



{ Procesrapport }

{ P@\$\$wordmanageren }

Creators = { Daniel, Rasmus, Benjamin }

Titelblad:

Deltagere: Daniel Vuust, Rasmus Thougard og Benjamin Hoffmeyer

Projektnavn: LockHive

Dato: 2024-04-05

Skole: ZBC Ringsted

Vejleder: ?

Daniel Vuust

Rasmus Thougard

Benjamin Hoffmeyer

Indholdsfortegnelse:

Titelblad:	1
Indholdsfortegnelse:	2
Definitioner, akronymer og forkortelser:	4
Indledning:	4
Baggrund:	4
Problemformulering:	5
Projektplanlægning:	5
Arbejdsmetode:	5
Teknisk arbejdsmetode:	6
Devops:	6
Git:	6
Risicimatrix:	8
Risicimodel:	9
Milepæle:	9
Tidsplan:	10
Estimeret tidsplan: Estimering af projekt.xlsx	10
Realiseret tidsplan:	10
Konklusion for tidsplan:	10
Arbejdsfordeling:	11
Design:	11
Implementering:	11
Test:	11
Rapport:	11
Metodevalg og teknologi:	11
Valg af Flutter (hybrid app):	11
Firebase:	12
Pakker brugt i app'en:	12
Dependencies:	12
Dev dependencies:	13
Valg af arkitektur:	14
Backend:	14
Microservice	14
Repository pattern	14
REST Api	14
EF Core	14
Clean architecture	14
Domain driven design	15
Event driven design	15
Command driven design	15
Frontend:	16

Bloc pattern:	16
Provider pattern:	16
Beskrivelse af væsentlige elementer (opmærksomhedspunkter) fra produktrapporten	16
Projektdagbog/logbog:	17
Konklusion:	23
Litteraturliste:	23

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

Indledning:

I en verden, der bliver mere og mere digitaliseret, står sikkerheden og beskyttelse af vores personlige oplysninger centralt. Grundet nemme og genbrugte passwords, sætter vi os selv i stor fare for at få stjålet vores identitet og personlige oplysninger ved hackerangreb. Vi vil dermed i følgende dokument redegøre for, hvordan vi kan øge sikkerheden for individet på internettet.

Baggrund:

Selvom 91%¹ af os kender de risici, der er ved at genbruge passwords, genbruger 62%² af bruger stadig deres passwords på tværs af hjemmesider. Dette har konsekvensen, at der ved et simpelt databrud på en hjemmeside, kan gives en kaskade af skader, da uvedkommende aktører kan få adgang til flere konti med data og personlige oplysninger, end hvis aktørens adgang var begrænset til den oprindelige komprimerede konto.

Men hvad er grunden til at, selvom vi kender risikoen for at genbruge passwords, at vi stadig gør det? LastPass prøvede at svare på dette spørgsmål i 2020, hvor 42%³ sagde at det var vigtigere for dem at kunne huske deres passwords end at de var unikke. Dermed

¹ <https://www.darkreading.com/vulnerabilities-threats/password-reuse-abounds-new-survey-shows>

² <https://www.lastpass.com/-/media/3c627ed089e84bc39ca2bf6bf1d7cdec.pdf>

³ <https://www.lastpass.com/it/resources/ebook/psychology-of-passwords-2020>.

vil det være en oplagt mulighed for at sikre at vi ikke genbruger passwords at lave en password manager, som holder styr på alle ens passwords et samlet sted hvor brugerne kan have et overblik over alle ens konti og dermed ikke genbruge passwords.

Dog skaber dette et større krav på sikkerhed, da alle passwords gemmes et centraliseret sted og dermed vil et dataleak her give uvedkommende aktører mulighed for at modtage adskillige logins og data på alle der brugerer som ligger i vores service. Det vil så sige at vi rykker ansvaret, men ikke konsekvenserne, af et hackerangreb fra individet til password manageren.

Problemformulering:

Hvordan kan en sikker password manager, som følger visse specifikationer fra owasp.org. Sikre at bruger ikke genbruger deres passwords på tværs af flere platforme, og dermed sænke hacking-relaterede databrud?

Projektplanlægning:

Arbejdsmetode:

Vi har valgt at lægge os op af en agil arbejdsmetode, hvori vi trækker diverse elementer ind fra fx scrum.

Vores projekt er opdelt i 5 sprints som hver vare 1 arbejdsuge, de 4 første indbærer udarbejdningen af projektopgaven og sidste sprint er forberedelse til fremlæggelse og forsvar af svendeprøveprojektet.

Måden vi arbejder på er, at vi hver dag holder et standup møde, som er kendt fra Scrum, hvor vi fortæller om hvad vi laver, hvilke problemer vi har, samt hvad vi forventer at lave i løbet af dagen. Under standup kigger vi også på vores [risici](#) og vurdere dagligt vores trusler, som er markeret at de skal observeres. Samtidig har vi også et retrospektivt møde i slutningen af hvert sprint, også kendt fra scrum, hvor vi kigger tilbage på sprinten og har en dialog omkring hvad vi kan gøre bedre og laver vi en overordnet evaluering af projektet og risici.

