

Kerntaak P1 - K1

P1- K1 W1: onderhoudt een applicatie

P1- K1 W1: Beheert gegevens

Tim Bouma

20-05-2021

### 

### 

[**Inleiding**](#_8wbwkh1403cq) **2**

[**P1-K1-W1 Onderhoud een applicatie**](#_ka0bhkmgxtiy) **3**

[De kandidaat analyseert en categoriseert incidentmeldingen (4x), en vraagt hierbij indien nodig om meer informatie.](#_8etir6fo9fnf) 3

[De kandidaat legt de mogelijke oplossingen voor de incidenten vast en toetst of de oplossingen uitgevoerd kunnen worden (rekening houden met eerder gemaakte afspraken of contract, de werking van het product en beschikbare middelen.](#_77zju08blz5) 4

[**P1-K1-W2 Beheert gegevens**](#_83mot2x634lu) **11**

[De kandidaat controleert gegevens van de applicatie op juistheid, volledigheid en bruikbaarheid.](#_dd7pdi4doz7x) 11

[De kandidaat documenteert en archiveert gegevens van de applicatie in een kwaliteitshandboek.](#_ujtg26mbgtu) 13

### Inleiding

**Project**

Als stage opdracht hebben besloten om een applicatie te maken in de programmeertaal Dart waarbij we het framework Flutter gebruiken om de applicatie in elkaar te zetten. Om gegevens op te slaan kiezen we er voor om firebase firestore te gebruiken als database. Als het lukt om deze applicatie opzetten via de voorgeschreven flutter ontwikkelmethodes krijg je een goed beeld van hoe je een cross-platform applicatie moet ontwikkelen.

Voordat ik stage ben gaan lopen heb ik als barista gewerkt bij de CoffeeCompany. Bij de coffeecompany ben ik veel bezig geweest met het zetten en branden van koffie. Op die manier zijn we met het idee gekomen om een koffie applicatie te ontwikkelen. In de applicatie kunnen gebruiker inloggen en registreren met Firebase authenticatie. Daarna kan een gebruiker zijn koffie voorkeur te noteren. Zo weet de stagiaire van elke werknemer hoe hij zijn koffie wilt. Het halen van koffie is namelijk een van de taken van een stagiair.

Het documenteren van het ontwikkelprocessen wordt gedaan met Git. Hier kan je de code op committen. Elke keer als de code is aangepast kan je hem opnieuw committen naar je Github repository. Hierdoor krijg je een goed overzicht van elke verandering in de code en heb je gelijk een versiebeheer. Daarnaast is ook alle documentatie van de applicatie beschikbaar in GitHub project.

**Verslag**

Het verslag P1-K1 onderhouden van een applicatie begint bij het ontvangen van een mail van de opdrachtgever. In deze mail worden verschillende incidentmeldingen genoemd. De volgende stap in dit verslag is het analyseren van de incidentmeldingen.

Na het analyseren worden er naar oplossingen gezocht voor de incidenten en uitgewerkt in dit verslag. Zo nodig vraagt de ontwikkelaar extra informatie aan de opdrachtgever. Als de oplossingen overeenkomen met de eisen die aan het project zijn gesteld kunnen ze worden doorgevoerd in de code.

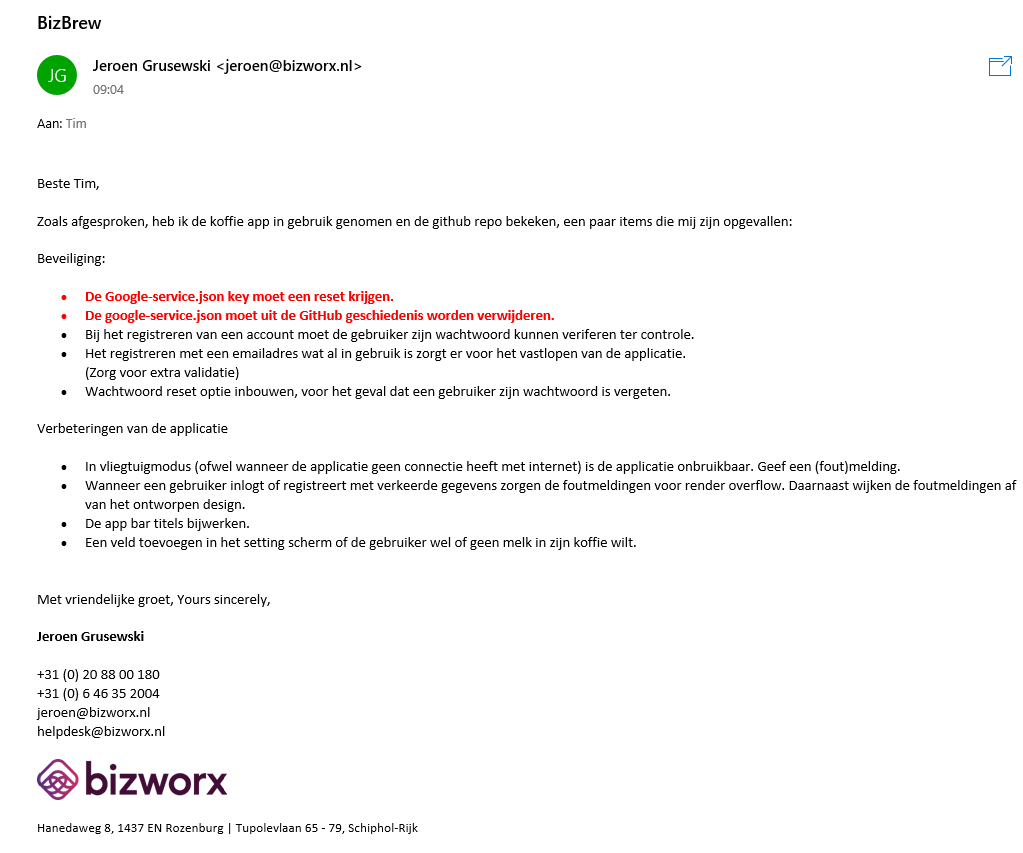
Aan de hand van afbeeldingen en het archiveren van de code word uitgelegd hoe de oplossingen voor de incidentmelding zijn verwerkt in de code. De opdrachtgever word van de voortgang van het onderhoud op de hoogte gehouden. Dit zal plaatsvinden via de mail of Microsoft Teams.

