opfylder kravene om minimum længde, specialtegn, tal, stort/lille bogstav.
Ikke funktionelle krav	
Åbne spørgsmål	

Create password

Use Case navn	**Create password**
Id	3
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren
Prekonditioner	1. Aktøren har en konto i password manager
Postkonditioner	Aktøren har oprettet et password i password manageren
Kort beskrivelse	Aktør ønsker at oprette et password i password manageren. Aktøren indtaster oplysninger om password og opretter password. Password bliver gemt i brugeren vault.
Normalt forløb	1. Brugeren åbner app 1.1. Aktøren ønsker opretter et password 1.1.1. Aktøren indtaster oplysninger om password 1.1.2. Aktøren trykker på gem knappen 1.2. Systemet gemmer password
Alternative forløb	1. Systemet giver tilbagemelding om fejl
Udvidelsesmuligheder	Password kan automatisk blive gemt, når aktøren opretter et password på en hjemmeside
Ikke funktionelle krav	
Åbne spørgsmål	

Get password

Use Case navn	**Get password**
Id	4
Version	1.0.0
Aktør(er)	Bruger

Trigger	Brugeren
Prekonditioner	1. Brugeren har en konto 2. Brugeren har minimum et password gemt i sin vault.
Postkonditioner	Brugeren har fået sine password
Kort beskrivelse	Aktøren ønsker at se sit password.
Normalt forløb	1. Aktøren logger ind 1.1. Aktøren får udleveret sit password
Alternative forløb	1. Aktør kan ikke logge ind 2. Systemet giver tilbagemelding om fejl
Udvidelsesmuligheder	
Ikke funktionelle krav	
Åbne spørgsmål	Hvad skal aktøren gøre, hvis aktøren ikke kan hente sine password

Get passwords

Use Case navn	**Get passwords**
Id	5
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren
Prekonditioner	3. Brugeren har en konto 4. Brugeren har minimum et password gemt i sin vault.
Postkonditioner	Brugeren har fået sine passwords
Kort beskrivelse	Aktøren ønsker at se sine passwords.
Normalt forløb	2. Aktøren logger ind 2.1. Aktøren får udleveret sine passwords
Alternative forløb	3. Aktør kan ikke logge ind 4. Systemet giver tilbagemelding om fejl
Udvidelsesmuligheder	
Ikke funktionelle krav	
Åbne spørgsmål	Hvad skal aktøren gøre, hvis aktøren ikke kan hente sine passwords

- **Login/sign up**

Login

Use Case navn	**Logge ind på app'en**
Id	6
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren åbner appen
Prekonditioner	Brugeren har en konto på app'en
Postkonditioner	
Kort beskrivelse	Aktøren ønsker at logger ind i password manageren
Normalt forløb	2. Brugeren åbner app 2.1. Brugeren bliver vist en login form 2.1.1. Brugeren logger ind og får vist sin data 2.1.2. Brugeren har glemt sit password og beder om at få det resat via en mail som bliver sendt til aktørens mail
Alternative forløb	1. Systemet giver tilbagemelding om fejl
Udvidelsesmuligheder	
Ikke funktionelle krav	
Åbne spørgsmål	

Sign up

Use Case navn	**Sign up i app'en**
Id	7
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren åbner appen og vil oprette en bruger
Prekonditioner	
Postkonditioner	
Kort beskrivelse	Aktøren åbner appen og vil oprette en bruger
Normalt forløb	3. Brugeren åbner app 3.1. Brugeren bliver vist en knap til at sign up 3.1.1. Brugeren indtaster email og password
Alternative forløb	1. Systemet giver tilbagemelding om fejl

Udvidelsesmuligheder	
Ikke funktionelle krav	
Åbne spørgsmål	

Reset password

Use Case navn	**Reset password**
Id	12
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren åbner appen og beder om nyt password
Prekonditioner	Brugeren har en konto på app'en
Postkonditioner	Brugeren har nulstillet sit master password
Kort beskrivelse	Aktøren ønsker at resette sit password
Normalt forløb	4. Brugeren åbner app 4.1. Brugeren bliver vist en login form 4.1.1. Brugeren trykker på reset password 4.1.1.1. Brugeren indtaster sin mail 4.1.1.1.1. Brugeren følger steps i mail der blev tilsendt
Alternative forløb	1. Systemet giver tilbagemelding om fejl
Udvidelsesmuligheder	
Ikke funktionelle krav	
Åbne spørgsmål	