For at sikre continuous development, i forhold til produktet, fokuserer vi først på at lave en MVP som opfylder vores krav fra kravspecifikation(se produktrapport). Hvor vi derefter vil have mulighed for at lave iterationer af produktet og tilføje funktioner som ikke er krav fra vores kravspecifikation. (Se [Milepæle](#) for opdelings af sprints)

Teknisk arbejdsmetode:

Devops:

For at styre vores projekt har vi gjort brug af Azure DevOps til at opdele opgaver samt planlægge sprint.

Vi har blandt andet implementeret en effektiv arbejdsgang ved brug af flere af DevOps's funktioner. Vi har gjort brug af Epics til at opdele opgaverne i mindre håndterbare opgaver. Det har også givet os mulighed for at planlægge hvilke opgaver der skal nås i de forskellige sprint og lave en plan under vores sprint planning.

☰ Prøve Svendeprøve Team ☆ 🔍

Backlog Analytics + New Work Item ⌚ View as Board 🗨 Column Options ...

Order	Work Item Type	Title	State	Effort	Value Area	Iteration Path	Tags
1	Product Backlog	Backend: As a user i would like to get a secure generated password	New		Business	Prøve Svendeprøve\Sprint 1	
	Task	Create endpoint to generate new password	Done			Prøve Svendeprøve\Sprint 1	
	Task	Create service to generate a password	Done			Prøve Svendeprøve\Sprint 1	
2	Product Backlog	Backend: As a user i would like to update my password	New		Business	Prøve Svendeprøve\Sprint 1	
	Task	Create endpoint to update a password	Done			Prøve Svendeprøve\Sprint 1	
	Task	Create service to update password	Done			Prøve Svendeprøve\Sprint 1	
	Task	Create worker handler to handle update command	Done			Prøve Svendeprøve\Sprint 1	
3	Product Backlog	Backend: As a user i want to delete a password	New		Business	Prøve Svendeprøve\Sprint 2	
	Task	Create service to delete password	Done			Prøve Svendeprøve\Sprint 2	
	Task	Create endpoint to delete password	Done			Prøve Svendeprøve\Sprint 2	
	Task	Create work handler to handle delete password command	Done			Prøve Svendeprøve\Sprint 2	
4	Product Backlog	Backend: As a user i would like to get all my stored passwords	New		Business	Prøve Svendeprøve\Sprint 1	
	Task	Backend: Create endpoint to get all stored passwords	Done			Prøve Svendeprøve\Sprint 1	
5	Product Backlog	Frontend: As a user I want to see password that I have stored	New		Business	Prøve Svendeprøve\Sprint 2	
	Task	API connection	To Do			Prøve Svendeprøve\Sprint 2	
	Task	UI to get stored password	Done			Prøve Svendeprøve\Sprint 2	
6	Product Backlog	Frontend: As a user I want to save password to vault	New		Business	Prøve Svendeprøve\Sprint 2	
	Task	Api connection	In Progress			Prøve Svendeprøve\Sprint 2	
	Task	UI to save new password	Done			Prøve Svendeprøve\Sprint 2	

Git:

Til udviklingen af vores produkt har vi gjort brug af git, der er hostet på Azure DevOps. Vi valgte at bruge Git for dens robuste versionsstyringssystem, som muliggør effektivt samarbejde blandt vores udviklingsteam. Ved at hoste vores repositionere direkte i DevOps har vi også haft muligheden for at lave en integrering mellem kodningen og projektstyring.

Firestore auth and ui changes to make it more generic

Completed 113 Daniel Vuust proposes to merge [feature/davu/FirebaseSignIn](#) into [main](#)

Overview Files Updates Commits

Daniel Vuust completed this pull request 11. mar. Cherry-pick Revert

Merged PR 13: Firestore auth and ui changes to make it more generic
87517846 Daniel Vuust 11. mar. at 10:49

























Show details

No merge conflicts
Last checked 11. mar.

Description

🔗 Add firestore auth to be able to sign up Done 🔗 Add Firestore auth to be able to login In Progress

Derudover har vi også oprettet en process til at tjekke vores kode ved brug af pull requests (PRs). Disse PR har gjort det muligt for os at sikre en ens kodestruktur og fange de fejl der blev overset af udvikler. Dog har vi valgt ikke at have kommentarfunktionen som en kernefunktion i vores arbejdsproces. Dette er valgt, da vi nemmere kan undgå misforståelser ved at tage det mundtligt og have en dialog omkring det. Det betyder dog også at vi mister noget dokumentation på nogle af de valg der er taget i forhold til koden, men dette er et valg vi har taget på baggrund af opgavens scope og deadlines for projektet.