In het tweede deel van verslag P1-K1 word nog een extra gecontroleerd op de juistheid van de documentatie. Ook worden alle hulpmiddel verzamelt en verwerkt in een GitHub project. Het Github project word aangewezen als het kwaliteitshandboek. Onderdelen die je zou verwachten in een kwaliteitshandboek zijn in dit geval verwerkt in een README.md.

### P1-K1-W1 Onderhoud een applicatie

#### **De kandidaat analyseert en categoriseert incidentmeldingen (4x), en vraagt hierbij indien nodig om meer informatie.**

Mail opdrachtgever:



In overleg met de opdrachtgever besloten om alle incidentmelding te verwerken. De omvang van de incidentmeldingen zijn niet van hele grote omvang. Het verwerken van alle incidentmeldingen zal daarom niet extra veel tijd innemen. Doordat alle incidenten te verwerken worden ze niet gecategoriseerd. Wel is gecontroleerd of ze in lijn zijn met eerder gemaakt afspraken die zijn beschreven in het *Plan van Aanpak* en *Functioneel Ontwerp.* Onderdelen die niet overeen zijn gekomen met de documentatie zijn besproken met de opdrachtgever en met zijn goedkeuring doorgevoerd.

In de mail van de opdrachtgever staan de volgende incidentmeldingen:

Beveiliging

* De Google-service.json key moet een reset krijgen.
* De google-service.json moet uit de GitHub geschiedenis worden verwijderen.
* Bij het registreren van een account moet de gebruiker zijn wachtwoord kunnen verifiëren ter controle.
* Het registreren met een emailadres wat al in gebruik is zorgt voor het vastlopen van de applicatie. (Zorg voor extra validatie)
* Wachtwoord reset optie inbouwen, voor het geval dat een gebruiker zijn wachtwoord is vergeten.

Verbeteringen van de applicatie

* In vliegtuigmodus (ofwel wanneer de applicatie geen connectie heeft met internet) is de applicatie onbruikbaar. Geef een foutmelding.
* Wanneer een gebruiker inlogt of registreert met verkeerde gegevens zorgen de foutmeldingen voor render overflow. Daarnaast wijken de foutmeldingen af van het ontworpen design.
* De app bar titels bijwerken.
* Een veld toevoegen in het setting scherm of de gebruiker wel of geen melk in zijn koffie wilt.

#### **De kandidaat legt de mogelijke oplossingen voor de incidenten vast en toetst of de oplossingen uitgevoerd kunnen worden (rekening houden met eerder gemaakte afspraken of contract, de werking van het product en beschikbare middelen.**

**Google service key**

Het resetten van de Google Service.json Api key kan met de google cloud console worden geregeld. In de google cloud console kan je het Biz Brew project selecteren. In het dashboard APIs & Services kan je de credentials van je API key opnieuw genereren. In de google-services.json file moet wel de oude API key vervangen worden door de nieuwe APi key.

Het verwijderen van een GitHub geschiedenis is in een aantal stappen gedaan:

Verwijder geschiedenis uit the git repository

* cd project repository
* rm -rf .git

Initieert een nieuwe repository

* git init
* git add .
* git commit -m “removed history, due to sensitive date”

push to remote

* git remote add origin url\_of\_repository.git
* git push -u --force origin main

Na het doorvoeren van de aanpassingen wordt de code opnieuw gecommit naar GitHub. Om te voorkomen dat de google services api key mee wordt gecommit, voegen we dit bestand toe aan de .gitignore file. De .gitignore file beschrijft alle bestanden die niet toegevoegd mogen worden aan een git commit.

