

Kerntaak P1 - K1

P1- K1 W1: onderhoudt een applicatie

P1- K1 W1: Beheert gegevens

Tim Bouma

20-05-2021

### Inleiding

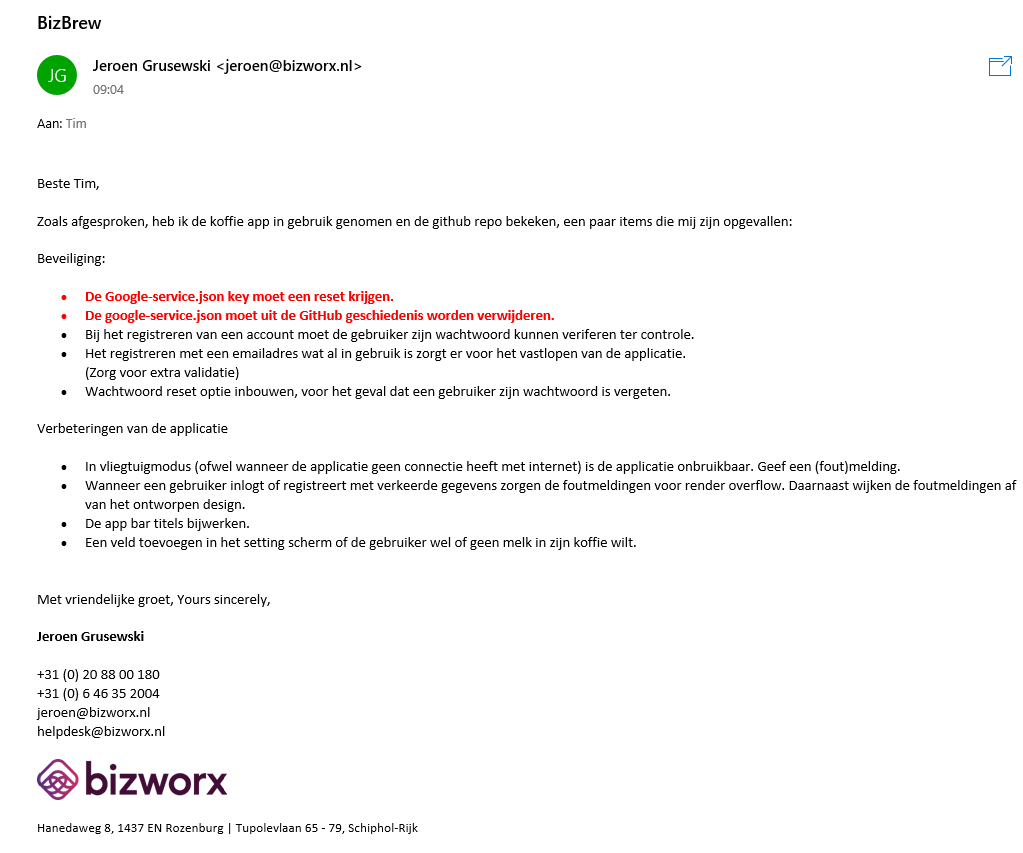
(geef aan waar het project over gaat en wat je in dit verslag gaat laten zien.)

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. Hiervoor heb ik als barista gewerkt bij de CoffeeCompany. Bij de CoffeeCompany ben ik veel bezig geweest met het zetten en branden van koffie. Het lijkt ons leuk om een applicatie te bouwen waar gebruiker kunnen inloggen en vervolgens kan aangeven wat hun koffie voorkeur is. 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. 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 versie beheer

### P1-K1-W1 Onderhoud een applicatie

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

Mail opdrachtgever:



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.

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 de 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 de foutmelding worden weergegeven aan de gebruiker. Firebase geeft op de nu achter grond een exceptie dat het e-mailadres in gebruik is. Door deze exceptie op te vangen en weergeven als een foutmelding zal de applicatie niet vastlopen. Daarnaast is de gebruiker gelijk ook op de hoogte van 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 dat wachtwoorden 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**

Het kan voorkomen dat een gebruiker zijn wachtwoord is vergeten. Daarom wordt het login scherm voorzien van een password reset. De applicatie maakt gebruik van Firebase authenticatie. Door het toevoegen van de functie sendPasswordResetEmail kan de gebruiker met het invoeren van zijn e-mailadres een wachtwoord reset opvragen. De gebruiker krijgt vervolgens een mail die hem doorverwijst naar een dashboard waar hij zijn wachtwoord kan veranderen.