Pull requests					New pull request
Mine Active Completed Abandoned					
Pull Request ID	Created by	Assigned to	Target branch		
 VaultKeys Context (Protect/Unprotect text) snitzelklau request 124 into  master			0	Completed Yesterday	
 Created key vault service Rasmus Kristensen request 123 into  master			0	Completed torsdag	
 Added delete password endpoint, service and worker snitzelklau request 119 into  master			0	Completed onsdag	
 51 - Get user passwords Rasmus Kristensen request 118 into  master			0	Completed onsdag	
 Added length parameter for password generation snitzelklau request 116 into  master			2	Completed tirsdag	
 #52 Migration of password table to include userId Rasmus Kristensen request 117 into  master			0	Completed tirsdag	
 49 Setup users table in database Rasmus Kristensen request 115 into  master			0	Completed tirsdag	
 #47 Created users service Rasmus Kristensen request 114 into  master			0	Completed tirsdag	

Risicimatrix:

Sandsynlighed(Y)	Risicimatrix:				
Sikkert (5)	5	10	15	20	25
Næsten sikkert (4)	4	8	12	16	20
Sandsynligt (3)	3	6	9	12	15
Muligt (2)	2	4	6	8	10
Usandsynligt (1)	1	2	3	4	5
Konsekvens(X)	Ubetydeligt(1)	Mindre(2)	Væsentligt (3)	Større(4)	Alvorligt (5)

Farvekoder og betydning:

Grøn	Grøn betyder at truslen er af mindre karakter og der ikke burde blive taget særlige handlinger eller diskussion .
Gul	Gul betyder at truslen skal diskuteres og plan for mitigering skal laves, samt mulig observation.
Orange	Orange betyder, at truslen har en større karakter, og at truslen diskuteres, mitigeres og observeres.
Rød	Rød betyder, at truslen har en alvorlig karakter og skal håndteres med det samme. (Projektet er i fare/ude af kontrol)

Risicimodel:

Risicimodel:								
Trussel	Konsekvens	Sandsynlighed	Risikotal	Handling / Mitigering	Konsekvens	Sandsynlighed	Mitigering	Observer
Sygdom	3	2	6	Overarbejde af resten af gruppen.	1	2	2	
Langtidssygemelding	4	2	8	Evaluerer ny ansvarsfordeling, samt overarbejde.	2	2	4	
Fejlbedømmelse af tidsestimering	4	3	12	Overarbejde.	3	2	6	✓
Kompleksitet af projektet er større end forventet	4	2	8	Overarbejde.	3	2	6	✓
For meget overarbejde og hjemmearbejde	4	2	8	Afgræns projektets omfang.	3	2	6	✓

Milepæle:

Vi har valgt at bruge milepæle for at sikre at vi når vores mål. Her er den overordnede struktur for vores mål udført i sprints:

- Sprint 1:
 - Færdiggørelse af første uddrag til kravspecifikation
 - Valg af teknologier og metoder samt start på rapporter og produkt
- Sprint 2:
 - Færdiggørelse af MVP
- Sprint 3:
 - Færdiggørelse af første uddrag til rapporterne
- Sprint 4:
 - Rettelser af produkt
 - Eventuelt rettelse til rapporter
 - Aflevering af projekt og rapporter
- Sprint 5:

- Færdiggørelse af præsentation

Tidsplan:

Estimeret tidsplan: X Estimering af projekt.xlsx

Emne:	BC	WC	mL	C	o^v	o^v %	Sansynlighed færdig	Standard afvigelse	Strategi => Planen
Backend: Password	32	80	48	51,2	9,6	0,20	67,2	31,25	Sansynlighed 95%
Backend: Payment card	32	80	48	51,2	9,6	0,20	67,2	31,25	
Backend: User Authentication	16	60	25	30,2	8,8	0,35	42,6	29,80	
Frontend: UI	40	80	50	54	8	0,16	66	18,52	
Frontend: Api connection	20	50	30	32	6	0,20	42	31,25	
Frontend: Worker	20	80	30	38	12	0,40	54	26,32	
Procesrapport	30	70	50	50	8	0,16	66	40,00	
Produktrapport	40	80	50	54	8	0,16	66	18,52	
Projektmanagement	20	40	30	30	4	0,13	38	33,33	
								509 timer	

Realiseret tidsplan:

Emne:	BC	WC	mL	C	o^v	o^v %	Sansynlighed færdig	Standard afvigelse	Strategi => Planen	Brugt tid
Backend: Password	32	80	48	51,2	9,6	0,20	67,2	31,25	Sansynlighed 95%	42
Backend: User	-	-	-	-	-	-	-	-	-	100
Backend: Kryptering	-	-	-	-	-	-	-	-	-	50
Backend: Payment card	32	80	48	51,2	9,6	0,20	67,2	31,25		20
Backend: User Authentication Firebase	16	60	25	30,2	8,8	0,35	42,6	29,80		15
Frontend: UI	40	80	50	54	8	0,16	66	18,52		50
Frontend: Api connection	20	50	30	32	6	0,20	42	31,25		20
Frontend: Worker	20	80	30	38	12	0,40	54	26,32		14
Procesrapport	30	70	50	50	8	0,16	66	40,00		60
Produktrapport	40	80	50	54	8	0,16	66	18,52		70
Projektmanagement	20	40	30	30	4	0,13	38	33,33		10
								509 timer		
										451 timer

Konklusion for tidsplan:

Ifølge vores realiseret tidsplan har vi haft et overskud på 58 timer, hvilket antyder at vi har overholdt vores tidsplan. Dette er dog ikke helt tilfældet, da vi har været mødt af en række udfordringer såsom større kompleksitet af projektet, herunder specielt User og Kryptering. Derefter har vi været ramt af sygdom og været dårlige til at måle vores realiseret tid for projektet.

