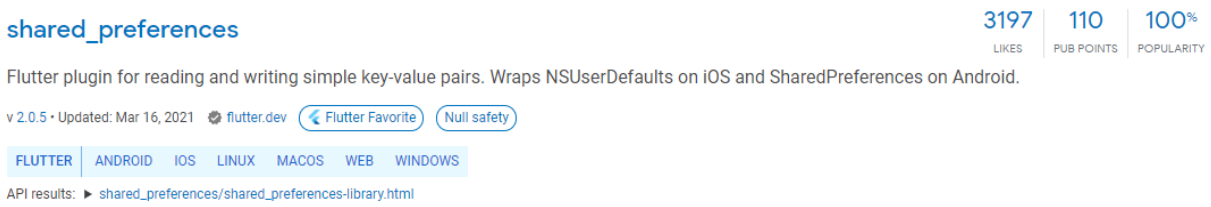
Werken met data in Flutter- 1

Shared Preferences

Een van de meest interessante opties voor gegevensopslag die Android zijn gebruikers biedt, zijn gedeelde voorkeuren (shared preferences). Gedeelde voorkeuren is de manier waarop men kleine hoeveelheden primitieve gegevens kan opslaan en ophalen als sleutel / waarde-paren naar een bestand op de apparaatopslag, zoals String, int, float, Boolean die uw voorkeuren vormen in een XML-bestand in de app op de apparaatopslag. Gedeelde voorkeuren kunnen worden gezien als een woordenboek of een sleutel / waarde-paar. U kunt bijvoorbeeld de sleutel 'gebruikersnaam' hebben en voor de waarde kunt u de gebruikersnaam van de gebruiker opslaan. En dan zou je dat kunnen achterhalen aan de hand van de sleutel (hier gebruikersnaam). U kunt een eenvoudige API voor gedeelde voorkeuren hebben die u kunt gebruiken om voorkeuren op te slaan en ze indien nodig terug te halen. De klasse Shared Preferences biedt API's voor het lezen, schrijven en beheren van deze gegevens. In Flutter kunnen we gebruik maken van een Dart package die het werken met shared preferences heel eenvoudig maakt:

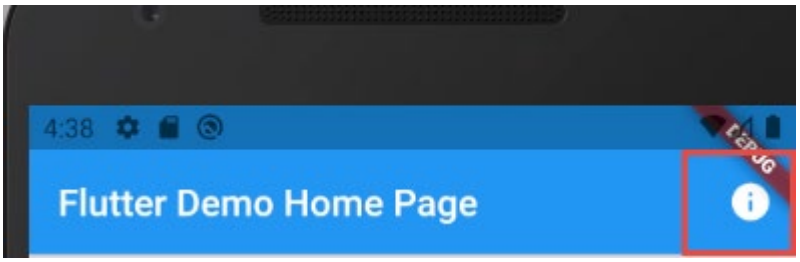


Opdracht 1a

Maak een Flutter App (default voorbeeld app) waarin de gebruiker de volgende gegeven kan opslaan in de app:

- Naam
- Email
- Geboortedatum
- Aantal geregistreerde clicks

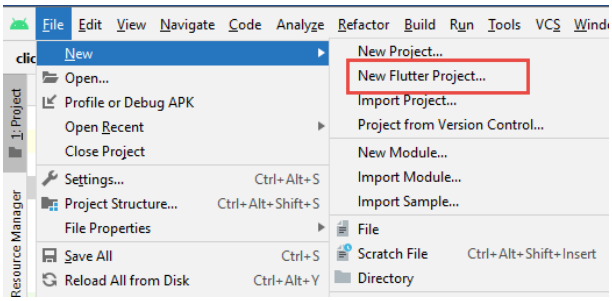
Het invoeren/wijzigen van de gegevens moet aangeroepen kunnen worden middels een Actions knop in de AppBar:

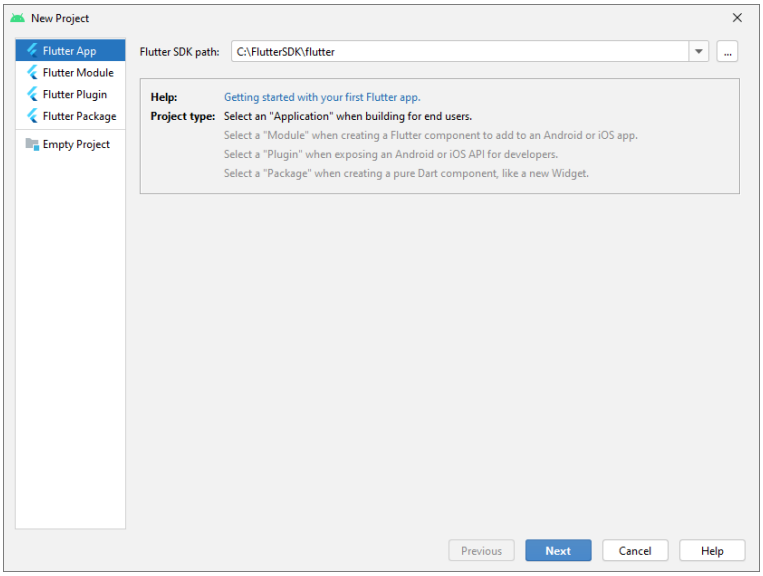


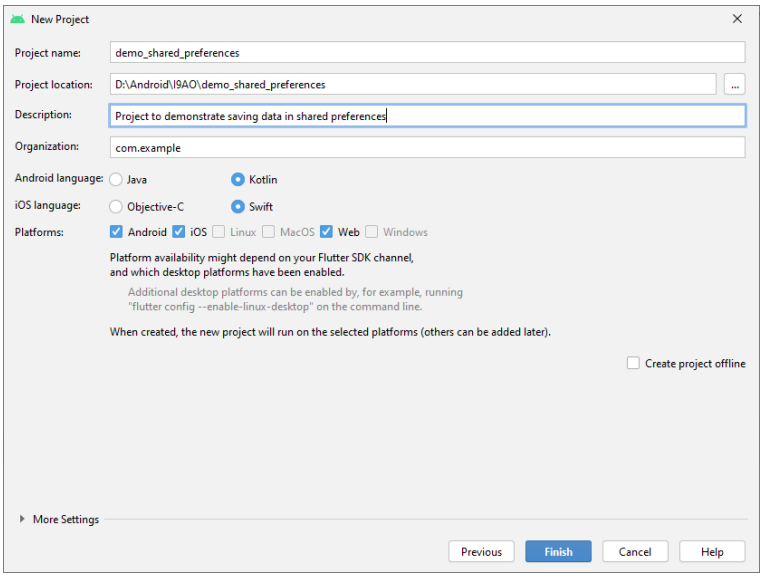
Voor het aanroepen van een tweede scherm in je app gebruik je de Navigator. De gegevens worden opgeslagen in shared preferences.

Stap 1: Project aanmaken

Maak een nieuw Flutter project aan via **F**ile -> **N**ew -> **N**ew Flutter Project …







Vul een projectnaam in en eventueel een beschrijving. De rest van de instellingen is oké. Klik op *Finish*.

Er is nu een voorbeeldapplicatie aangemaakt waarin de gebruiker kan bijhouden hoe vaak er op de button is geklikt.

Stap 2 – Invoerscherm aanmaken

We gaan nu het invoerscherm aanmaken waarmee de gebruiker de gewenste data kan invoeren. Voor het invoeren van de naam en het e-mailadres gebruiken we tekstvakken en voor het invoeren van de datum gaan we in een latere stap een data picker toevoegen, waarbij de geboortedatum in een read-only tekstvak wordt getoond. Als we de geboortedatum alten invoeren middels een tekstvak dan heeft de gebruiker de mogelijkheid om van alles in te voeren, terwijl we natuurlijk uitsluitend geldige datums willen accepteren (denk o.a. aan schrikkeljaren).