Sign out

Use Case navn	**Sign out**
Id	13
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren åbner appen og er logget ind
Prekonditioner	Brugeren har en konto på app'en og er logget ind
Postkonditioner	Brugeren af logget ud

Kort beskrivelse	Aktøren ønsker at logge ud af sin konto
Normalt forløb	5. Brugeren åbner app 5.1. Brugeren trykker på sin konto øverst på skærmen 5.1.1. Brugeren trykker på log ud
Alternative forløb	1. Systemet giver tilbagemelding om fejl
Udvidelsesmuligheder	
Ikke funktionelle krav	
Åbne spørgsmål	

- **Payment card**

Create paymentcard

Use Case navn	**Create payment card**
Id	8
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren
Prekonditioner	2. Aktøren har en konto i password manageren 3. Aktøren har et payment card
Postkonditioner	Brugeren har oprettet et payment card
Kort beskrivelse	Aktør ønsker at oprette et payment card i password manageren. Aktøren indtaster oplysninger om payment card og opretter payment card. Payment card bliver gemt i brugeren vault.
Normalt forløb	2. Brugeren åbner app 2.1. Aktøren ønsker opretter et payment card 2.1.1. Aktøren indtaster oplysninger om payment card 2.1.2. Aktøren trykker på gem knappen 2.2. Systemet gemmer payment card
Alternative forløb	2. Systemet giver tilbagemelding om fejl
Udvidelsesmuligheder	Payment card kan blive automatisk gemt, når aktøren bruger payment card til betaling på en hjemmeside.
Ikke funktionelle krav	
Åbne spørgsmål	

Update paymentcard

Use Case navn	**Update payment card**
Id	9
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren
Prekonditioner	Aktøren har en konto med et eller flere payment card(s
Postkonditioner	Aktøren har opdateret det ønsket payment card
Kort beskrivelse	Aktøren ønsker at opdater et payment card. Aktøren indtaster de opdaterede værdier ind.
Normalt forløb	2. Aktøren åbner app 2.1. Aktøren finder det ønsket password, der skal opdateres 2.1.1. Aktøren indtaster de værdier, som ønskes at opdateres 2.1.2. Aktøren trykker på gem knappen. 2.2. Systemet gemmer det opdateret payment card
Alternative forløb	3. Systemet giver en tilbagemelding om fejl 4. Payment card eksisterer ikke.
Udvidelsesmuligheder	
Ikke funktionelle krav	
Åbne spørgsmål	Hvad gøres der hvis opdatering af brugerens payment card fejler?

Get payment cards

Use Case navn	**Get passwords**
Id	10
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren
Prekonditioner	5. Aktøren har en konto 6. Aktøren har minimum et payment card gemt i sin vault.
Postkonditioner	Aktøren kan hente sine payment card
Kort beskrivelse	Aktøren ønsker at se sine payment card.
Normalt forløb	3. Aktøren logger ind 3.1. Aktøren får udleveret sine passwords
Alternative forløb	5. Aktør kan ikke logge ind 6. Systemet giver tilbagemelding om fejl
Udvidelsesmuligheder	

