

# Mobile App "CoinDart"

von Alexander Neumann und Pascal Groß

## Inhalt

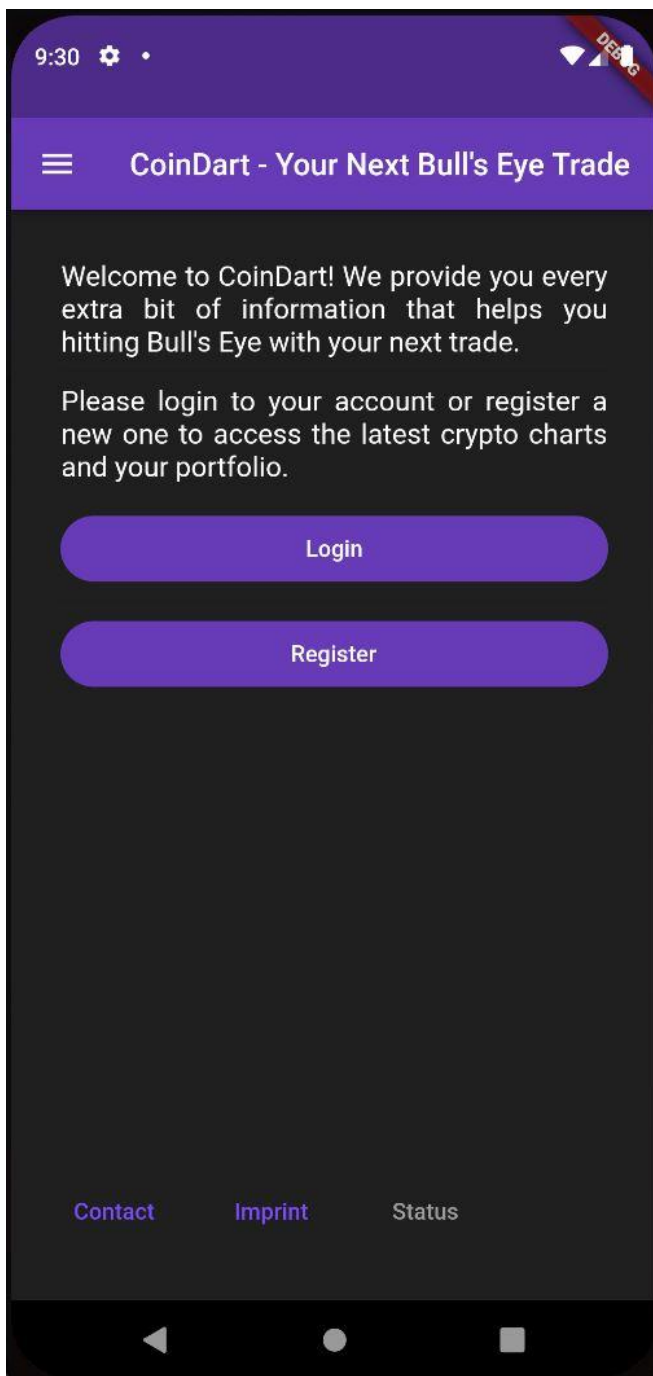
Einführung .....	3
Schnellstart .....	3
Hardwarevoraussetzungen .....	4
Versionsverwaltung .....	4
Entwicklerteam .....	4
Technologie: Überblick .....	4
Übersicht über die Screens .....	5
Für alle Benutzer .....	5
Homescreen .....	5
Loginscreen .....	7
Registerscreen .....	8
Contactscreen .....	9
Imprintscreen .....	10
Statusscreen .....	11
Nur für authentifizierte Benutzer .....	13
Coinlistscreen .....	13
Favoritescreen .....	14
Detailscreen .....	15
Profilescreen .....	16
Projektdetails .....	17
Ordnerstruktur .....	17
Beispielhafte API-Calls .....	18
Statusscreen .....	18
Coinlistscreen .....	19
Detailscreen .....	20

## Einführung

Die mobile App **CoinDart – Your Next Bull's Eye Trade** bietet ihren Nutzern wichtige Daten und Informationen über den Markt der Kryptowährungen. Der Benutzer erhält eine Übersicht über die meistgehandelten Kryptowährungen und kann das Marktgeschehen verfolgen. Darüber hinaus kann der Benutzer sich in der App registrieren, anmelden und Kryptowährungen als Favorit markieren, um den nächsten gewinnbringenden Trade durchzuführen.

## Schnellstart

Nach dem Starten der Anwendung werden mehrere Elemente auf dem Bildschirm angezeigt. Ein in die Anwendung einführender Text sowie zwei Button. Um die Grundfunktionalität von CoinDart nutzen zu können, führen Sie folgende Schritte aus:



1. Berühren Sie den "Register"-Button, um Ihren neuen Account zu registrieren.
2. Geben Sie eine Ihnen verfügbare E-Mail-Adresse sowie ein Passwort ein. Diese Daten nutzen Sie, um sich in Zukunft gegenüber der Anwendung zu authentifizieren.
3. Nach erfolgreicher Registrierung werden Sie zum *Loginscreen* weitergeleitet.
4. Geben Sie Ihre E-Mail-Adresse und Ihr Passwort ein, um sich anzumelden.
5. Nach erfolgreichem Login gelangen Sie zum *Coinlistscreens*.
6. Auf dem Coinlistscreens werden Kryptowährungen und deren Kurse angezeigt, außerdem können Sie Kryptowährungen als Favorit markieren. Dies ist die Grundfunktionalität von CoinDart, Sie sind startbereit. Details dazu erhalten Sie in den kommenden Abschnitten dieser Dokumentation.

## Hardwarevoraussetzungen

Sollte der Schnellstart aus beliebigen Gründen nicht auf Anhieb gelungen sein, sind die Hardwarevoraussetzungen zu prüfen:

- Um einen Account registrieren, sich in diesen einzuloggen und Kryptowährungsdaten abrufen zu können, ist eine Internetverbindung erforderlich
- Der Google Play Store muss auf dem Gerät installiert sein
- Folgende Geräte wurden während der Entwicklungsarbeiten getestet:
  - Google / LG Nexus 5 (2013) mit einer Auflösung von 1080 x 1920 und einer Bildschirmgröße von 4,95"
  - Google Pixel (2013) mit einer Auflösung von 1080 x 1920 und einer Bildschirmgröße von 5,0"
  - Google / LG Nexus 5X (2015) mit einer Auflösung von 1080 x 1920 und einer Bildschirmgröße von 5,2"

## Versionsverwaltung

Das Projekt wird verwaltet via Git und GitHub: <https://github.com/interN3rd/coindart>

## Entwicklerteam

Das Projekt wird von zwei Entwicklern getragen:

Alexander Neumann

Pascal Groß

## Technologie: Überblick

Die App wird geschrieben in der Programmiersprache Dart. Als Entwicklungswerkzeug wird das Framework Flutter verwendet. Für die Entwicklungsarbeiten wird die Entwicklungsumgebung Android Studio verwendet.