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 {

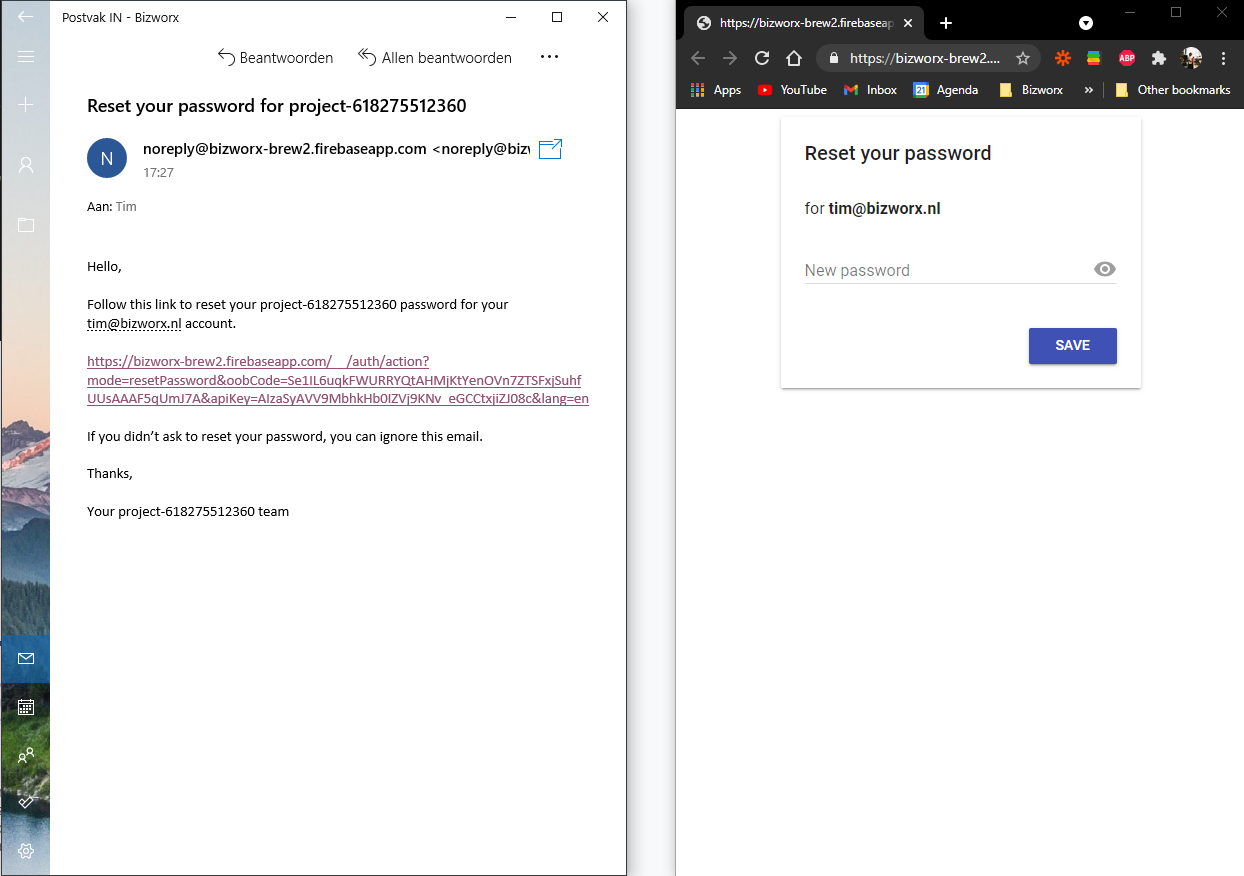
try {

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

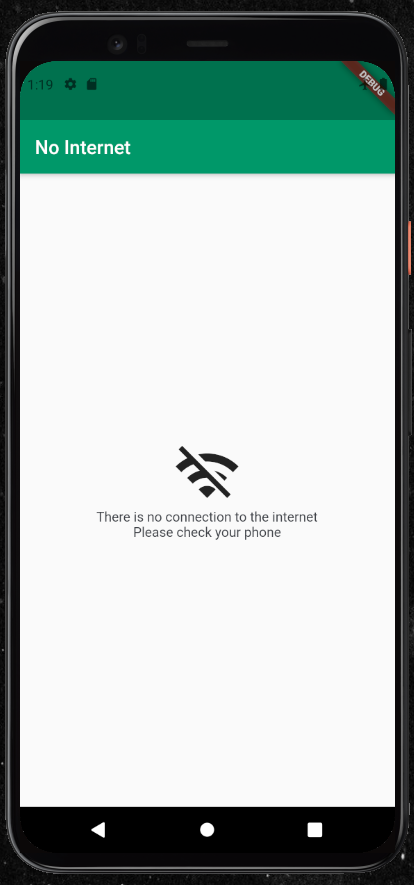
return true;

}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. Voor het controleren van verbinding gaan we twee oplossingen gebruiken. De Firestore SDK geeft een error als er gegevens opgeslagen of geladen worden zonder internet verbinding. Daarnaast is er op de Pub Dev website een package beschikbaar die connectivity controleert. Deze controle gaan we toevoegen aan de landing page. Deze pagina controleert ook de gebruiker is ingelogd.