Det kan derfor tænkes, at overskuddet af timerne stammer fra afgrænsning af projektet, samt vores unøjagtige måling af vores tidsforbrug. Derudover kan det tænkes, at vi har sat vores arbejdsvilje og motivation for projektet for højt, som kan have resulteret i et forhøjet estimat af tidsplanen.

Arbejdsfordeling:

Herunder er en generel arbejdsfordeling til projektet og det ansvar der er blevet givet til hver især. Dette betyder dog ikke at denne beskrivelse er visende for hvem der har lavet hvad, vi har arbejdet på tværs og hjulpet til hvor der har været manglet ressourcer, men at vi har fået ansvarsområder, som vi har været ansvarlige for at sikre blev udført.

Design:

Rasmus: Ansvarlig for systemarkitektur og database design.

Daniel: Ansvarlig for design af UI/UX.

Benjamin: Ansvarlig for sikkerhed

Implementering:

Rasmus: Ansvarlig for backend-udvikling, user- og password-servicen.

Benjamin: Ansvarlig for backend-udvikling af keyVault servicen

Daniel: Ansvarlig for frontend-udvikling.

Rasmus: Ansvarlig for hosting af database, server og service bus i Azure.

Test:

Benjamin & Rasmus: Ansvarlig for automatisk test af backend.

Daniel: Ansvarlig for automatisk test af frontend.

Rapport:

Benjamin, Rasmus og Daniel: Procesrapport og produktrapport

Metodevalg og teknologi:

Valg af Flutter (hybrid app):

Til at udvikle vores app har vi gjort brug af Flutter. Flutter bygger på ideologien om hybrid udvikling, der gør det muligt at have en kodebase til flere platforme, herunder bla. Android, iOS, Windows, Linux og web. Dette gør det muligt at have en kodebase til vores app, som så kan køres på både Android og iOS.

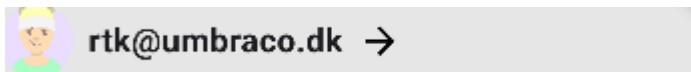
Selvom Flutter ikke er den eneste hybrid udviklingsform, er valget til at bruge Flutter taget på baggrund af vores gruppes kompetencer.

Firestore:

Vi har valgt at gøre brug af Firestore til at styre dele af vores projekt. Firestore gør det muligt at tilføje en eller flere elementer såsom authentication, dashboard for fejl, A-B testing, Firestore database og mere. Vi har indtil videre gjort brug af authentication, som gør det muligt for os at have et login flow som styres af Firestore.

Robohash.org⁴

Robohash er blevet brugt til at vise unikke profilbilleder i appen:



Pakker brugt i app'en:

Følgende er en liste over brugte pakker til app'en med kort beskrivelse over hvad de gør:

Dependencies:

`flutter_bloc`:⁵

Flutter Bloc bruges til at holde state i app'en og gør det muligt at adskille UI fra logik og state.

`logger`:⁶

Logger bruges til at logge beskeder som kan defineres til at være på forskellige niveauer og vi kan også filtrere på hvilket niveau vi vil se log beskeder på. Pakken giver os mulighed for at definere og filtrere på niveauerne trace, debug, info, warning, error og fatal.

`animated_background`:⁷

Animated background gør det muligt for os at have en animeret baggrund i vores app, som vi bruger på alle vores sider.

`firebase_auth`:⁸

Firestore Auth holder styr på alle login informationer, hjælper os med at holde styr på hvilken bruger der er logget ind på vores app og giver os mulighed for at oprette tidsbegrænset tokens som vi bruger til at authenticate til backend'en.

⁴ <https://robohash.org/>

⁵ https://pub.dev/packages/flutter_bloc

⁶ <https://pub.dev/packages/logger>

⁷ https://pub.dev/packages/animated_background

⁸ https://pub.dev/packages/firebase_auth

firebase_ui_auth:⁹

Firebase ui auth danner UI til login, sign up, m.m og styrer logikken bag. Se produktoprapporten for mere.

firebase_core:¹⁰

Firebase core er den generelle pakke der bruges når der er Firebase elementer i ens app.

settings_ui:¹¹

Settings UI bruges til at danne vores bruger UI.

Dev dependencies:

mocktail:¹²

Mocktail bruges til at mock'e klasser til testing.

flutter_test:¹³

Flutter test er en standard pakke som giver muligheden om at skrive tests til app'en.

flutter_lints:¹⁴

Flutter lints er en standard pakke som sikre af koden der bliver skrevet er god kodning praksis.

build_runner:¹⁵

Build runner er en pakke som giver udviklere mulighed for at definere en klasse med forskellige attributter hvorefter Build runner oprettet metoder til at konvertere fra og til json.

json_annotation:¹⁶

Json annotation er en forlængelse af json converting som giver mere fleksible muligheder ved klasse definitioner.

test:¹⁷

Test er en pakke som giver mulighed for at skrive tests.