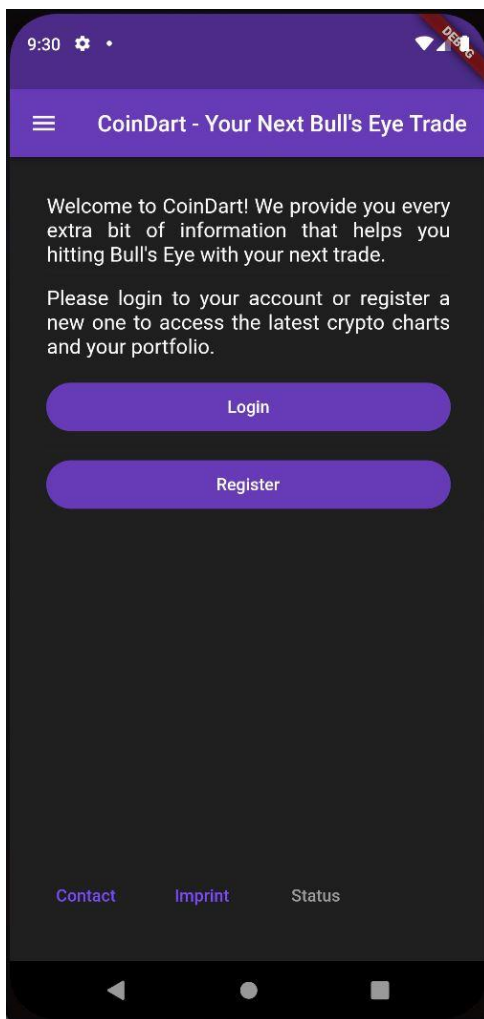
## Übersicht über die Screens

Manche Screens sind für alle Benutzer zugänglich: "*Home*", "*Login*", "*Register*", "*Contact*", "*Imprint*" und "*Status*". Um aber alle Funktionen von CoinDart nutzen zu können, ist ein registrierter Benutzeraccount erforderlich.

Diese Screens sind nur für authentifizierte Benutzer zugänglich: "*Coinlist*", "*Favorite Coins*", "*Details*" und "*User Profile*".

### Für alle Benutzer

#### Homescreen



Vom Homescreen aus sind folgende Funktionen erreichbar:

In der *AppBar* befindet sich ein *Burger-Menu*. Wird dieses angeklickt, öffnet sich ein Menü mit den folgenden Optionen für nicht authentifizierte Nutzer: "*Home*", "*Login*", "*Contact*", "*Imprint*", "*Status*" und "*X*", um das Menü zu schließen.

Außerdem befinden sich zwei Button auf dem Homescreen: "*Login*" und "*Register*", um sich in einen bestehenden Account einzuloggen oder einen neuen Account zu registrieren.

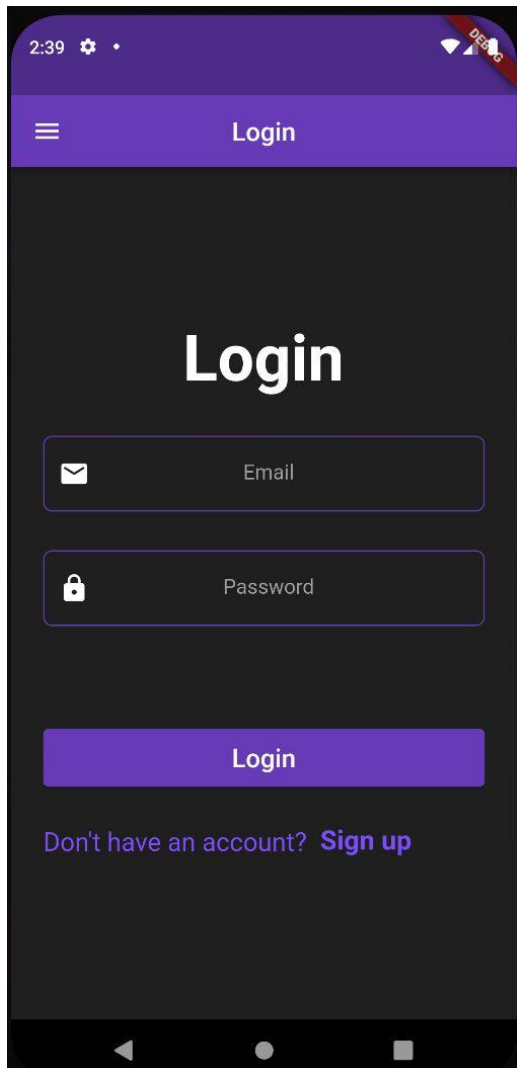
Am unteren Bildschirmrand befinden sich die Optionen "*Contact*", "*Imprint*" und "*Status*", um ein Kontaktformular aufzurufen, das Impressum aufzurufen oder eine Statusseite aufzurufen, die anzeigt, ob ein Benutzer authentifziert und die API verfügbar ist.

Abhängig vom Status der Authentifizierung werden dem Benutzer auf dem *Homescreen* unterschiedliche Inhalte angezeigt:

```
body: Container(  
  padding: const EdgeInsets.all( 30 ),  
  child: Column(  
    children: <Widget> [  
      firebaseUser == null ? const GreetingText() : const  
LoggedInText(),  
      const Divider(),  
      firebaseUser == null ? const LoginButton() : const  
InvisibleWidget(),  
      const Divider(),  
      firebaseUser == null ? const RegisterButton() : const  
InvisibleWidget(),  
      const Spacer(),  
      const FooterMenu(),  
    ],  
  ),  
)
```

Ist ein Benutzer eingeloggt, wird eine Tradingweisheit auf dem Bildschirm ausgegeben. Außerdem werden die Button zum Login und Registrieren ersetzt durch ein *InvisibleWidget*, welches keinen sichtbaren Inhalt darstellt und keinen Platz auf dem Screen einnimmt.