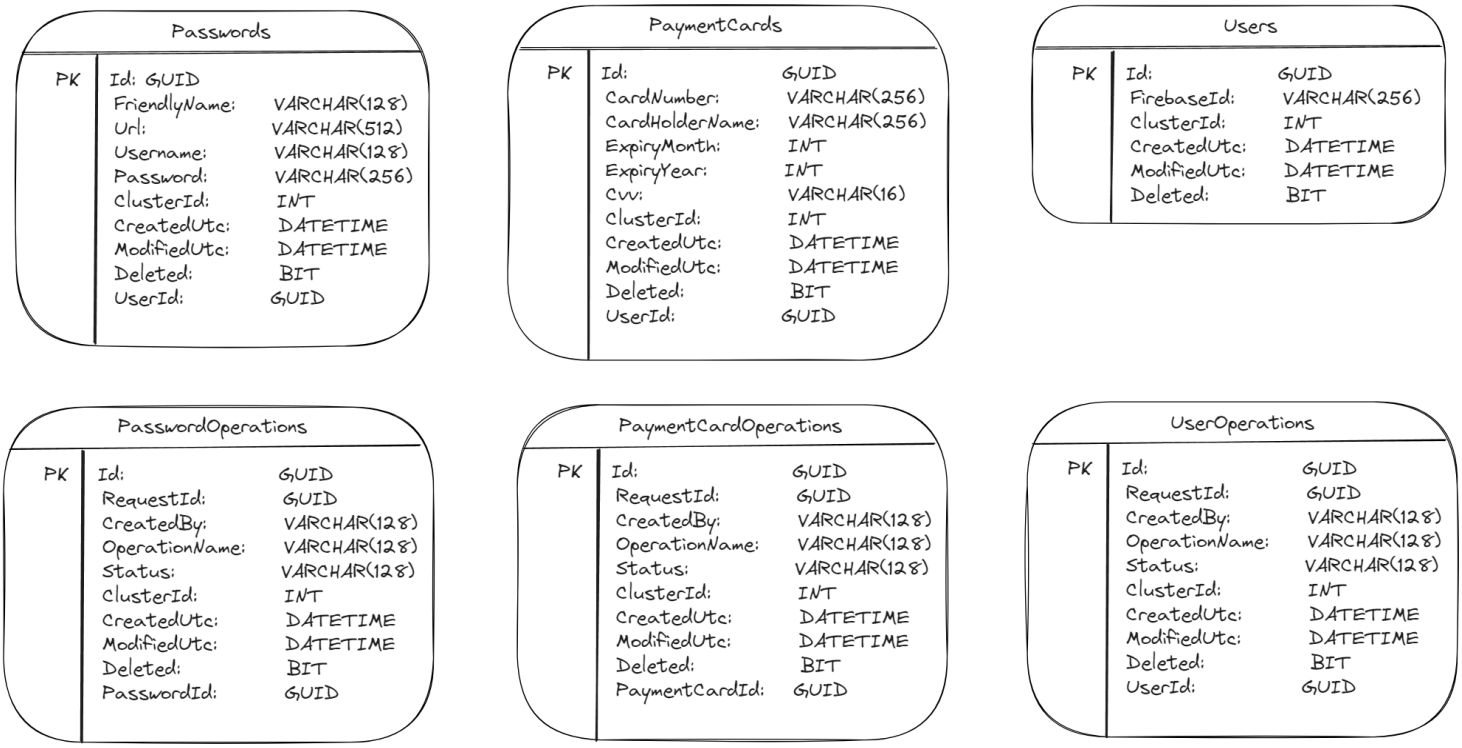
Ikke funktionelle krav	
Åbne spørgsmål	Hvad skal aktøren gøre, hvis aktøren ikke kan hente sine payment card(s)?

Get payment card

Use Case navn	**Get payment card**
Id	11
Version	1.0.0
Aktør(er)	Bruger
Trigger	Brugeren
Prekonditioner	7. Aktøren har en konto 8. Aktøren har minimum et payment card gemt i sin vault.
Postkonditioner	Aktøren har fået sine payment card
Kort beskrivelse	Aktøren ønsker at se sit payment card.
Normalt forløb	4. Aktøren logger ind 4.1. Aktøren får udleveret sit payment card
Alternative forløb	7. Aktør kan ikke logge ind 8. Systemet giver tilbagemelding om fejl
Udvidelsesmuligheder	
Ikke funktionelle krav	
Åbne spørgsmål	Hvad skal aktøren gøre, hvis aktøren ikke kan hente sine payment card

Backend:

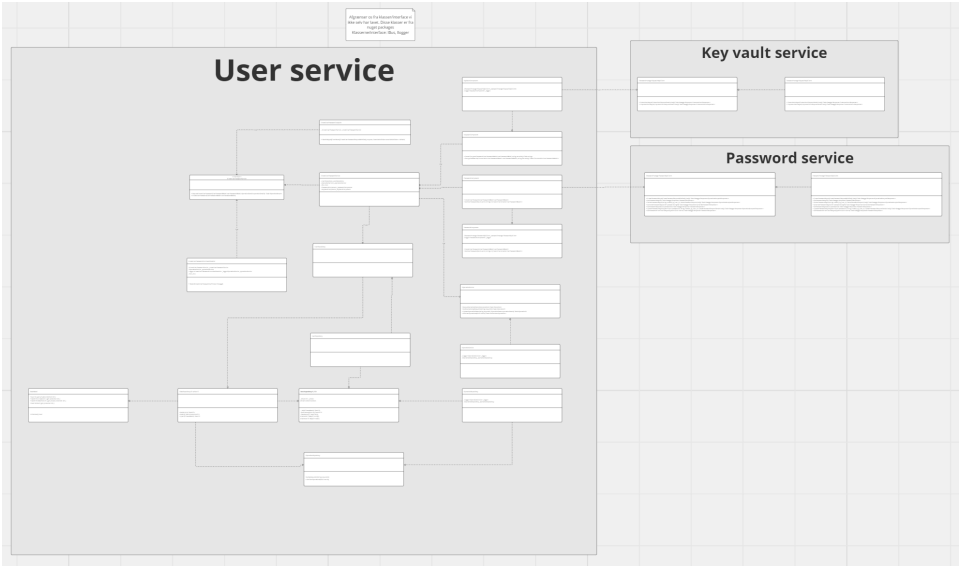
Database Struktur:



https://link.excalidraw.com/l/6tKVY9Mx7RF/AU3ItuYFhcY

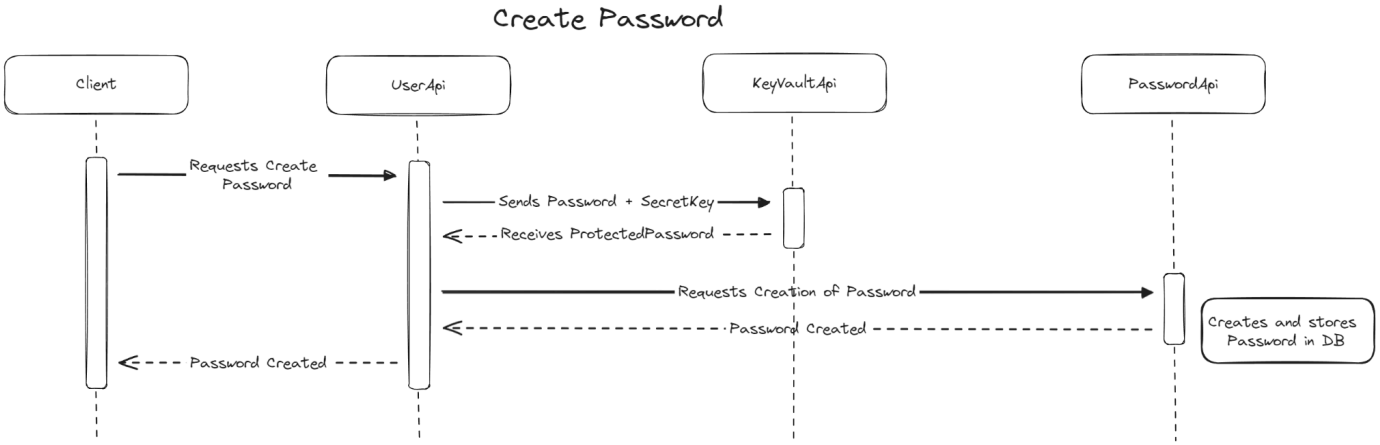
Klasse diagram:

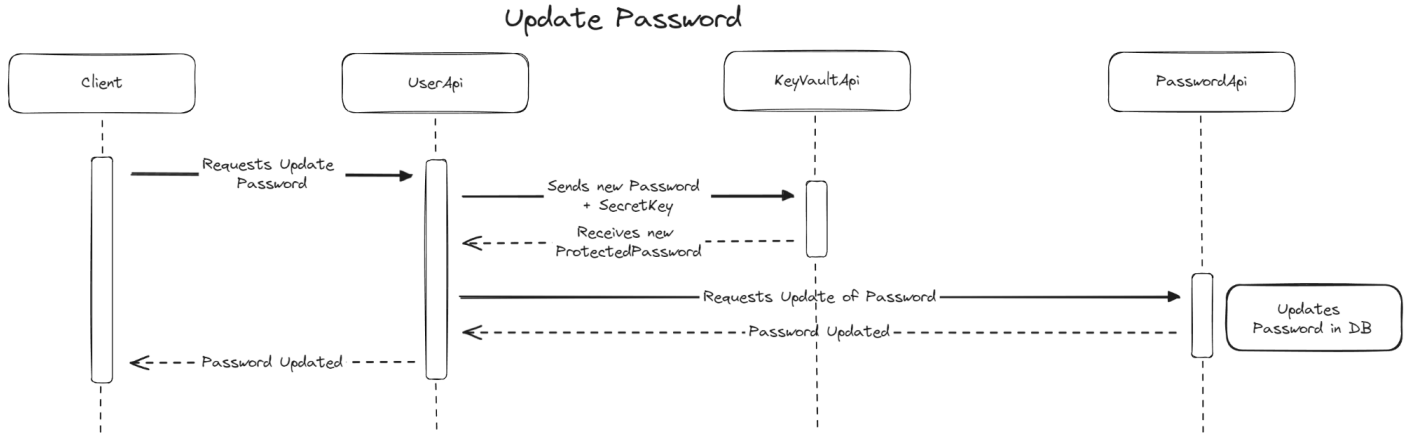
Her er klasse diagram over user service med reference til keyVault- og password-service api client. Klik på linket for at se klassediagrammet.