⁹ https://pub.dev/packages/firebase_ui_auth

¹⁰ https://pub.dev/packages/firebase_core

¹¹ https://pub.dev/packages/settings_ui

¹² <https://pub.dev/packages/mocktail>

¹³ https://api.flutter.dev/flutter/flutter_test/flutter_test-library.html

¹⁴ https://pub.dev/packages/flutter_lints

¹⁵ https://pub.dev/packages/build_runner

¹⁶ https://pub.dev/packages/json_annotation

¹⁷ <https://pub.dev/packages/test>

firebase_auth_mock¹⁸

Firebase auth mocks tillader at lave tests op imod Firebase authentication så det er muligt at teste authenticationen.

Valg af arkitektur:

Backend:

Microservice

Vi har valgt at bruge microservice for at opdele password manager i forskellige services. Grunden til vi har valgt microservice er hovedsagelig for at få det rene snit mellem domænerne i password manageren og skaber en stor fleksibilitet, da vi kan arbejde på hver sin service uden at ændre/ødelægge kode for hinanden, der eventuel kan skabe en merge conflict, som vi skal bruge til på at løse.

Repository pattern

Vi har valgt at bruge repository pattern, fordi en af de primær fordele ved dette mønster er, at det adskiller business logic laget fra data access-laget. Udover det slipper vi fra duplikeret kode, da vi har et interface, som har nogle af elementerne fra CRUD.

REST Api

Fordi vi bruger microservice, har vi ekstra stort behov for at udnytte Application Programming Interface(api), som tillader os at kommunikere og udveksle data på kryds og tværs af vores services og systemer. REST definerer funktionerne i CRUD.

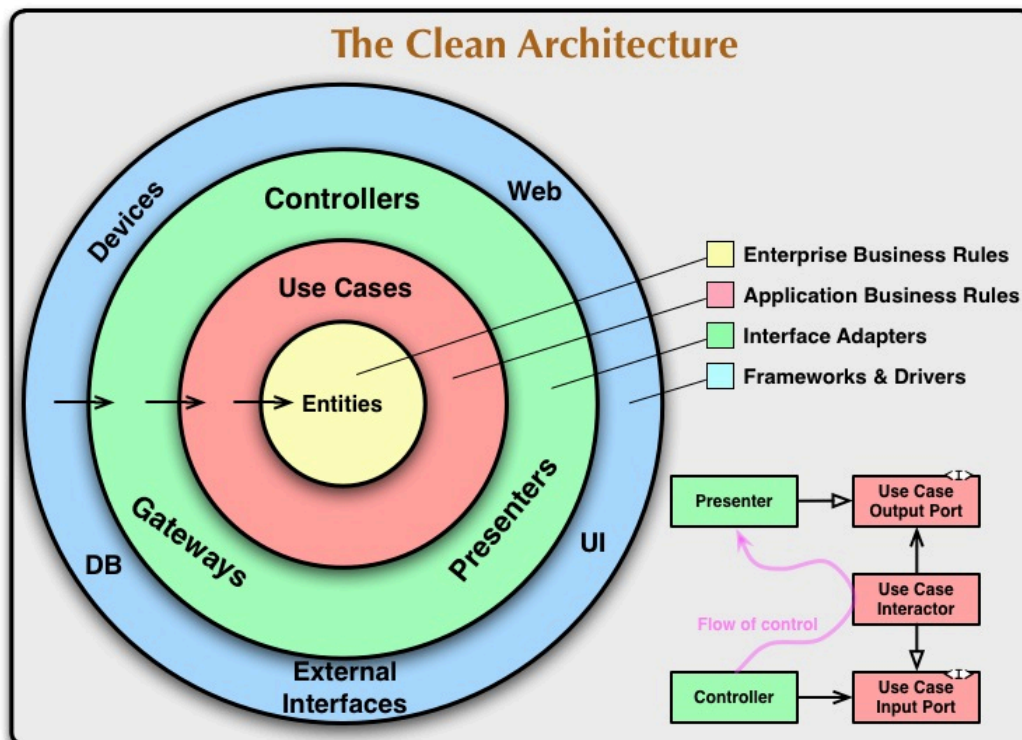
EF Core

Vi har valgt at brug EF Core, fordi det har åbnet muligheden for at udvikle en database ved brug af .NET objekter. Udover det er det ekstremt nemt at bruge, og man kan etablere en database og tabeller på ingen tid.

Clean architecture

Vi har valgt clean architecture, fordi det gør at vi adskiller business logic, brugergrænsefladen og data access. Den adskilles er med til at vi nemt kan udskifte hvilken type af database eller hvilken brugergrænseflade i forhold til hvis det ikke var adskilt i disse lag.

¹⁸ https://pub.dev/packages/firebase_auth_mock



Domain driven design

Fordi vi har valgt at bruge microservice og clean architecture ligger Domain driven design lige til højrebænet, da microservice og clean architecture bygger på at servicen skal være afhængig af domain. Domain-driven design bygger også på forståelsen mellem udviklere og domæneeksperter i forretningsområdet.

Event driven design

Grunden til vi har valgt event driven design er, fordi det kan tage tid at oprette f.eks et password, som skal krypteres osv. Derfor har vi valgt at bruge event driven design til at publicere et event, når f.eks et password er oprettet, så brugeren ikke skal sidde og vente i den tid det tager at oprette et password. Brugeren vil blive informeret om at password er blevet oprettet, når det er sket. Udover det, hvis skalering af password manager sker, kan vi starte et stykke arbejde i et andet system baseret på det event.