## Loginscreen

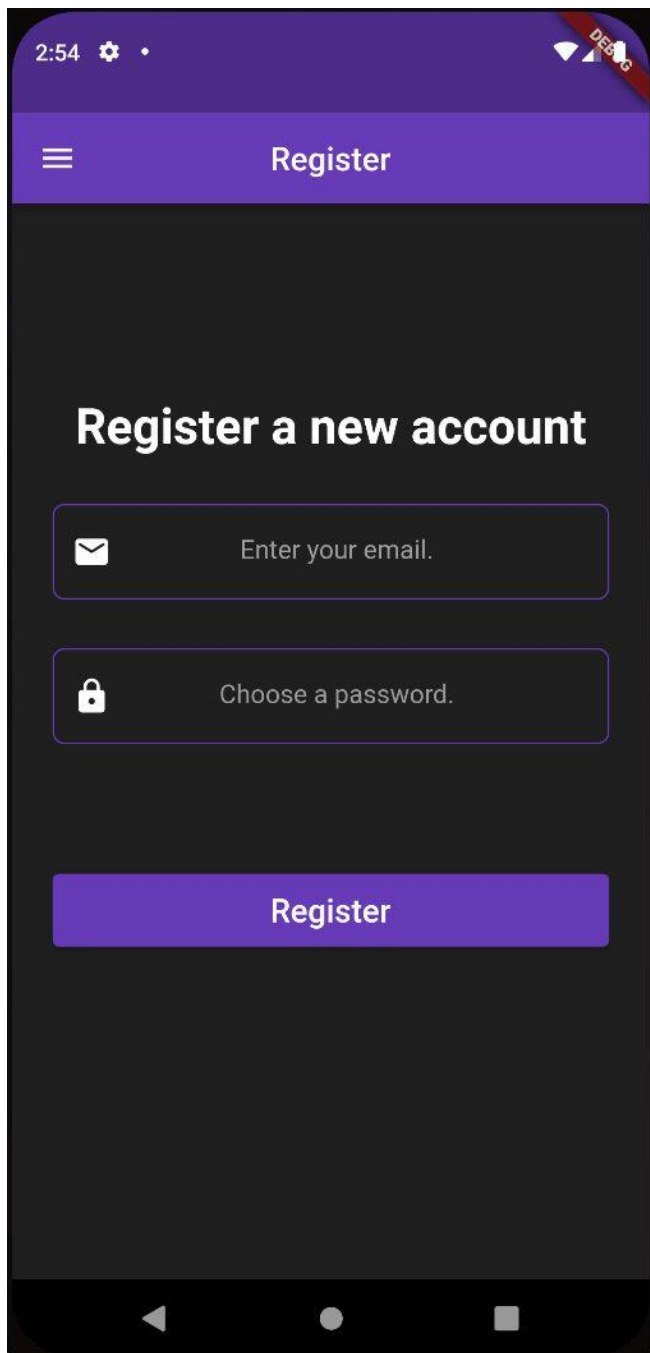


Der *Loginscreen* besteht aus einem *Textwidget*, zwei Eingabefeldern, einem Login-Button und von einem *GestureDetector* ummantelten Text, welcher nach einem Tap auf das Registrieren-Formular weiterleitet.

Nach erfolgreichem Login in einen via Google Firebase registrierten Account erfolgt eine Weiterleitung auf den *Coinlistscreen*.

```
ontapp: () async {  
  if (formkey.currentState!.validate()) {  
    setState(() {  
      isloading = true;  
    });  
    try {  
      await _auth.signInWithEmailAndPassword(  
        email: email, password: password);  
  
      await Navigator.of(context).push(  
        MaterialPageRoute(  
          builder: (context) => const Coinlist(),  
        ),  
      );  
    }  
  }  
}
```

## Registerscreen

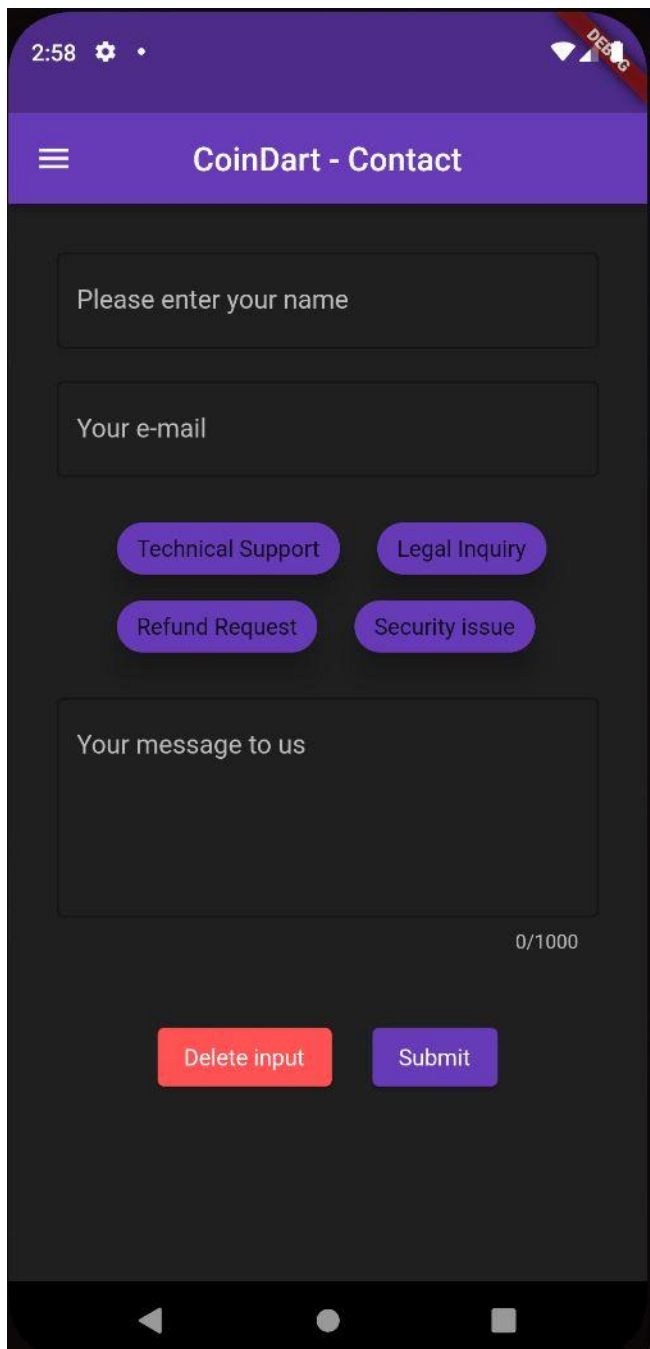


Der Registerscreen besteht aus einem *Textwidget*, zwei Eingabefeldern und einem *Register-Button*. Nach einem *Tap* auf den *Register-Button* wird in Google Firebase ein neuer Account angelegt. Die Authentifizierung erfolgt fortan via E-Mail und Passwort.

```
ontapp: () async {  
  if( formKey.currentState!.validate() ) {  
    setState( () {  
      isLoading = true;  
    });  
  
    try {  
      await _auth.createUserWithEmailAndPassword(email: email, password:  
password);  
    }  
  }  
}
```