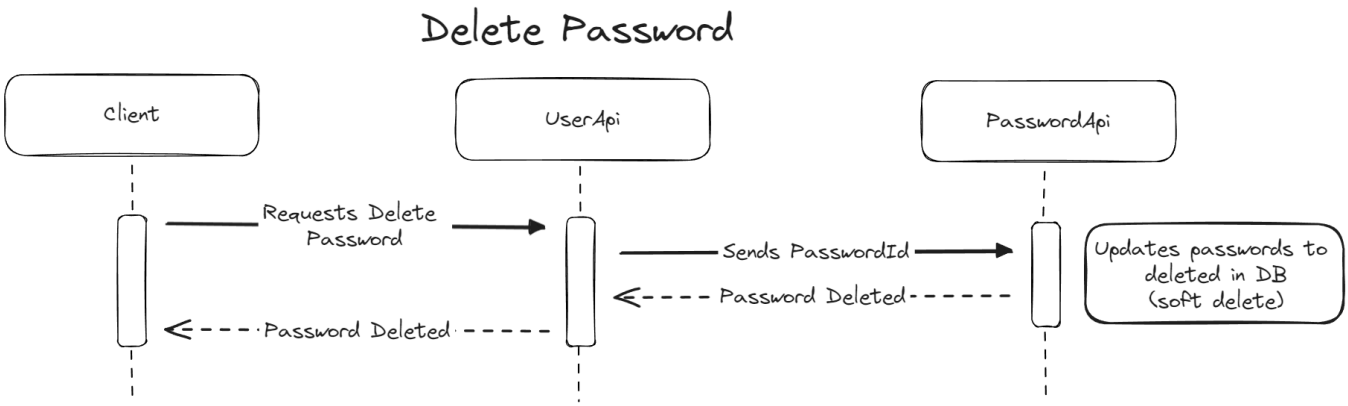


https://miro.com/app/board/uXjVKYw4KmQ=/?share_link_id=325131230635

Sekvens diagrammer:





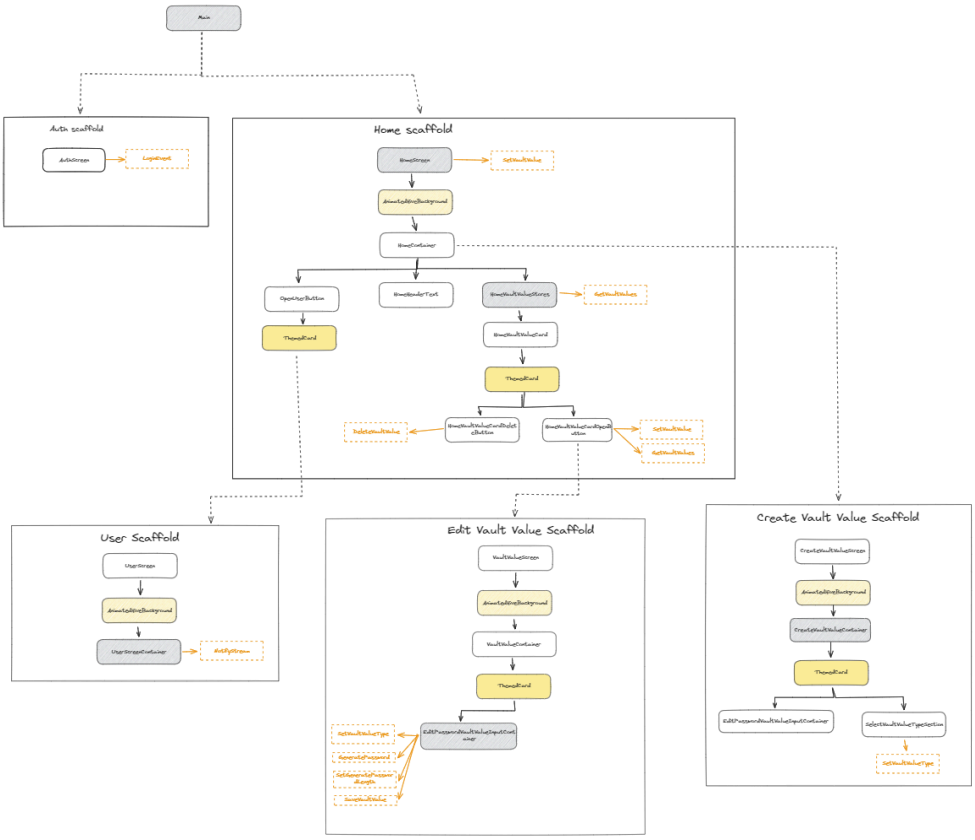


Frontend:

Strukturen over frontend'ens klasser og metoder er opdelt i to diagrammer. Første diagram er et widget diagram som beskriver hvordan widgets er sammenhængende og viser kun hvilke events der bliver kaldt fra de widgets. Det andet diagram er et klassediagram som beskriver disse klasser som bliver kaldt fra widgets og hvordan de relatere til resten af logikken i appen.

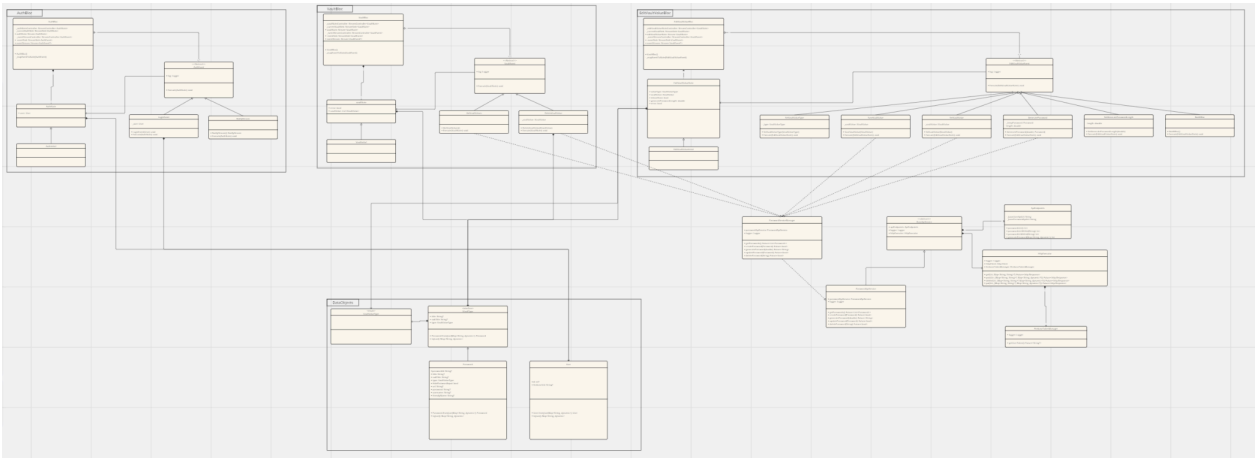
Widget diagram:

For at læse widget diagrammet korrekt er der følgende regler som blev overholdt ved diagrammering: Når en baggrund er en solid er der tale om en stateless widget mens hvis baggrunden er lavet af linjer er der tale om en stateful widget. En gullig farve betyder det at der er tale om en generisk widget som kræver en child parameter, herunder både stateful og stateless. De orange stiplede firkanter signalere et bloc-event som kan matches i nedenstående klassediagram.