Command driven design

Vi har valgt at bruge Command driven design for adskillelse af ansvarsområder. Den måde vi vil bruge det på er, at der sker en forespørgsel på f.eks oprettelse af password. I forespørgelse tjekker vi for xyz, hvis alt er okay, sender vi en command, som en handler samler op og begynder arbejdet med oprettelsen af password.

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

Frontend:

Bloc pattern:

Til udviklingen af fronten har vi gjort brug af bloc pattern. Dette er valgt grundet muligheden for at adskille UI fra state og logik, hvilket også giver en mere struktureret og vedligeholdelsesvenlig kodebase.

Provider pattern:

I sammenhæng med bloc pattern er der blevet brugt provider pattern til at give widget træet adgang til forskellige blocs.

Væsentlige elementer fra produktrapporten

Vi har valgt at gøre brug af microservice i backenden, som skal være med til at give os et stort overblik, og nemt kan udvides. Det har givet os overblikket, men det har skabt nogle problemer med udviklingen, fordi vi har 30 projekter, som skal køre hele løsningen, dette har gjort det kompliceret at starte være password manager og tage x antal tid hver gang, vi skal manuel teste endpoints osv.

Vi valgte at gøre brug af operation for at holde styr på status over en operation. Dette har været med til at gøre det mere kompliceret end højst nødvendigt.

Vi valgte at bruge Azure Devops, som et værktøj til at have et overblik over opgaver vi skal løse for at lave en feature. Vi brugte det meget i starten, hvor vi planlagde product backlog items med opgaver for at lave en feature, men efterhånden at projektet udvidede sig, blev det uoverskueligt.

Vi valgte at strukturere backend med en api og worker. Api'en er en forespørgsel på at starte en proces med en forespørgslen, hvis forespørgslen var okay, sendte vi en kommando, der starter worker servicen. Worker servicen begynder med den reelle arbejdsproces, såsom at oprette et password. Fordi vi har valgt at gøre det på denne måde, har medført, at vi skal skrive mere kode og øge kompleksiteten af projektet mere end det vi havde regnet med.

Vi brugte en del tid på at meningsudveksle hvordan vi skulle kryptere og dekryptere password. Vi begyndte at programmere det med udgangspunkt i vores forståelse af hvordan opgaven skal løses. Den løsning var ikke korrekt, og derfor begyndte vi igen at meningsudveksle hvordan vi kunne løse opgaven. Vi kom frem til at lave et rigt billede og derefter et SD diagram, så vi havde dokumentation og huske hvad vi blev enig om. Dette medførte en brugbar løsning.

Vi valgte at bruge Azure, hvor vi fik hosted vores SQL server og database, det er utroligt brugbart, fordi vi kan connecte til databasen uanset om vi sidder på skolen, derhjemme eller et helt tredje sted. Det har skabt en større frihed, at vi kan arbejde på projektet derhjemme efter skole.

Brugen af Firebase har givet os mulighed for at afgrænse os fra datasikkerheden ved user login, og giver ansvaret for det over til Firebase. Samt har muligheden for at oprette en automatisk UI med pakken firebase_ui_auth givet ekstremt meget værdi i forhold til hvor meget tid der blev brugt på det. Samt har Firebase også gjort det muligt for os at authenticate i backenden på en sikker måde. Samtidig har brugen af Robohash givet os muligheden for at personliggøre vores app til brugeren på en nem måde.

Det bloc pattern vi har brugt for skabt en skapt linje mellem UI og state/logik og den specielle måde vi har gjort det på har sikret bla, at der er muligt at teste alle event og begrænset uvidende fejl da alle ændringer skal defineres explicit.

Projektdagbog/logbog:

**Hver dag har vi, som beskrevet i [Projektplanlægning](#), holdt standup, dette er undladt fra logbogen.*

**Under standup har vi hver dag vurderet de risici der skal observeres, og dette er undladt i logbogen hvis risikoen ikke er realiseret.*

Uge 1:	
2024-03-04	
Daniel:	<ul style="list-style-type: none">- Vi har læst og forstået casen.- Vi har lagt en plan for hvilke mål vi gerne vil opnå med svendeprøven.- Vi har lavet en plan c, b og a. I planen indgår der hvilke mål for svendeprøven som vi vil prioritere.- Vi har snakket om arkitekturen og lavet overordnet ansvarsfordeling.- Vi har oprettet procesrapport og produktrapport samt valgt elementer som skal indgå.- Vi har snakket om hvor vi vil holde styr på vores arbejde(dev Azure).
Rasmus:	
Benjamin:	
2024-03-05	
Fælles:	<ul style="list-style-type: none">- Udviklede rigt billede samt lavet epic, features...- Sparede sammen angående den teknologiske tilgang til både app'en og backenden