**Registratie**

Wanneer een gebruiker registreert met een e-mailadres wat al in gebruik is loopt de applicatie vast. Om het vastlopen van de applicatie te voorkomen moet een foutmelding worden weergegeven aan de gebruiker. Firebase geeft nu achter grond een exceptie dat het e-mailadres in gebruik is. Door deze exceptie op te vangen en weer te geven als een foutmelding aan de gebruiker zal de applicatie niet vastlopen en weet de gebruiker gelijk ook wat er mis is gegaan tijdens het registreren.

Future registerEmailPassword(String email, String password) async {

try {

UserCredential userCredential =

await FirebaseAuth.instance.createUserWithEmailAndPassword(

email: email,

password: password,

);

return true;

} on FirebaseAuthException catch (e) {

return e.message;

}

}

Center(child: Text(exception.isNotEmpty ? exception : ""),),

Bij het registreren van een account moet de gebruiker zijn wachtwoord kunnen verifiëren. Om de gebruiker zijn wachtwoord te kunnen controleren moet er een extra wachtwoord veld komen. Het toegevoegde wachtwoordveld krijg als validator een if-statement die vraagt of beide wachtwoord velden overeenkomen. Daarnaast krijgt het toegevoegde wachtwoordveld een knop die beide wachtwoordvelden zichtbaar maken voor de gebruiker. Het toevoegen van deze oplossingen zorgt er voor dat wachtwoorden zowel door het systeem en gebruiker worden geverifieerd.

validator: (val) => valFunc.validatePassword(password, repeatPassword)

? "Please repeat the same password"

: null,

bool validatePassword(String password, String repeatPassword) {

return password == repeatPassword ? false : true;

}

void \_togglePasswordView() {

setState(() {

hidden = !hidden;

});

}

**Sign In**

TextButton(

onPressed: () async {

if (email.isEmpty) {

setState(() =>exception = "Enter your email address");

} else {

dynamic result =await AuthService().sendPasswordResetEmail(email);

if (result == true) {

setState(() => exception = "Password reset is send to your email!");

} else {

setState(() {

exception = result;

});