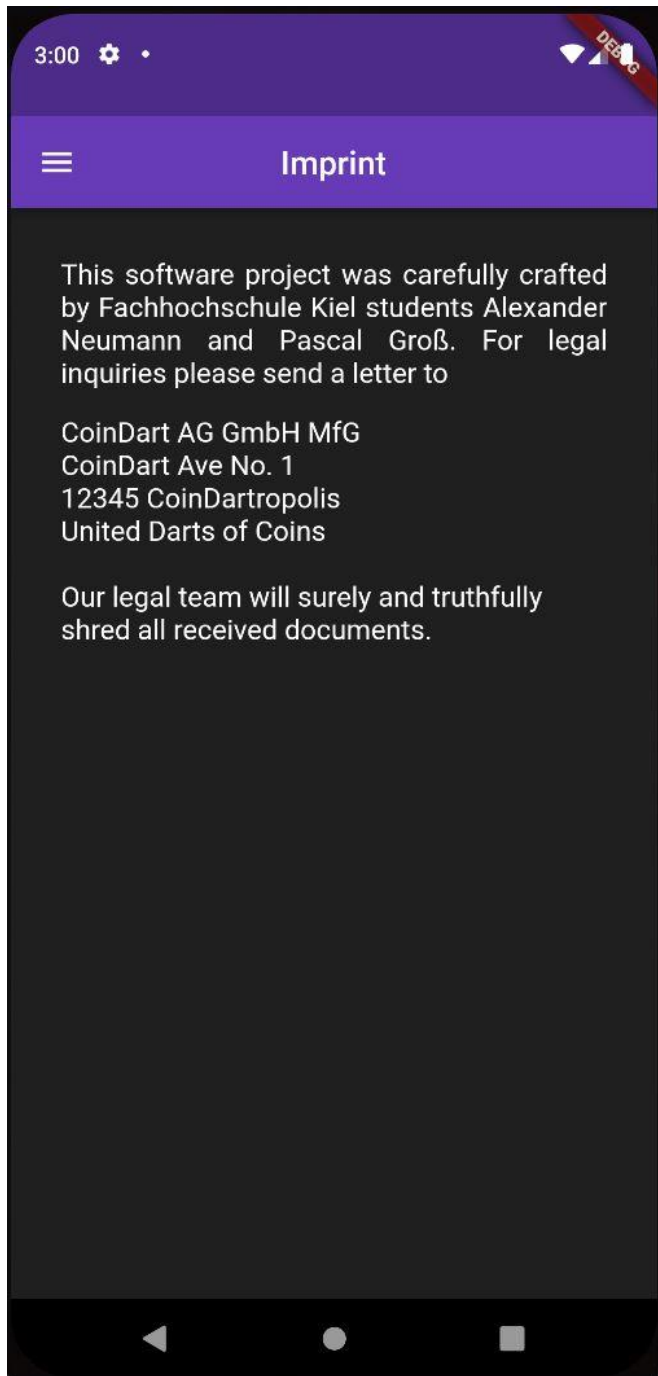
## Contactscreens



The screenshot shows a mobile application interface for 'CoinDart - Contact'. The status bar at the top displays the time 2:58, a settings icon, and a 'Debug' label. The app's title bar is purple with a hamburger menu icon and the text 'CoinDart - Contact'. The main content area has a dark background and contains the following elements: a text input field with the placeholder 'Please enter your name', another text input field with the placeholder 'Your e-mail', four purple buttons for 'Technical Support', 'Legal Inquiry', 'Refund Request', and 'Security issue', a large text area for 'Your message to us' with a '0/1000' character count, and two bottom buttons: a red 'Delete input' button and a purple 'Submit' button. The bottom of the screen shows the standard Android navigation bar.

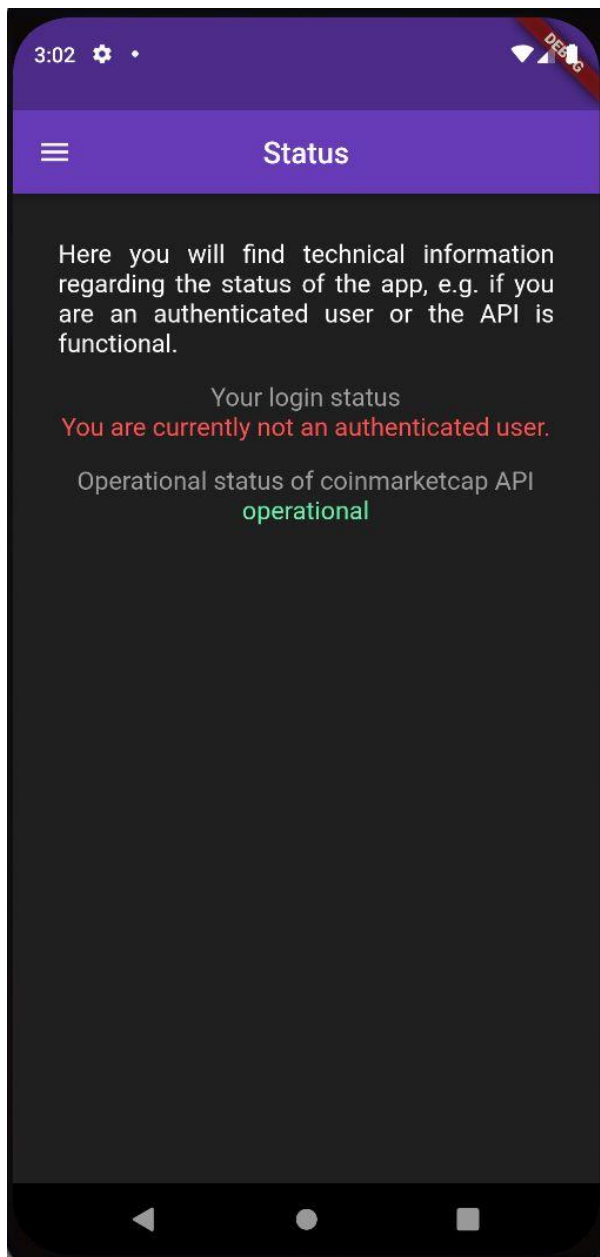
Der *Contactscreens* enthält ein Kontaktformular. Der Benutzer kann einen Namen eingeben, eine E-Mail-Adresse, ein Themenbereich auswählen sowie eine Nachricht hinterlassen. Unter dem Kontaktformular befinden sich die Button "*Delete Input*", um alle Eingaben aus dem Formular zu löschen, und "*Submit*", um die Formulardaten an Google Firebase zu senden.