Daniel:	<ul style="list-style-type: none"> - Startede udviklingen af kerneelementer til app'en som ikke er afhængig af fremtidige dialoger i Flutter - Vurderede og skrev først uddrag til baggrund for opgaven
Rasmus:	<ul style="list-style-type: none"> - Oprettet password manager repository for backend. - Oprettet Sql server på azure - Oprettet Sql database på Azure - Oprettet servicebus på Azure - Lavet endpoint til at hente password
Benjamin:	<ul style="list-style-type: none"> - Lavet GetPassword og GetAllPasswords endpoints samt logik hertil - Skrevet i procesrapport herunder kravspecifikation.
2024-03-06	
Fælles:	<ul style="list-style-type: none"> - Havde samlet en opfølgende dialog omkring valg af teknologi og database tilgang. - Lavede samlet en tidsplan for projektet - Udforskede app tema til optimal bruger glæde - Undersøgt hvad OWASP er
Daniel:	<ul style="list-style-type: none"> - Udviklede overordnet struktur over appen og lavede diagram over widgets
Rasmus:	<ul style="list-style-type: none"> - Review create password endpoint
Benjamin:	<ul style="list-style-type: none"> - Lavet CreatePasswordEndpoint og logikken hertil. - Arbejdet på produktrapport. - Påbegyndt arbejde på UpdatePasswordEndpoint.
2024-03-07	
Fælles:	
Daniel:	<ul style="list-style-type: none"> - Udviklede UI til app
Rasmus:	<ul style="list-style-type: none"> - Review create password pull request - Lavet endpoint til at opdater password - Lavet service til at opdater password - Lavet handler til at opdater password
Benjamin:	<ul style="list-style-type: none"> - Arbejdet på dokumentation
2024-03-08	
Fælles:	<ul style="list-style-type: none"> - Arbejdet på fully dressed use cases - Vejledning - Revurderet risici matrix - Retrospektiv

Daniel:	<ul style="list-style-type: none"> - Udviklede diagram over nye ui ændringer - Udersøgte autofill-service til Android - Implementerede Firebase authentication i app'en
Rasmus:	<ul style="list-style-type: none"> - Klargjort paymentcard service
Benjamin:	<ul style="list-style-type: none"> - Lavet GeneratePassword endpoint samt logik og tests hertil

Uge 2:	
2024-03-11	
Fælles:	<ul style="list-style-type: none"> - Udarbejdet fully dressed use cases - Meningsudvekslet arkitekturen af vores password manager - Planlægning af sprint 2
Daniel	<ul style="list-style-type: none"> - Ændrede frontend design og flow
Rasmus:	
Benjamin:	<ul style="list-style-type: none"> - Arbejdet på kryptering af passwords
2024-03-12	
Fælles:	-
Daniel:	<ul style="list-style-type: none"> - Oprettede bruger siden i app'en
Rasmus:	<ul style="list-style-type: none"> - Opsætning af user servicen
Benjamin:	<ul style="list-style-type: none"> - Arbejdet på kryptering af passwords
2024-03-13	
Fælles:	-
Daniel:	<ul style="list-style-type: none"> - Oprettede udkast til hvordan Firebase brugen authenticates i backend - Rettede fejl i app'en
Rasmus:	<ul style="list-style-type: none"> - Udarbejdet endpoint to at hente og create user password
Benjamin:	<ul style="list-style-type: none"> - Arbejdet på krypterings context (KeyVault) herunder: SecurityKey database og DAL.
2024-03-14	

Fælles:	- Lavet diagram over overordnet arkitektur
Daniel:	- Hjælp med backend udvikling og tilføjede firebase auth til backend
Rasmus:	-
Benjamin:	- Arbejdet på krypterings context (KeyVault) herunder: Protect, Unprotect, Reprotect og delete logik.
2024-03-15	
Fælles:	- Retrospektiv
Daniel:	- Testede og skrev automatiske test til frontend
Rasmus:	- Fandt en måde at genere api client, så vores services kan snakke med hinanden.
Benjamin:	- Lavet krypterings context (KeyVault) herunder: Protect, Unprotect, Reprotect og Delete endpoints.

Uge 3:	
2024-03-18	
Fælles:	- Meningsudvekslet om struktur i user-, keyVault- og password-service.
Daniel:	- Skrev rapport samt ændrede app'en til at passe sammen med backend'en
Rasmus:	- Implementeret kryptering af keyVault service i user service. - Skrevet om tekniske produktdokumentation i produkt rapporten.
Benjamin:	- Påbegyndte arbejde på en revideret og forsimplet version af KeyVault.
2024-03-19	
Fælles:	- Ved vores normale vurdering af risici har vi vurderet at vores tidsestimering af projektet har været for optimistisk, og hermed mitigere vi med mere overarbejde. - Vejledning af produkt og procesrapport - Meningsudveksling af hvordan vi skal krypter og decrypter passwords på
Daniel:	-