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();

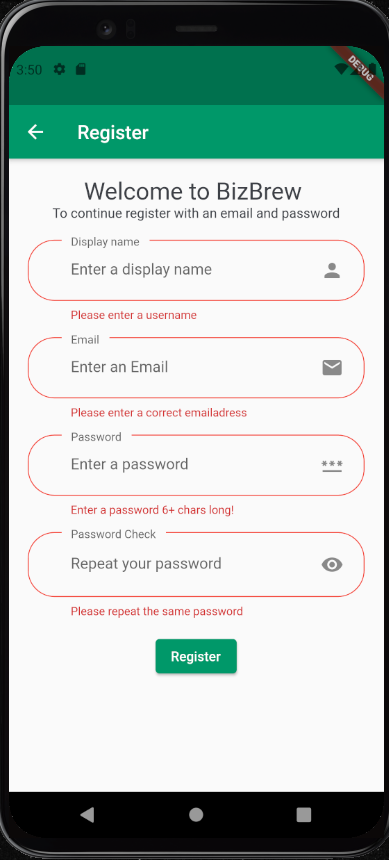
},

);

}

**Design**

Een render overflow ontstaat door het weergeven van de foutmeldingen. De foutmeldingen die veranderen de lengte van het form, waardoor een deel onder het toetsenbord terecht komt. 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 app bar titels van de applicatie moeten bijgewerkt worden. Sommige waren een overduidelijk. Zoals de titel voor het settings scherm *“Settings Screen”*. Deze titel veranderen we simpelweg door *“Settings”*. Dit moet geld hetzelfde voor het login, registratie en home scherm. In alle gevallen wordt de titels op dezelfde manier gewijzigd.

Ook wil de opdrachtgever graag weten of een gebruikers 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.

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

//Get uid of the current user

final firebaseUser = auth.currentUser;

final uid = firebaseUser.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"));

}

Daarna moet het settings scherm worden voorzien van een extra form veld waar de gebruiker zijn voorkeur kan aanpassen. Uiteindelijk moet de opgeslagen gegevens worden weergegeven op het homescherm. Een functie bepaald aan de hand van de gegevens wat er op het scherm wordt weergegeven.

String checkMilk(String milk) {

return milk != "no"

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

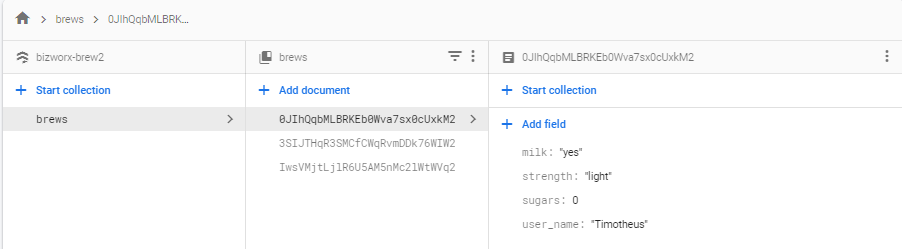
: "and don't want any milk in his coffee.";

}

**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. Schaalbaar is een ander voordeel. NoSQL db’s zijn horizontaal schaalbaar waardoor ze makkelijk mee groeien met de hoeveelheid data.