https://excalidraw.com/#room=7767be9ffa970b1b3cc7,NFiGWffKoSbmXz-D9QSb5Q

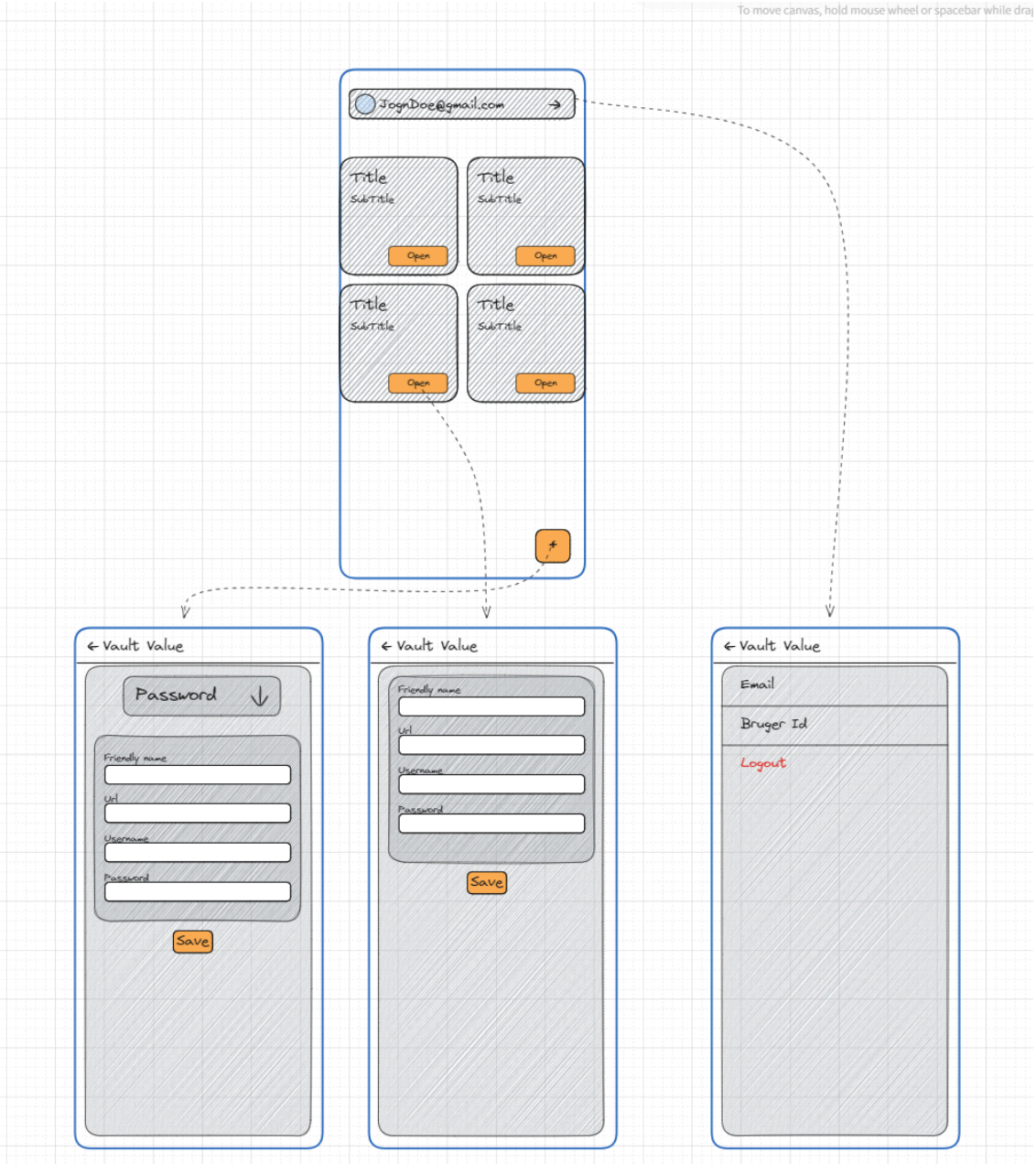
Klassediagram:



https://miro.com/app/board/uXjVKXCF4rI=/

Mockup:

Login side er exkluderet fra mockup da det er pakken firebase_ui_auth der styrer dette. Se Firebase for mere.



https://excalidraw.com/#json=YnV-TX9XQif3DmHeSdh8n,A1UIHfS5st5IZpcBT2EHIA

Testrapport:

Backend

Unit test

Her er der et eksempel på en unit test, test code coverage og overblik over test, der er vellykket.