Rasmus:	- Udvikling af user service for at oprette passwords
Benjamin:	- Færdiggjort revidering af KeyVault. - Påbegyndt arbejde på PaymentCard.
2024-03-20	
Fælles:	-
Daniel:	- Lavede app'en mere generisk med mulighed for at forlænge med andre typer af gemte values
Rasmus:	- Lavet rigt billede af flow for oprettelse af password, som bliver krypteret og hvordan vi henter encrypted password. - Skrevet valg af arkitektur for backend.
Benjamin:	- Arbejdet på PaymentCard model, database, repositories samt applikations logik
2024-03-21	
Fælles:	-
Daniel:	- Opdaterede app i forhold til backend - Skrev diagrammer til app
Rasmus:	- Ændret unprotect + protect endpoint til at returner krypteret password + ikke krypteret password med password id - Samlet hele flowet for at i user servicen til at oprette et password der bliver krypteret og flowet for at hente et password ud fra user og password url
Benjamin:	- Arbejdet på PaymentCard Workerservice og create endpoint så man kan oprette et payment card.
2024-03-22	
Fælles:	-
Daniel:	- Udforskede muligheden for at bruge Autofill services i Android - Opnåede 69% unit test coverage
Rasmus:	- Skrevet teknisk produkt dokumentation
Benjamin:	- Arbejdet på PaymentCard CRUD endpoints og logik hertil.

Uge 4:

2024-04-02	
Fælles:	-
Daniel:	- Udforskede implementeringen af autofill i Android
Rasmus:	- Sygdom
Benjamin:	- Arbejdet på Password og PaymentCard context og fixet problemer. - Arbejdet på User context
2024-04-03	
Fælles:	- Ved vores normale risici vurdering har vi vurderet at vi skal afgrænse projektets omfang grundet for meget overarbejde til at sikre at vi er færdige ved deadline. Dermed afgrænser vi autofill service i appen og integration test i backenden.
Daniel:	- Skrev rapport i forhold i app'en
Rasmus:	- Sygdom
Benjamin:	- Arbejdet på user context og færdiggjort UserPassword endpoints og logik
2024-04-04	
Fælles:	- Skrevet konklusion på procesrapport
Daniel:	- Oprettede de sidste diagrammer til app'en
Rasmus:	- Skrevet unit test for create user password i user servicen og create password i password servicen. - Lavet klasse diagram over create user password
Benjamin:	- Lavet SD diagrammer for Password CRUD flows. - Lavet flow chart over
2024-04-05	
Fælles:	- Lyttet på fremlæggelser for at få inspiration til hvordan vi kan gøre det. - Small touches på rapporterne
Daniel:	- Oprettede de sidste meningsfyldte unit tests samt fejlrettelser
Rasmus:	- Skrevet væsentlige elementer, som vi har oplevet ved udvikling af backend
Benjamin:	- Sygdom (men lavet database diagram og rettet SD diagrammer)

Konklusion:

Under vores proces har der været bump på vejen. Det første bump var, at vi gerne ville igang med at programmere så hurtigt som muligt uden at tænke os grundigt om, og lave diagrammer over hvordan struktur skulle være. Dette medførte en masse snak og misforståelse.

Derudover har vi arbejdet alt for selvstændigt med de forskellige use cases, hvor vi ikke inddrager hinanden i processen. Vi valgte at arbejde hjemmefra i tre dage, hvor vi ikke havde meget standup eller lignede kommunikation om processen med arbejde.

Selvom vi også have planlagt vores projekt i sprints har der været tider hvor vi ikke har haft en logisk næste opgave, eksempler på dette var til tidspunkter hvor frontend'en var foran backenden og dermed skulle udviklingen af frontenden vente uden andre oplagte opgaver.

Til sidst fandt vi også ud af, at vi havde sat vores forventning til hvad vi kunne opnå i perioden alt for højt og dermed ramte vi en af vores risici fra vores risici matriks og måtte dermed sænke forventningerne.

Ved at implementere funktionalitet, som gør det nemt og intuitivt for brugeren at autogenerere password, kan brugeren undgå fristelsen til at genbruge password, og dermed styrke deres digitale sikkerhed. Når brugeren gemmer sit password, sørger vi for at kryptere. Uanset om en hacker får fat i det krypterede password, kan den hackeren ikke bruge password til noget. Da vi både følger OWASP bedste praksis og krypterer password, bidrager vores password manager til at sænke risikoen for hacking-relaterede databrud betydeligt, og øge den overordnede sikkerhed på tværs af platforme. Dog vil en logisk fortsættelse på dette være at tilføje autofyld funktionalitet som supporteres af både iOS og Android, men vi blev grundet tidspres nød til at skære dette fra.

Litteraturliste:

- <https://lp-cdn.lastpass.com/lporcamedia/document-library/lastpass/pdf/en/LastPass-Enterprise-The-Password-Expose-Ebook-v2.pdf>
- <https://projektledelsekursus.dk/successiv-kalkulation-estimering/>
- https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/assets/docs/OWASP_SCP_Quick_Reference_Guide_v21.pdf
- <https://excalidraw.com/#room=7767be9ffa970b1b3cc7,NFiGWffKoSbmXz-D9QSb5Q>
- Event driven design

- <https://medium.com/@seetharamugn/the-complete-guide-to-event-driven-architecture-b25226594227>
- Domain driven design
<https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/best-practice-an-introduction-to-domain-driven-design>
- EF core
<https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx>
- API
<https://aws.amazon.com/what-is/api/#:~:text=API%20stands%20for%20Application%20Programming,other%20using%20requests%20and%20responses.>