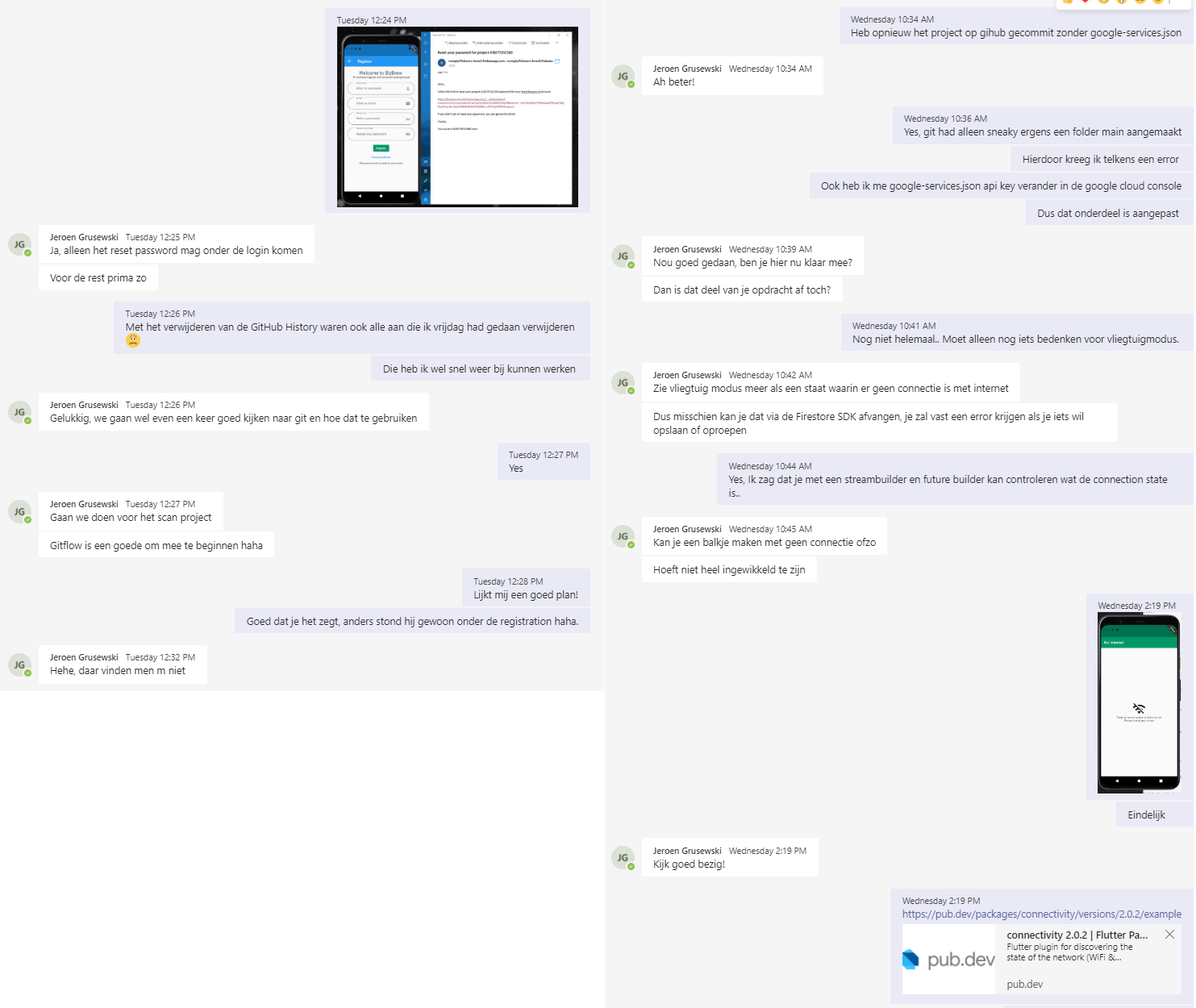
De updateUserDat() functie hierboven is tijdens het onderhoud aangepast. Door het gebruik van een NoSQL database zijn de aanpassingen aan de code ook gelijk doorgevoerd naar de database. In de code is er een melk veld toegevoegd. Zoals je kan zien aan de afbeelding is dit veld ook toegevoegd aan de database.



Het project kan je vinden op GitHub als je alle code van het project wilt bekijken: <https://github.com/Baqiwaqi/Biz_Brew>

De kandidaat houdt de betrokkenen op de hoogte van de voortgang

Tijdens het bijwerken hou ik de opdrachtgever op hoogte van de voortgang. Dit doe ik via het portaal Microsoft Teams. Bij het verwerken van een incident geeft ik hem een update en vraag om goedkeuring. Vervolgens verwerk ik het volgende incident enzovoort.Nadat alle incidenten zijn verwerkt stuur ik de opdrachtgever een mail. In de mail bevestig ik dat alle incidenten zijn verwerkt. In de bijlage van de mail staat de documentatie 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.

(Geef aan welke documenten er bestaat en geef verslag van verbeteringen die je erin hebt aangebracht.)

**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 de door de opdrachtgever. Hierdoor was 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 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. 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 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 voor dat een gebruiker gegevens in een database wilde opslaan maar dat er een NULL gegeven uiteindelijk 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 een scenario aangehouden die voor hem de minste foutmeldingen gaf. In de toekomst moeten voordat er code wordt geschreven eerst scenario;s worden bedacht die tijdens het programmeren getest kunne 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 is moet dit schema gemaakt worden voordat de ontwikkelaar gaat beginnen met het schrijven code. 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. Zou kan bij elk document naar de laatste titel gaan om in een overzicht bekijken wat de laatste aanpassingen waren van 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 i Functioneel wordt dit nogmaals gedaan maar dan aan de hand van afbeeldingen. Boven de afbeeldingen is met bulletpoints beschreven wat de aanpassingen en status is van de applicatie per scherm.

**GitHub**

De laatste aanpassingen van de applicatie zijn gecommit naar GitHub. Een GitHub project is 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.

**Gegevens:**

* Specificaties
* Inloggegevens
* Databasegegevens
* Aanpassingen
* Eisen
* Wijzigingen
* Versiebeheer

(Laat zien waar je de documentatie opslaat en hoe je ervoor zorgt dat iedereen erbij kan.)