## Imprintscreen



Auf dem *Imprintscreen* werden *TextWidgets* angezeigt, welche Informationen enthalten über uns Entwickler dieser Anwendung, eine Anschrift und einen rechtlichen Hinweis.

## Statusscreen



Der Statusscreen enthält Informationen über den Login-Status des Benutzers. Es wird auf dem Bildschirm angezeigt, ob ein Benutzer eingeloggt ist oder nicht. Darunter wird der Status der Coinmarketcap-API ausgegeben.

Der Status des Benutzers wird erfasst mit:

```
User? firebaseUser = FirebaseAuth.instance.currentUser;
```

Der Status der Coinmarketcap-API wird ermittelt durch ein Future, welches asynchron einen Request abschickt:

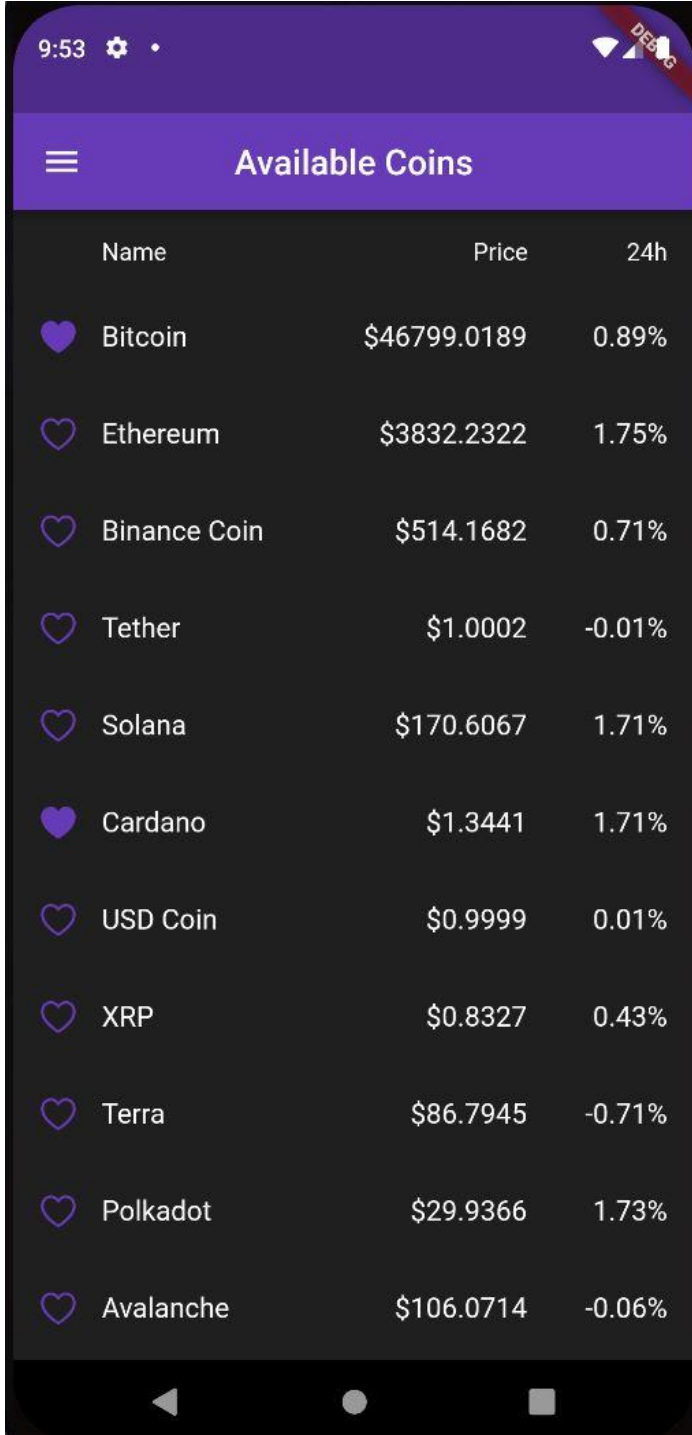
```
const url = 'https://pro-api.coinmarketcap.com/v1/cryptocurrency/listings/latest?limit=1';
```

Daraufhin wird der HTTP-Statuscode auf den Wert 200 überprüft. Diese Information wird im Rahmen eines FutureBuilders abgefragt:

```
FutureBuilder<bool>(  
  future: futureData,  
  builder: (context, snapshot) {  
    if (snapshot.hasData) {  
      return Text(snapshot.data == true ? "operational" : "not  
operational",  
        textAlign: TextAlign.justify,  
        style: TextStyle(  
          color: snapshot.data == true ? Colors.greenAccent :  
Colors.redAccent,  
          fontSize: 17  
        )  
      );  
    } else if (snapshot.hasError) {  
      return const Text("Unable to confirm API status");  
    }  
    return const Center(child: CircularProgressIndicator());  
  })
```

Nur für authentifizierte Benutzer

Coinlistscreen



	Name	Price	24h
♥	Bitcoin	\$46799.0189	0.89%
♥	Ethereum	\$3832.2322	1.75%
♥	Binance Coin	\$514.1682	0.71%
♥	Tether	\$1.0002	-0.01%
♥	Solana	\$170.6067	1.71%
♥	Cardano	\$1.3441	1.71%
♥	USD Coin	\$0.9999	0.01%
♥	XRP	\$0.8327	0.43%
♥	Terra	\$86.7945	-0.71%
♥	Polkadot	\$29.9366	1.73%
♥	Avalanche	\$106.0714	-0.06%

Nach einem erfolgreichen Login wird der Benutzer weitergeleitet auf den *Coinlistscreen*.