}

}

},

child: Text("Password Reset"),

),

Center(child: Text(exception.isNotEmpty ? exception : ""),),

Future sendPasswordResetEmail(String email) async {

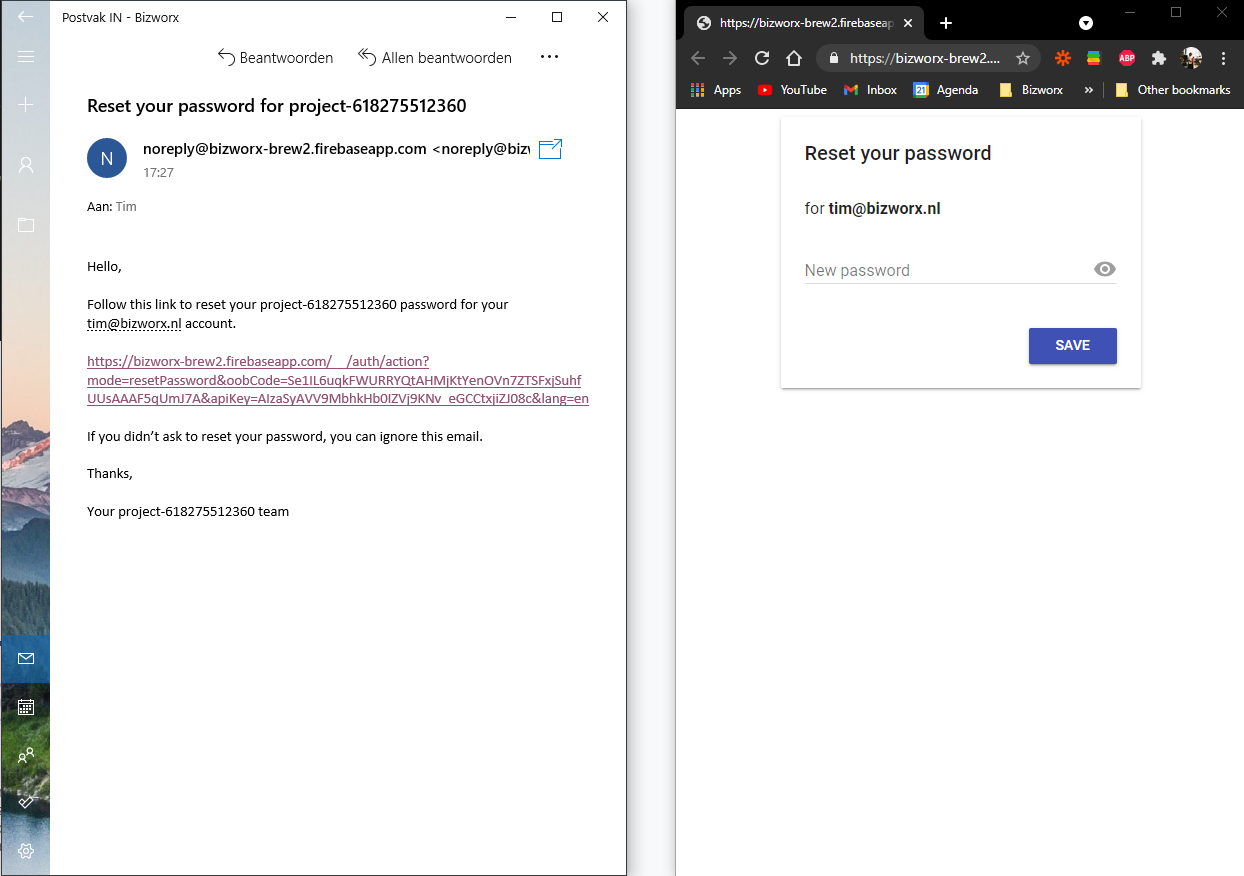
try {

await FirebaseAuth.instance.sendPasswordResetEmail(email: email);

return true;

}on FirebaseAuthException catch (e) {

return e.message;

}

}

**VliegtuigModus**

Een van de incidenten was dat een gebruiker was inlogt en vervolgens internetverbinding verloor. De gebruiker kon wel navigeren tussen het home en setting scherm maar de applicatie was onbruikbaar. Dit incident was verwarrend voor de gebruiker. Die had niet door dat verbinding met het internet was verbroken.

Om dit te voorkomen moeten er controle komen of er verbinding is met het internet.Het controleren van verbinding gaan we oplossen op twee manieren. De Firestore SDK geeft een status van de *connection state* als er gegevens opgeslagen of geladen. Als de verbinden wegvalt veranded ook de status van de *connection state*. Door het opvangen en controleren van de status kunnen we een melding geven aan de gebruiker.

Daarnaast is er op de Pub Dev website een package beschikbaar die connectivity controleert. Deze controle gaan we toevoegen aan de landing page. Wanneer een gebruiker dan volgens verbinding to het internet komt te zitten krijg die het *No Internet* scherm te zien. Deze pagina controleert ook de gebruiker is in of uitgelogd.

Future<void> \_updateConnectionStatus(ConnectivityResult result) async {

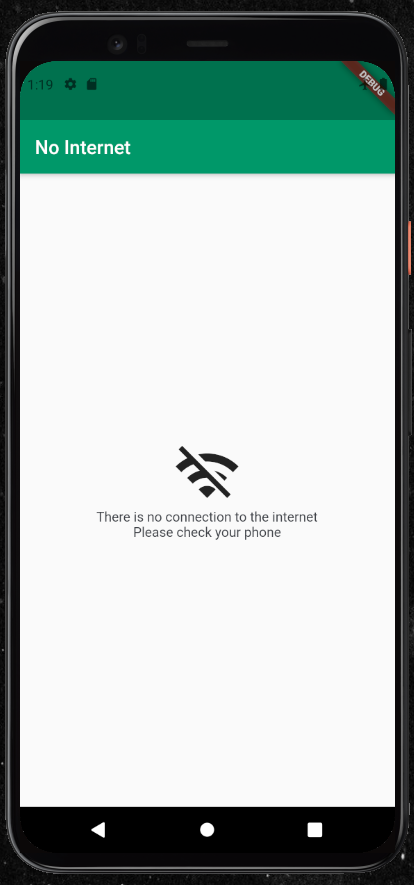
switch (result) {

case ConnectivityResult.wifi:

case ConnectivityResult.mobile:

case ConnectivityResult.none:

setState(() => \_connectionStatus = result.toString());

break;

default:

setState(() => \_connectionStatus = 'failed');

break;

}

}

@override

Widget build(BuildContext context) {

return StreamBuilder(

stream: FirebaseAuth.instance.authStateChanges(),

builder: (context, snapshot) {

if(\_connectionStatus == "ConnectivityResult.none"){

return NoConnectivty();

}

if (snapshot.connectionState == ConnectionState.active) {

User user = snapshot.data;

//if statement for authStateChange

if (user == null) {

return Authenticate();

} else {

return HomeScreen();

}

}

return Loading();

},

);

}

**Design**

Een render overflow ontstaat door het weergeven van de foutmeldingen in een form(Zie afbeelding). De foutmeldingen verandert de lengte van het form,. Omdat de length veranderd door de foutmeldingen verschuift een deel van het form onder het toetsenbord. Wanneer dit gebeurt ontstaat de incident render overflow. Het toevoegen van een *SinglChildScrollView()* maakt het mogelijk dat een gebruiker kan scrollen door een pagina. Hierdoor kan er ook geen render overflow meer ontstaan.