Maak eerst in de map **Lib** een nieuw dart bestand aan met als naam ***user_data.dart***. Voer in dit bestand de volgende code in. Dit moet je overtypen uit onderstaande afbeelding. Daar leer je meer van dan van het domweg kopiëren en plakken.

```
import 'package:flutter/material.dart';

class UserData extends StatefulWidget {
  UserData();

  @override
  _UserDataState createState() => _UserDataState();
}

class _UserDataState extends State<UserData> {
  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        appBar: AppBar(
          title: Text('User data'),
        ),
        body: Padding(
          padding: const EdgeInsets.all(24.0),
          child: SingleChildScrollView(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.start,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                TextField(
                  decoration: InputDecoration(
                    border: OutlineInputBorder(),
                    labelText: 'Your name',
                  ),
                ),
                SizedBox(
                  height: 16.0,
                ),

              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

Het scherm bevat nu een label en een tekstvak voor het invoeren van de naam. We gaan nu eerst zorgen dat de pagina vanuit het hoofdscherm kan worden geopend middels een actions knop in de AppBar. Later moet je zelf de controls toevoegen voor het invoeren van het e-mailadres en de geboortedatum.

Stap 3 – Action knop toevoegen aan AppBar

We gaan nu een knop toevoegen aan de AppBar van het hoofdscherm, waarmee het formulier geopend kan worden dat we in de vorige stap hebben aangemaakt.

Eerst moeten we de package waarin we het invoerscherm hebben gemaakt importeren in het hoofdscherm (main.dart):

```
import 'package:flutter/material.dart';
import 'package:click_counter_demo/user_data.dart';
```

Vervolgens voegen we de volgende code toe aan de AppBar in main.dart (na de title):

```
return Scaffold(
  appBar: AppBar(
    // Here we take the value from the MyHomePage object that was created by
    // the App.build method, and use it to set our appbar title.
    title: Text(widget.title),
    actions: [
      IconButton(
        icon: Icon(Icons.info),
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => UserData(),
            ),  // MaterialPageRoute
          );
        },
      ),  // IconButton
    ],
  ),  // AppBar
```

Test nu je applicatie in de emulator of op een Android telefoon. Als het goed is moet je nu een knop hebben in de AppBar, waarmee het tweede scherm wordt geopend. De invoerpagina is nog lang niet af, maar je moet deze wel kunnen openen en sluiten.

Stap 4 – Het invoerscherm uitbreiden voor e-mail

Voeg aan het invoerformulier een label en tekstvak toe om een e-mailadres in te voeren:

```
      TextField(
        decoration: InputDecoration(
          border: OutlineInputBorder(),
          labelText: 'Your emailaddress',
        ),  // InputDecoration
      ),  // TextField
      SizedBox(
        height: 16.0,
      ),  // SizedBox
```

Stap 5 – TextEditingControls toevoegen

Om de tekst uit de tekstvakken te kunnen gebruiken maken we gebruik van een TextEditingController. Daarmee kunnen we o.a. de tekst in een tekstvak ophalen of instellen. We maken deze aan binnen de class _UserDataState:

```
class _UserDataState extends State<UserData> {
  TextEditingController nameController = TextEditingController();
  TextEditingController emailController = TextEditingController();
  TextEditingController birthdateController = TextEditingController();
```

Vervolgens voegen we de controllers toe aan de tekstvakken:

```
TextField(
  controller: nameController,
  decoration: InputDecoration(
    border: OutlineInputBorder(),
    labelText: 'Your name',
  ),  // InputDecoration
),  // TextField
SizedBox(
  height: 16.0,
),  // SizedBox
TextField(
  controller: emailController,
  decoration: InputDecoration(
    border: OutlineInputBorder(),
    labelText: 'Your emailaddress',
  ),  // InputDecoration
),  // TextField
SizedBox(
  height: 16.0,
),  // SizedBox
```

Stap 6 – Een date picker toevoegen voor de geboortedatum

We gaan nu een date picker toevoegen waarmee de gebruiker zijn geboortedatum kan instellen.

We beginnen met het definiëren van een variabele om de datum in op te slaan. Standaard stellen we de waarde in op de huidige datum:

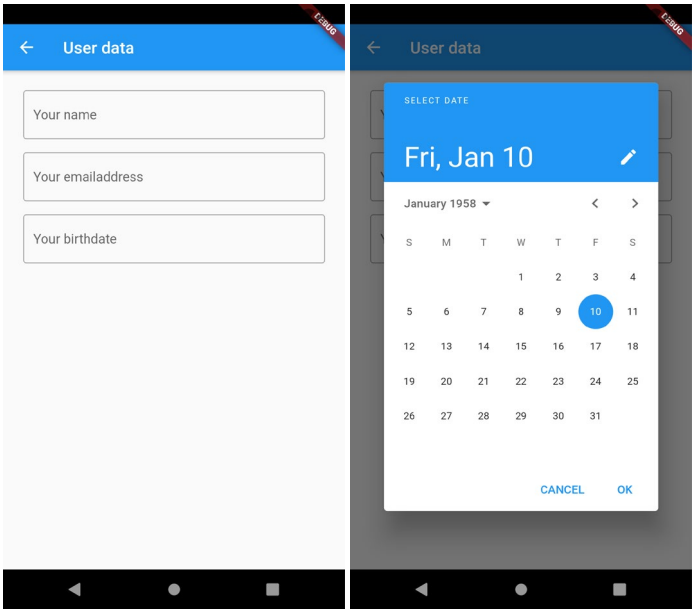
```
class _UserDataState extends State<UserData> {
  TextEditingController nameController = TextEditingController();
  TextEditingController emailController = TextEditingController();
  TextEditingController birthdateController = TextEditingController();