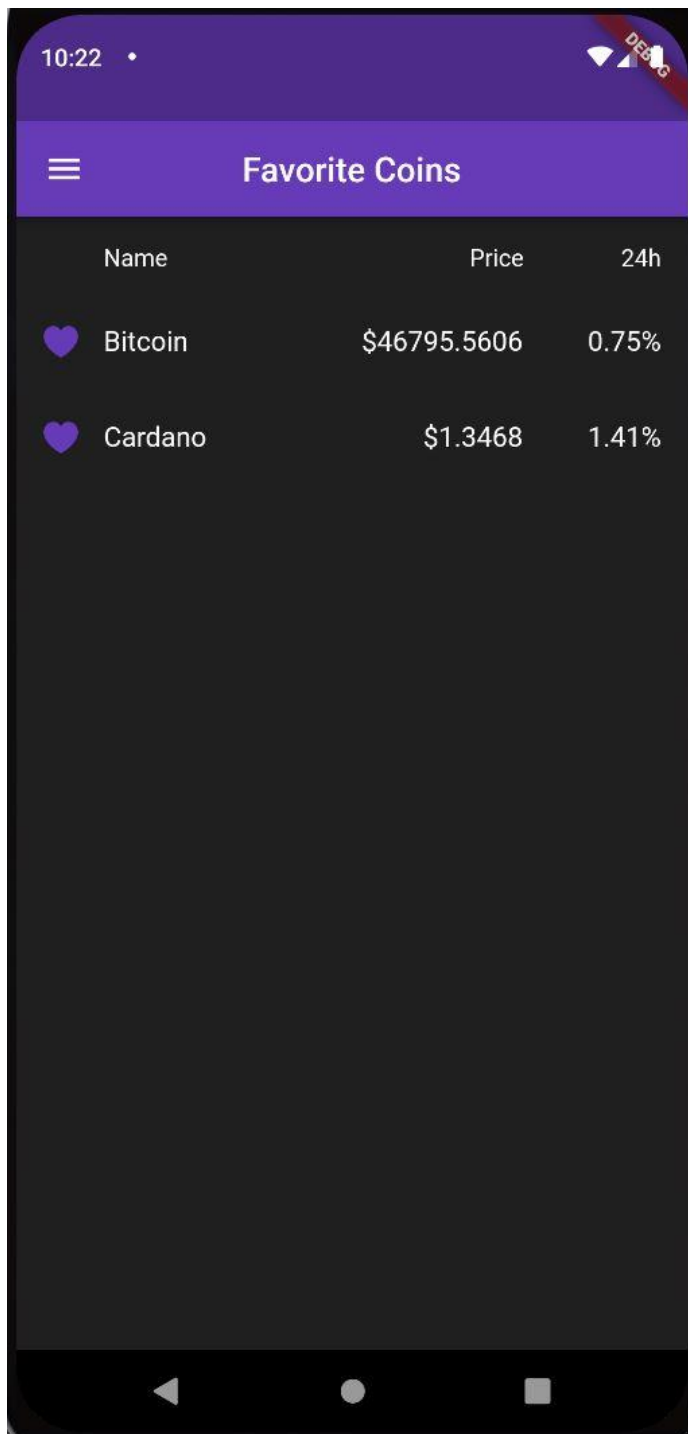
Auf diesem Bildschirm dargestellt wird links ein Favorit-Symbol. Führt ein angemeldeter Benutzer einen Tap auf das Favorit-Symbol aus, wird in der Datenbank gespeichert, dass der Benutzer diese Kryptowährung als Favorit markiert hat.

Im weiteren Bildschirmbereich werden Kryptowährungen aufgelistet, deren jeweils aktueller Preis und die Preisveränderung im Vergleich zum Preis vor 24 Stunden. Diese Daten stammen aus der Coinmarketcap-API.

Führt ein Benutzer einen Tap auf eine Zeile aus, öffnet sich der Detailscreen, auf dem zusätzliche Informationen zu der gewünschten Kryptowährung angezeigt werden.

Eine genauere Beschreibung der Internetkommunikation und API-Calls erfolgt im Abschnitt [API-Calls](#).

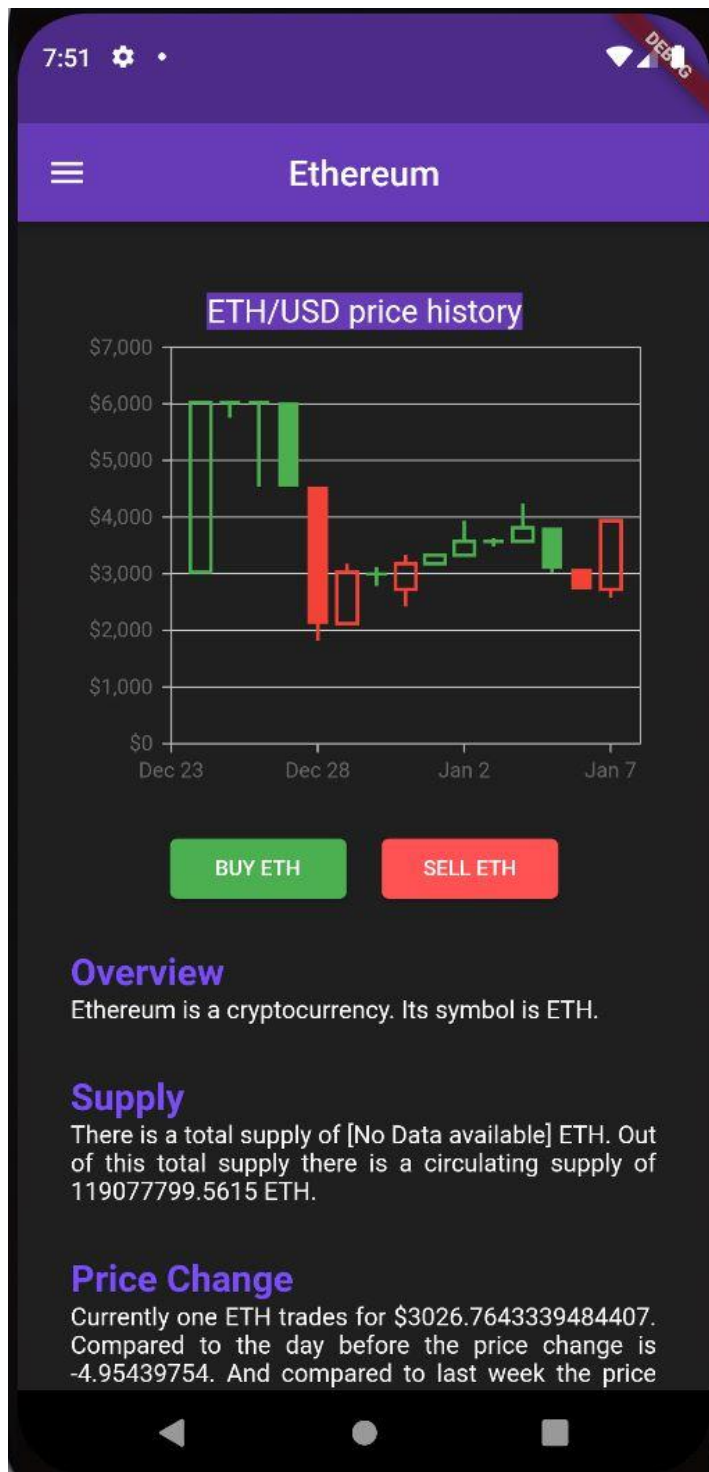
## Favoritescreen



Der *Favoritescreen* ist für eingeloggte Benutzer erreichbar über das *Burger-Menu*.