De foutmeldingen in de form velden worden niet weergegeven volgens het design van de applicatie. De situatie is nu dat de decoratie om het invulvelden verdwijnt op het moment dat de validator een foutmelding geeft. Door toevoegen van extra decoratie extra decoratie worden nu ook de foutmelding weergegeven volgens het design van de applicatie.

errorBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(28),

borderSide: BorderSide(color: Colors.red),

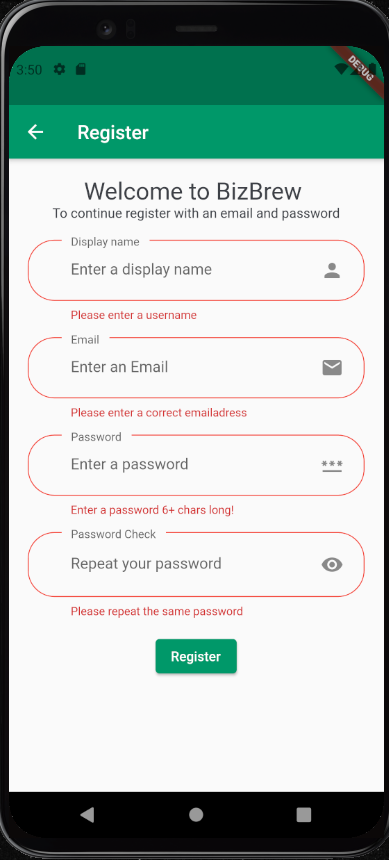
gapPadding: 10,

),

focusedErrorBorder: OutlineInputBorder(

borderRadius: BorderRadius.circular(28),

borderSide: BorderSide(color: Colors.red),

gapPadding: 10,

),

De bovenste balk van de applicatie die aangeeft op welke pagina je zit heet de App Bar. De App Bar titels van de applicatie moeten bijgewerkt worden. Sommige waren niet veranderd na het ontwikkelen. Titels zoals voor het settings scherm *“Settings Screen”* zijn veranderen we naar *“Settings”*. Hetzelfde geld voor het login, registratie en home scherm. In alle gevallen zijn de App Bar titels op dezelfde manier gewijzigd.

**Settings**

De opdrachtgever wil van zijn gebruiker weten of ze melk in hun koffie willen. Het toevoegen van dit gegeven moet in een aantal stappen worden gedaan. Als eerste moeten de functies add en update userdata aangepast worden zodat ze de toegevoegde gegevens kunnen opslaan in de database. (Zie functie)

Future updateUserData(String username, int sugars, String strength, String milk) async {

//Get uid of the current user

final uid = auth.currentUser.uid;

//add userdata to the Firestore database

return await brewCollection.doc(uid).update({

'user\_name' : username,

'sugars' : sugars,

'strength' : strength,

'milk' : milk

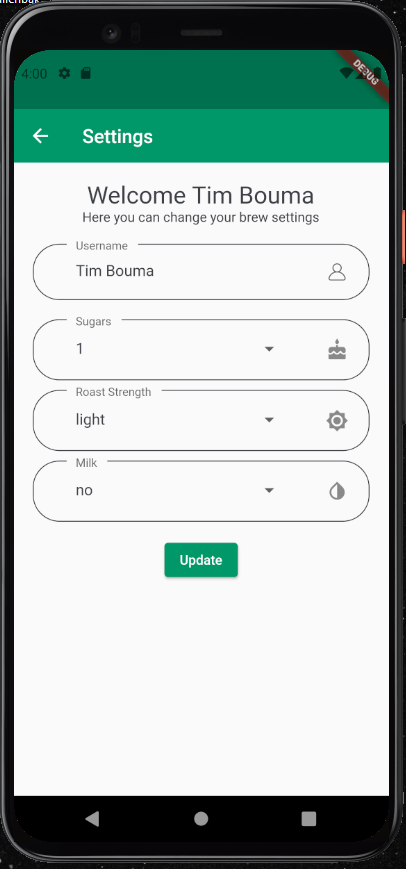
}).then((value) => print("User updated"))

.catchError((error) => print("Failed to add user: $error"));

}

## call function addUserData()

await DatabaseService().addUserData(username, 1, 'light', "no");

Vervolgens moet het settings scherm voorzien worden van een extra invoerveld voor melk. Hierin kan de gebruiker yes of no invoeren. Het weergeven van de melk voorkeur van de gebruikers gebeurt met een functie checkMilk(). 

String checkMilk(String milk) {

return milk != "no"

? "and would like some milk in his coffee."