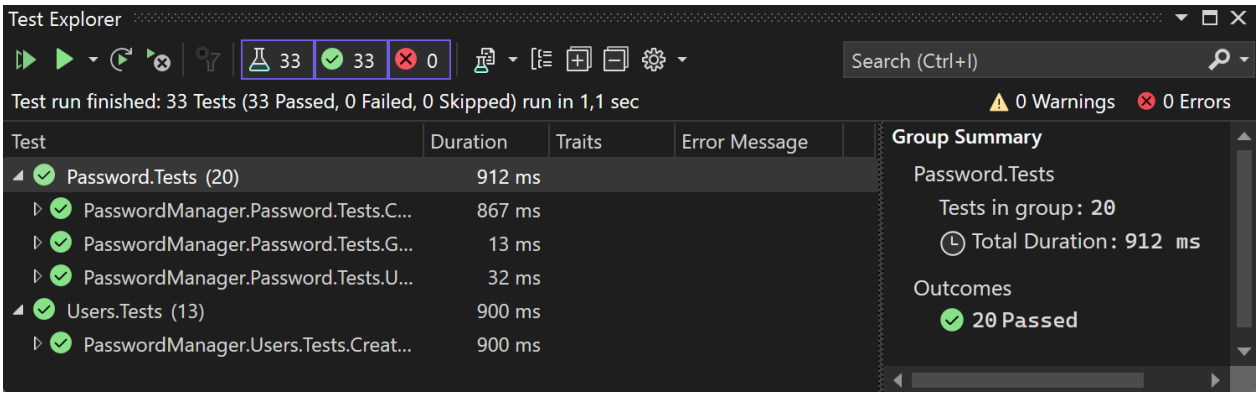
Eksempel på unit test af request create user password metoden

```
[Test]
| 0 references | Rasmus Kristensen, 1 day ago | 1 author, 1 change
public async Task RequestCreateUserPassword_ReturnsOperationInvalidStateResult_WhenUserIsNotFound()
{
    var userPasswordModelFixture = UserPasswordModelFixture.Builder().WithId(_userId).Build();
    var operationDetails = new OperationDetails(_createdBy);

    _userRepositoryMock.Setup(user => user.Get(It.IsAny<Guid>())).ReturnsAsync(null as UserModel);

    var requestCreateUserPasswordResult = await _createUserPasswordService.RequestCreateUserPassword(userPasswordModelFixture, operationDetails);

    Assert.That(requestCreateUserPasswordResult, Is.Not.Null);
    Assert.That(requestCreateUserPasswordResult.Status, Is.EqualTo(OperationResultStatus.InvalidOperationRequest));

    _operationServiceMock.Verify(operation => operation.QueueOperation(It.IsAny<Operation>()), Times.Never);
    _keyVaultComponentMock.Verify(password => password.CreateEncryptedPassword(It.IsAny<UserPasswordModel>(), It.IsAny<string>()), Times.Never);
    _busMock.Verify(bus => bus.Send(It.IsAny<CreateUserPasswordCommand>(), null), Times.Never);
}
```

Code coverage:

Umbraco_DESKTOP-GH6GIV5 2024-04-04 10_32_06.coverage	3306	810	1164	47
passwordmanager.users.tests.dll	916	33	165	16
users.worker.service.dll	91	106	55	2
passwordmanager.users.domain.dll	93	76	104	0
passwordmanager.users.testfixtures.dll	57	30	76	0
passwordmanager.password.tests.dll	1538	10	264	25
users.messages.dll	16	0	26	0
password.messages.dll	22	2	35	0
passwordmanager.password.domain.dll	130	100	125	0
passwordmanager.password.testfixtures.dll	46	12	63	0
password.worker.service.dll	188	196	117	4
passwordmanager.password.applicationservices.dll	152	152	88	0
passwordmanager.users.applicationservices.dll	57	93	46	0

Test explorer af vellykket test:



Frotend:

Unit tests:

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level	Lines:	197	243	81.1 %
Test:	lcov.info	Functions:	0	0	-
Date:	Mon Apr 8 00:04:52 2024				

Directory	Line Coverage		Functions	
models\blocs\auth_bloc\bloc	22.2 %	4 / 18	-	0 / 0
models\blocs\create_vault_value_bloc\bloc	62.5 %	30 / 48	-	0 / 0
models\blocs\vault_bloc\bloc	50.0 %	11 / 22	-	0 / 0
models\dto_models	100.0 %	18 / 18	-	0 / 0
services\backend_services\api_endpoints	100.0 %	9 / 9	-	0 / 0
services\backend_services\api_utilities	100.0 %	5 / 5	-	0 / 0
services\backend_services\password_api_service	100.0 %	36 / 36	-	0 / 0
services\http_executor	89.7 %	26 / 29	-	0 / 0
services\service_managers\password_service	100.0 %	58 / 58	-	0 / 0

Generated by: LCOV version 1.15.alpha0w

Bilag:

- https://miro.com/app/board/uXjVKYw4KmQ=/?share_link_id=325131230635
- https://excalidraw.com/#json=Zg1H7buOvCs7gNCmaeKmC,49z3iMIm6W7XMwhjRix6kw
- https://excalidraw.com/#room=7767be9ffa970b1b3cc7,NFiGWffKoSbmXz-D9QSb5Q
- https://miro.com/app/board/uXjVKXCF4rI=/
- https://excalidraw.com/#json=YnV-TX9XQif3DmHeSdh8n,A1UIHfS5st5IZpcBT2EHIA
- https://excalidraw.com/#json=1Pj0UwL1-UDGyGqZiej_I,LklxqqJ7lQiH_ZaMKVfmCw

Litteraturliste

- https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-messaging-overview
- https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure
- https://moodle.zbc.dk/course/view.php?id=33549