Ruft ein Benutzer diesen Screen auf, werden dem jeweiligen Benutzer dessen als Favorit markierte Kryptowährungen angezeigt.

## Detailscreen



Führt ein Benutzer einen Tap auf einen Eintrag in der Liste "Available Coins" aus, wird er weitergeleitet auf den Detailscreen.

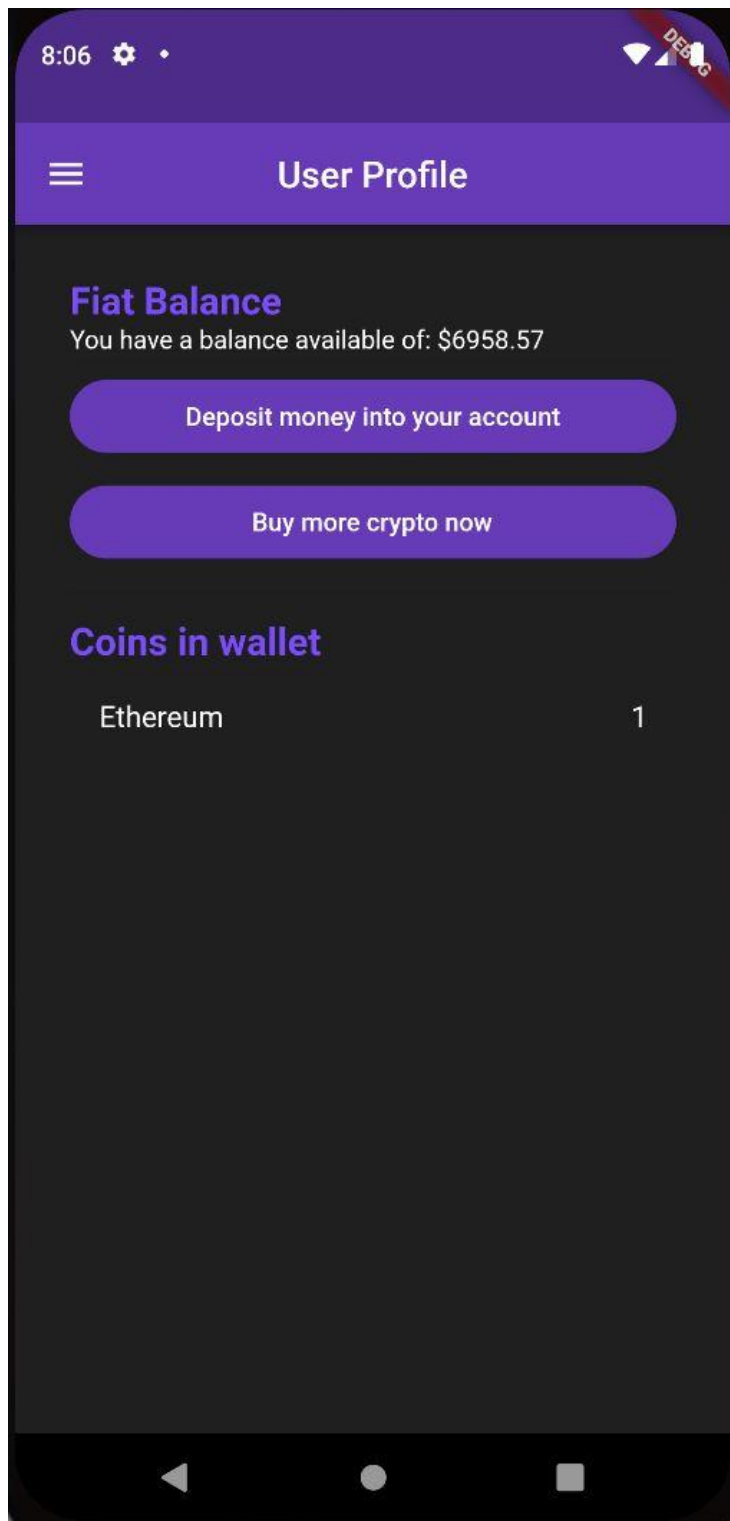
Der Detailscreen enthält einen Preischart, genauer: einen Candle-Chart wie es beim Handeln üblich ist.

Unterhalb des Candle-Charts kann ein Benutzer die jeweilige Kryptowährung kaufen oder seinen Bestand verkaufen.

Unterhalb der beiden Buttons werden fünf Abschnitte dargestellt, die zusätzliche Marktdaten der gewählten Kryptowährung enthalten.

Steht in einem der Abschnitte "[No Data available]", stellt die CMC-API keine Daten bereit. Das ist in der Regel beim Wert des maximalen Supplies einer Kryptowährung der Fall. Es gibt theoretisch unendlich viele Ethereum, deswegen gibt es keine Obergrenze beim maximalen Supply.

## Profilscreen



In dem Benutzerprofil wird im oberen Bildschirmbereich die aktuell für den Benutzer verfügbare Währung angezeigt.

Darunter befinden sich zwei Button, wobei ein Tap auf einen der Button dazu führt, dass der Benutzer Geld einzahlen kann in sein Konto oder weitere Kryptowährungen kaufen kann.