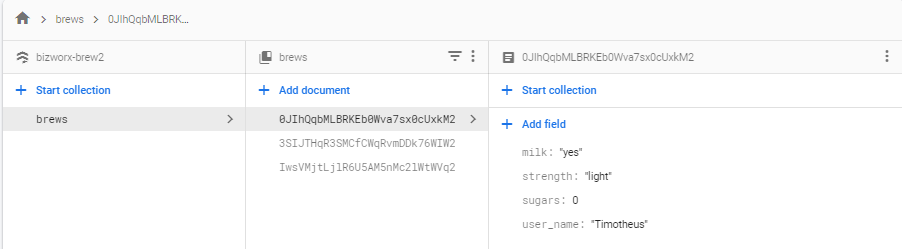
: "and don't want any milk in his coffee.";

}

Bij het registreren word in het begin vaste gegevens opgeslagen die daarna aangepast kunnen worden in het settings scherm. Om geen foutmeldingen te krijgen bij het registreren moeten deze vaste gegevens ook worden bijgewerkt. De volgorde van de gegevens zijn: gebruikersnaam, aantal suiker, roast en melk.

**Database**

De database voor dit project is een Cloud Firestore NoSQL Database. Het voordeel hiervan is dat het het ontzettend flexibel is. Data hoeft je niet te modelleren en te normaliseren. Schaalbaarheid is een ander voordeel. NoSQL db’s zijn horizontaal schaalbaar waardoor ze makkelijk mee groeien met de hoeveelheid data. De updateUserData() en addUserData() zijn functie die tijdens het onderhoud zijn aangepast. Door het gebruik van een NoSQL database zijn de aanpassingen aan de code ook gelijk doorgevoerd naar de database. In de code is er een melk veld toegevoegd. Zoals je kan zien aan de afbeelding is dit veld ook toegevoegd aan de database. Het extra veld in de database is automatisch toegevoegd door de code. Er dus niet zoals bij een SQL database eerste een extra kolom toegevoegd voordat er gegevens in worden opgeslagen.



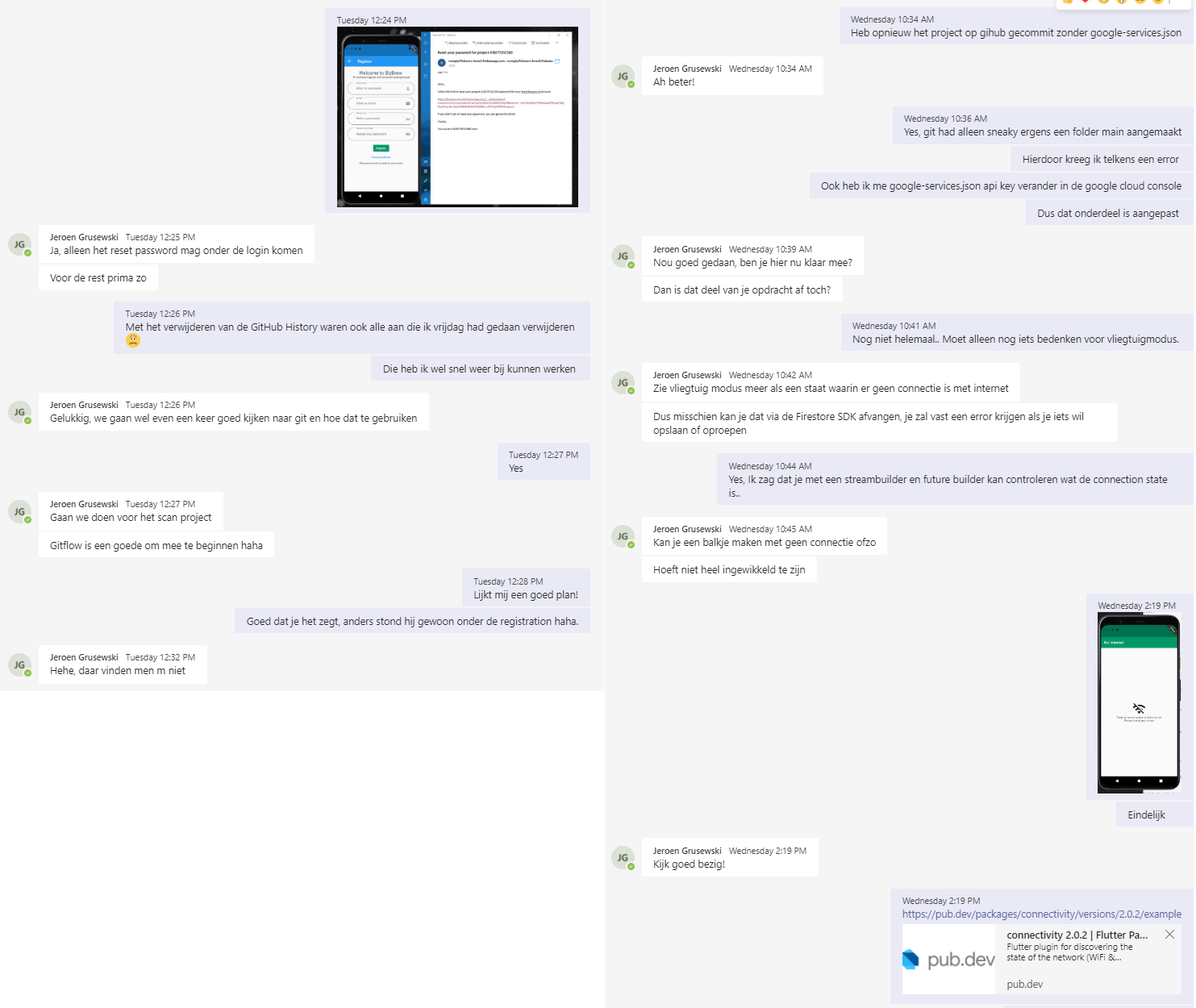
Het project kan je volledig vinden op GitHub. Naast de code kan je ook de bijbehorende documentatie vinden. De documentatie kan je zowel downloaden als een .docs bestand of bekijken als een markdown file.