  String birthdate = DateTime.now().toString();
```

Vervolgens maken we een tekstvak aan waarin de datum getoond gaat worden en deze voegen we toe onder de controls van de e-mail:

```
TextField(
  onTap: () async {
    var date = await showDatePicker(
        context: context,
        initialDate:
            DateTime.parse(birthdate) ?? DateTime.now(),
        firstDate: DateTime(1900),
        lastDate: DateTime(2100));
    birthdateController.text = date.toString().substring(0, 10);
  },
  controller: birthdateController,
  decoration: InputDecoration(
    border: OutlineInputBorder(),
    labelText: 'Your birthdate',
  ),  // InputDecoration
),  // TextField
SizedBox(
  height: 16.0,
),  // SizedBox
```

Het is nu tijd om je applicatie weer te testen en te controleren of de layout van het invoerscherm er enigszins acceptabel uitziet. Door op het tekstvak van geboortedatum te klikken moet de date picker worden geopend.



Stap 7 – Het aantal clicks laten zien in het invoerscherm

Behalve dat in het invoerscherm de gebruikersdata moeten worden getoond is het ook de bedoeling om het totale aantal clicks te laten zien. Daarvoor moeten we de waarde van de variabele **_counter** in het hoofdscherm doorsturen naar het invoerscherm. Daarvoor passen we eerst de class *UserData* aan door een variabele aan te maken voor het aantal clicks, die we vervolgens via de constructor in gaan stellen:

```
class UserData extends StatefulWidget {
  UserData({this.clicks = 0});
  int clicks;
  @override
  _UserDataState createState() => _UserDataState();
}
```

In het hoofdscherm moeten we vervolgens de waarde van **_counter** meegeven bij het aanroepen van de constructor van UserData:

```
IconButton(
  icon: Icon(Icons.info),
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => UserData(
          clicks: _counter,
        ), // UserData
      ), // MaterialPageRoute
    );
  },
), // IconButton
```

Rest alleen nog het toevoegen van een Tekst widget onder de geboortedatum:

```
TextField(
    onTap: () async {
      var date = await showDatePicker(
          context: context,
          initialDate:
              DateTime.parse(birthdate) ?? DateTime.now(),
          firstDate: DateTime(1900),
          lastDate: DateTime(2100));
      birthdateController.text = date.toString().substring(0, 10);
    },
    controller: birthdateController,
    decoration: InputDecoration(
      border: OutlineInputBorder(),
      labelText: 'Your birthdate',
    ), // InputDecoration
  ), // TextField
SizedBox(
    height: 16.0,
  ), // SizedBox
Text('Number of clicks: ${widget.clicks}'),
```

Test nu weer je applicatie en kijk of het aantal clicks juist wordt weergegeven in het invoerscherm.

Stap 8 – Het aantal clicks opslaan in shared preferences

Voor het opslaan van data in shared preferences maken we gebruik van een package (https://pub.dev/packages/shared_preferences). Lees de documentatie bij de package om te zien hoe je deze dient te gebruiken en hoe je het kunt installeren.
Voeg de package toe aan *pubspec.yaml*:

```
  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.0
  shared_preferences: ^2.0.5
```

Vergeet niet om eerst de package te installeren door op **Pub get** te klikken.

Om data toe te voegen aan of op te halen uit shared preferences maken we een helper file aan. Voeg een map toe aan **lib** met als naam **helpers** en voeg aan die map een nieuw dart bestand toe met als naam **shared_prefs_utils.dart**. Neem onderstaande code over. Ook nu is weer bewust gekozen voor het overtypen van de code:

```
import 'package:shared_preferences/shared_preferences.dart';

class MySharedPreferences {
  MySharedPreferences._privateConstructor();

  static final MySharedPreferences instance =
      MySharedPreferences._privateConstructor();

  setStringValue(String key, String value) async {
    SharedPreferences myPrefs = await SharedPreferences.getInstance();
    myPrefs.setString(key, value);
  }

  Future<String> getStringValue(String key) async {
    SharedPreferences myPrefs = await SharedPreferences.getInstance();
    return myPrefs.getString(key) ?? "";
  }

  setIntegerValue(String key, int value) async {
    SharedPreferences myPrefs = await SharedPreferences.getInstance();
    myPrefs.setInt(key, value);
  }

  Future<int> getIntegerValue(String key) async {
    SharedPreferences myPrefs = await SharedPreferences.getInstance();
    return myPrefs.getInt(key) ?? 0;
  }

  setBooleanValue(String key, bool value) async {
    SharedPreferences myPrefs = await SharedPreferences.getInstance();
    myPrefs.setBool(key, value);
  }

  Future<bool> getBooleanValue(String key) async {
    SharedPreferences myPrefs = await SharedPreferences.getInstance();
    return myPrefs.getBool(key) ?? false;
  }

  Future<bool> containsKey(String key) async {
    SharedPreferences myPrefs = await SharedPreferences.getInstance();
    return myPrefs.containsKey(key);
  }

  removeValue(String key) async {
    SharedPreferences myPrefs = await SharedPreferences.getInstance();
    return myPrefs.remove(key);
  }

  removeAll() async {
    SharedPreferences myPrefs = await SharedPreferences.getInstance();
    return myPrefs.clear();
  }
}
```

Toelichting:

- Eerst wordt de package geïmporteerd.

- Er wordt een class aangemaakt met daarin diverse methodes om diverse datatypes weg te kunnen schrijven en op te kunnen halen. Voor nadere uitleg over het gebruik van de package: zie de documentatie bij de package op https://pub.dev/packages/shared_preferences
  Bij het ophalen gegevens wordt gebruik gemaakt van Futures. Het gaat voor deze tutorial te ver om daar dieper op in te gaan, maar het is wel belangrijk dat je weet waarom Futures worden gebruikt en hoe je ze moet gebruiken. De volgende video geeft een aardig inzicht: https://youtu.be/OTS-ap9_aXc maar ook in les 8 van de cursus Flutter op LinkedIn wordt het erg goed uitgelegd.
- Er wordt een methode toegevoegd om te kijken of een bepaalde key bestaat (containsKey).
- Er wordt een methode toegevoegd om een waarde te verwijderen (removeValue).
- Er wordt een methode toegevoegd om alle waarden te verwijderen (removeAll).

Deze helper-class kun je in ieder ander project ook gebruiken. Je hoeft dan alleen maar te bedenken welke methode je moet aanroepen om een waarde op te slaan of op te vragen.

In ons project gaan we op 2 plaatsen gebruiken maken van deze helper-class:

1. Zodra de gebruiker in het hoofdscherm op de knop klikt wordt de teller verhoogd. De nieuwe waarde gaan we opslaan in shared preferences.
2. In het invoerscherm gaan we een knop toevoegen waarmee de gebruiker de ingevoerde waardes kan opslaan. Bij het opnieuw openen van het invoerscherm halen we de opgeslagen waardes weer op en tonen die in de tekstvakken. Voor het bewerken van de tekst in de tekstvakken gebruiken we TextControllers, waarvan je er al één hebt toegevoegd aan het invoerscherm.

We gaan nu eerst de hoeveelheid clicks opslaan. Importeer eerst het bestand **shared_prefs_utils.dart** in main.dart:

```
import 'helpers/shareded_prefs_utils.dart';
```

Voeg de volgende code toe aan het hoofdscherm, net nadat de waarde van **_counter** is verhoogd:

```
void _incrementCounter() {
  setState(() {
    // This call to setState tells the Flutter framework that something has
    // changed in this State, which causes it to rerun the build method below
    // so that the display can reflect the updated values. If we changed
    // _counter without calling setState(), then the build method would not be
    // called again, and so nothing would appear to happen.
    _counter++;
    MySharedPreferences.instance.setIntegerValue('clicks', _counter);
  });
}
```

Wat nog rest is dat de opgeslagen waarde moet worden opgehaald uit shared preferences, zodat de waarde in **_counter** juist wordt ingesteld, bijvoorbeeld als de applicatie opnieuw wordt opgestart. Voeg de volgende code toe aan de methode build:

```
@override
Widget build(BuildContext context) {
  MySharedPreferences.instance
      .getIntegerValue('clicks')
      .then((value) => setState(() {
            _counter = value;
          }));
```

Test je applicatie door de teller te verhogen en daarna de applicatie af te sluiten en opnieuw op te starten. De teller moet nu beginnen waar hij de laatste keer was gebleven.

Stap 9 – De user data opslaan in shared preferences

We gaan nu een knop toevoegen aan het invoerscherm, waarmee de ingevoerde gegevens opgeslagen worden in shared preferences.

Voor het toevoegen van de knop maken we gebruik van de volgende code (plaats deze onder de laatste Tekst widget):

```
SizedBox(
  height: 16.0,
),  // SizedBox
Text('Number of clicks: ${widget.clicks}'),
IconButton(
  onPressed: () {
    MySharedPreferences.instance
        .setStringValue("birthdate", birthdateController.text);
    MySharedPreferences.instance
        .setStringValue("name", nameController.text);
    MySharedPreferences.instance
        .setStringValue("email", emailController.text);
  },
  icon: Icon(Icons.save),
),  // IconButton
```

In het onPressed event worden de waarden opgeslagen in shared preferences.

Stap 10 – De teller resetten

Om de gebruiker de gelegenheid te geven de teller te resetten naar 0 voegen we nog de volgende code toe, waarmee een button wordt toegevoegd om de teller te resetten:

```
    MySharedPreferences.instance
        .setStringValue("email", emailController.text);
  },
  icon: Icon(Icons.save),
), // IconButton
IconButton(
  onPressed: () {
    setState(() {
      widget.clicks = 0;
      MySharedPreferences.instance
          .setIntegerValue('clicks', widget.clicks);
    });
  },
  icon: Icon(Icons.clear),
), // IconButton
```

Stap 11 – User data teruglezen in het invoerformulier

We zijn er bijna. Als laatste stap moeten de user data worden teruggelezen in het invoerscherm als dat opnieuw wordt geopend, zodat de gebruiker zijn eigen instellingen kan zien en aanpassen, zonder het telkens opnieuw te moeten invoeren.

Daarvoor maken we eerst een methode aan met als naam readUserData en deze roepen we aan in een ook nieuw te maken methode initState(). Deze laatste methode is een override van dezelfde methode uit de hoger class. Iedere keer als het scherm wordt geopend wordt deze methode aangeroepen door het operating system.

```
void readUserData() {
  MySharedPreferences.instance
      .getStringValue("birthdate")
      .then((value) => setState(() {
            if (value != null) {
              birthdateController.text = value;
            }
          }));
  MySharedPreferences.instance
      .getStringValue("name")
      .then((value) => setState(() {
            nameController.text = value;
          }));
  MySharedPreferences.instance
      .getStringValue("email")
      .then((value) => setState(() {
            emailController.text = value;
          }));
}

@override
void initState() {
  super.initState();
  readUserData();
}
```

Het moment is gekomen voor de laatste test van de app.

1. Start je applicatie.
2. Klik enkele keren op de button.
3. Ga naar het invoerscherm.
4. Voer je naam in.
5. Voer je emailadres in.
6. Klik op het veld van de geboortedatum.
7. Selecteer jouw geboortedatum.
8. Sla de gegevens op door op de knop met de schijf te klikken.

Bronnen:

How to add data picker in Flutter

Dart package – shared_preferences

Flutter cookbook