Link: <https://github.com/Baqiwaqi/Biz_Brew/tree/main/documentation>

De kandidaat houdt de betrokkenen op de hoogte van de voortgang

Tijdens het bijwerken van de documentatie en code hou ik de opdrachtgever op hoogte van de voortgang. De voortgang meld ik via de mail of Microsoft Teams. Tijdens het verwerken van een incident geeft ik hem een update en vraag om goedkeuring. Vervolgens verwerk ik het volgende incident. Op deze manier zijn uiteindelijk alle incidentmeldingen verwerkt.

Nadat alle incidenten zijn verwerkt, stuur ik de opdrachtgever een mail voor bevestiging. De bijlage van de mail bestaat uit de documentatie, afbeeldingen en de GitHub repository van het project. De documentatie is aangepast met informatie over hoe de incidentmeldingen zijn verwerkt in de code.



### P1-K1-W2 Beheert gegevens

#### De kandidaat controleert gegevens van de applicatie op juistheid, volledigheid en bruikbaarheid.

**Documentatie:**

* GitHub Repository
* Programma van Eisen
* Functionele Ontwerp
* B1-K3 - De acceptatietest
* B1-K3 - Het product opleveren
* B1-K3 - Evalueer het opleverproces
* P1-K1 - Onderhoudt van een applicatie

**Algemeen**

Elke iteratie van de documentatie is goedgekeurd door de opdrachtgever. Hierdoor is de voortgang van het project afhankelijk van de goedkeuring van de opdrachtgever. Tijdens het ontwikkelen is de opdrachtgever op de hoogte gehouden van de aanpassingen aan de applicatie. Na afloop van het documenteren ben ik samen met de opdracht door de documentatie heen gaan lopen om te controleren op fouten. Hier ben ik gewezen op het verbeteren van de spelling en zinsopbouw. Veel zinnen waren opgebouwd alsof ze in het engels waren geschreven.

Verder is veel feedback geanalyseerd in kerntaak 3 en het eerste deel P1-K1-W1. Oplossingen zijn hierin beschreven en worden weerlegd met voorbeelden van code en afbeeldingen. De resultaten van de oplossingen zijn omschreven in het functioneel ontwerp. Met behulp van afbeeldingen is duidelijk te zien wat de resultaten zijn van de aanpassingen.

**Programma van Eisen**

Na *onderhoud van de applicatie* zijn ook de eisen van de applicatie aangepast. Door het ingebruik nemen van de applicatie heeft de opdrachtgever verschillen incidenten een meldingen gedaan. Deze incidentmeldingen zijn verwerkt tot nieuwe eisen waar de applicatie aan moet voldoen. Deze eisen zijn vervolgens toegevoegd aan het Programma van Eisen met een label (nieuw).

In het *Programma van Eisen*  heb ik twee stukken toegevoegd onder de kop “*Aanpassingen”*. In het stuk de *analyse*, geef ik een beschrijving van fouten die ik in een volgend project kan vermijden. Hierin geeft bijvoorbeeld advies voor een programmeer structuur BLoC. Om de kwaliteit van de applicatie op production niveaus te houden. Tijdens het testen zijn we er achter gekomen dat een fout meerdere keren is gemaakt. Dit was het valideren van gegevens voordat het in een database wordt opgeslagen. Het kwam namelijk voordat een gebruiker gegevens in een database wilde opslaan maar dat er een NULL gegeven werd opgeslagen. Oplossingen en advies voor een volgende project zijn hier in verwerkt.

De *conclusie* is een reflectie op het project. Door het maken van verschillende iteratie van de applicatie heeft de ontwikkelaar verschillende ontwikkelmethode onderzocht. Ook is er basis gelegd voor het ontwikkelen met Flutter en Firebase. Natuurlijk zijn ook fouten gemaakt tijdens het ontwikkelen. De ontwikkelaar heeft tijdens het zelftesten één scenario aangehouden die voor hem de geen foutmeldingen gaf. In de toekomst moeten voordat er code wordt geschreven eerst scenario’s worden bedacht die tijdens het programmeren getest kunnen worden. Hiermee voorkom je dat tijdens een acceptatietest nieuwe foutmeldingen naar boven komen.

**Functioneel Ontwerp**

Net zoals bij het Programma van Eisen is het Functioneel Ontwerp aangepast op functionaliteiten waar de applicatie aan moet voldoen. De functionaliteiten die zijn toegevoegd hebben ook het label (nieuw) gekregen. Dit is voor nu handig om het voor en na resultaat van elkaar te onderscheiden. Normaal gesproken zou je dit niet zien in een functioneel ontwerp.

De aanpassing die ik daarna heb doorgevoerd is die van aan het schema. Ik heb er een extra schema toegevoegd die het oude schema vervangt. Het nieuwe schema die is ook meer uitgebreid. Hierin wordt het verloop van de gebruiker duidelijk omschreven met verschillende scenario’s. In een volgend project moet dit schema gemaakt worden voordat de ontwikkelaar gaat beginnen met ontwikkelen. Hierdoor kan de programmeur zeer gericht te werk gaan en zullen er minder fout en incidentmelding ontstaan.

Onder de titel *aanpassingen* staan alle punten die uit de acceptatietest en incidentmeldingen zijn gekomen. Dit is gedaan zodat een ontwikkelaar niet door andere documenten op zoek hoeft te gaan naar deze punten. Zelf vind ik dit ook prettig. Zo kan bij elk document naar de laatste stuk gaan om in een overzicht bekijken wat de laatste aanpassingen zijn aan de applicatie.

In het functioneel wordt met behulp van afbeeldingen de applicatie weergegeven. De afbeeldingen geven het resultaat weer van alle aanpassingen die zijn gedaan aan de applicatie. Een groot deel van aanpassingen zijn al beschreven in P1-K1-W1. Maar in het Functioneel wordt dit nogmaals gedaan maar dan aan de hand van afbeeldingen. Boven de afbeeldingen is met bulletpoints beschreven wat de aanpassingen en statussen zijn van de applicatie per scherm.

**GitHub**

De laatste aanpassingen van de applicatie zijn gecommit naar GitHub. Een GitHub project is een repository die alle bestanden van je project bevatten. Het project kan je bekijken op: <https://github.com/Baqiwaqi/Biz_Brew>. Elke aanpassing die is aangebracht aan de applicatie en documentatie is hierop bijgehouden. GitHub heeft hierdoor een geschiedenis van alle veranderingen die hebben plaatsgevonden. Door het gebruik van GitHub hebben we op deze manier overzichtelijk documentatie met versiebeheer.

Aanpassingen zijn ook beschreven in de README.mk file. Het Readme bestand geeft informatie over wat het project doet. Waar het project voor gebruikt kan worden. Hoe gebruikers kunnen werken aan het project. Als je de README.mk toevoegt aan je project, herkend GitHub dit bestand automatisch. Github geeft dit bestand vervolgens onderaan het project weer. Deze informatie is ook na de aanpassingen van de code bijgewerkt en up-to-date.

De kandidaat documenteert en archiveert gegevens van de applicatie in een kwaliteitshandboek.

De documentatie is beschikbaar gemaakt in het github project: <https://github.com/Baqiwaqi/Biz_Brew>. Dit project is public gemaakt zo iedereen kan het project bekijken en downloaden. Aanpassingen en versiebeheer is zijn vastgelegd door GitHub. Functionele eisen waar het project aan moet voldoen staan in de README.mk van het project. Daarin staan ook een aantal links in die iemand kunnen helpen bij het ontwikkelen van een Flutter project. Zou toevallig iemand het project willen draaien om te kijken hoe het werkt. Dan staan er de stappen uitgewerkt in die je moet nemen omdat voorelkaar te krijgen. Wel moet moet jezelf een Firebase project opzetten.

Opzetten van het project:

```

git clone https://github.com/Baqiwaqi/Biz\_Brew.git

cd project

flutter pub get

flutter run

```

Opzetten van firebase:

* Documentatie van firebase: https://firebase.flutter.dev/docs/overview
* Documentatie van Flutter Fire: https://firebase.google.com/codelabs/firebase-get-to-know-flutter
* Google Codelab die je helpt bij het opzetten van een Flutter & Firebase project: https://firebase.google.com/codelabs/firebase-get-to-know-flutter

Hulpmiddelen die ik tijdens het programmeren heb gebruikt, staan ook in het Readme bestand. Voor ontwikkelaars die de documentatie willen bekijken, staan onderaan de README.mk links die je doorverwijzen naar de documentatie die is geschreven voor het project. De documentatie is verwerk in Google Docs. Google Docs laat makkelijk meerdere personen tegelijk werken aan een bestand. Versiebeheer wordt door Google Docs geregeld. Via de knop bestand kan je de versiegeschiedenis bekijken.