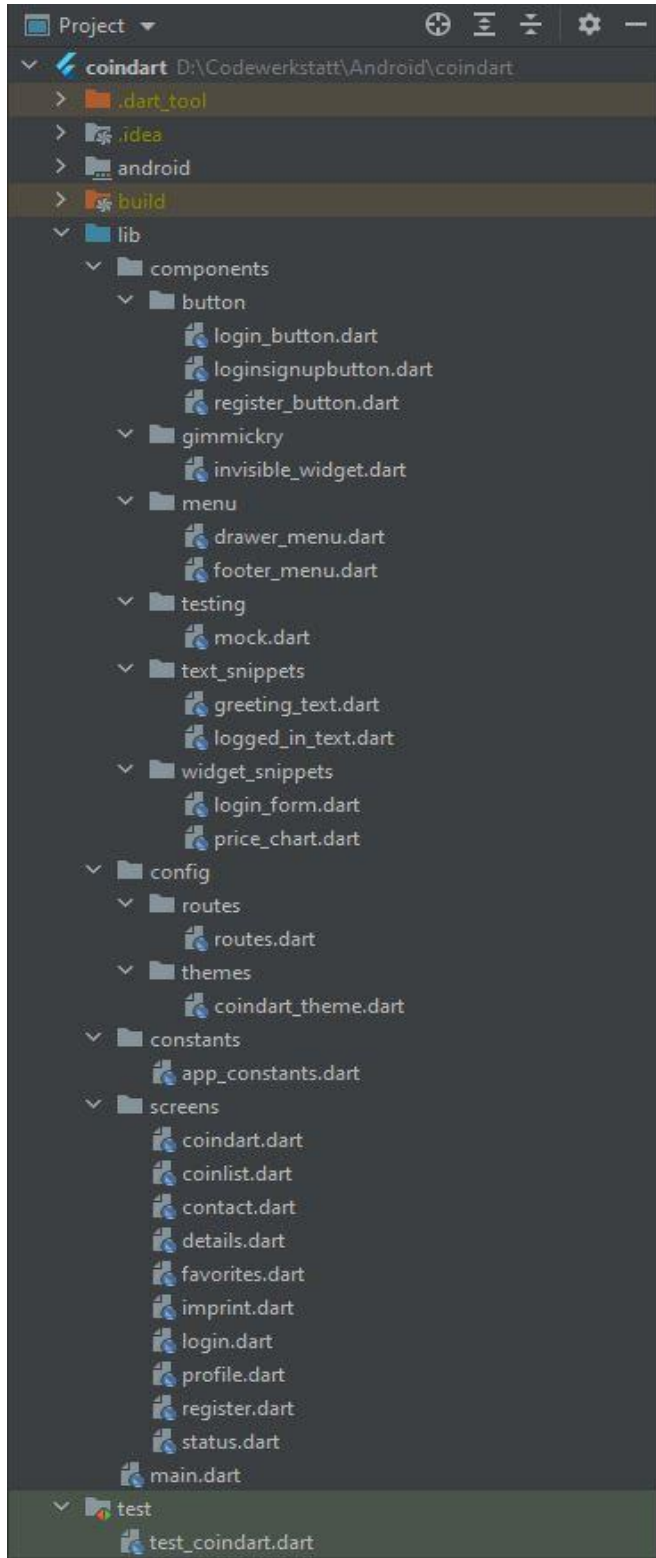
Tapped der Benutzer "Buy more crypto now", wird er weitergeleitet zur Liste der zum Trading verfügbaren Coins.

In dem Abschnitt "Coins in wallet" wird eine Liste angezeigt, die aus den Coins besteht, welche ein Benutzer erworben hat. Hat ein Benutzer keine Coins erworben, weist ein Text darauf hin, dass keine Kryptowährungen besessen werden und fortan gehandelt werden können.



## Projektdetails

### Ordnerstruktur



Wir haben unser Projekt so strukturiert, dass wir die *lines of code* pro Datei minimal halten können.

In der Datei *main.dart* verweisen wir auf den *Homescreen coindart.dart*. Die für Benutzer sichtbaren Screens sind abgelegt im gleichnamigen Ordner.

Darüber hinaus gibt es die Ordner "*components*" und "*config*". In erstgenanntem Ordner befinden sich verschiedene Bausteine, Komponenten, welche in den einzelnen Screens verwendet werden: Das Burger-Menü zum Beispiel, welches in der AppBar jedes Screens angezeigt wird.

## Beispielhafte API-Calls

Dynamische Daten werden bezogen von der CoinMarketCap-API (CMC-API):  
<https://coinmarketcap.com/api/documentation/v1/#>

## Statusscreen

Auf dem Statusscreen wird angezeigt, ob die API grundsätzlich erreichbar ist. Dies wird folgendermaßen bewerkstelligt:

```
Future<bool> healthCheck() async {
  // healthCheck to provide unauthenticated user information about
  // operational status of coinmarketcap api
  // API-URL and API-Key
  const url = 'https://pro-
api.coinmarketcap.com/v1/cryptocurrency/listings/latest?limit=1';
  final Map<String, String> tokenData = {
    "X-CMC_PRO_API_KEY": "8836be1d-8855-43d4-8689-3e9f9f0911c7",
  };

  // API-Call
  final response = await http.get(Uri.parse(url), headers: tokenData);

  if( response.statusCode == 200 ) {
    return true;
  } else {
    return false;
  }
}
```

Zunächst werden die Komponenten für den API-Call bestimmt: eine URL und HTTP-Header-Daten. Anschließend wird der eigentliche HTTP-Request durchgeführt und der Statuscode in der Variable `response` gespeichert.

Lautet der Antwortcode 200, ist die API online. In allen anderen Fällen ist das Gerät entweder nicht mit dem Internet verbunden oder die CMC-API ist nicht erreichbar. Die Verarbeitung des Antwortcodes auf dem für den Benutzer sichtbaren Screen erfolgt folgendermaßen:

```
FutureBuilder<bool>({
  future: futureData,
  builder: (context, snapshot) {
    if (snapshot.hasData) {
      return Text(snapshot.data == true ? "operational" : "not
operational",
        textAlign: TextAlign.justify,
        style: TextStyle(
          color: snapshot.data == true ? Colors.greenAccent :
Colors.redAccent,
          fontSize: 17
        )
      );
    } else if(snapshot.hasError) {
      return const Text("Unable to confirm API status");
    }
    return const Center(child: CircularProgressIndicator());
  })// FutureBuilder
```

## Coinlistscreen

Der Coinlistscreen zeigt eine Liste von Kryptowährungen an. Dargestellt werden die folgenden Informationen: Der Name der Kryptowährung, der aktuelle Preis und die Preisveränderung im Vergleich zum Preis vor 24 Stunden. Der Abruf der Daten erfolgt folgendermaßen:

```
Future<List<Coin>> fetchCoin() async {

    // API-URL and API-Key
    const url = 'https://pro-api.coinmarketcap.com/v1/cryptocurrency/listings/latest?limit=20';
    final Map<String, String> tokenData = {
        "X-CMC_PRO_API_KEY": "8836be1d-8855-43d4-8689-3e9f9f0911c7",
    };

    // API-Call
    final response = await http.get(Uri.parse(url), headers: tokenData);

    // Get Favorites from Database
    List<num> favIds = [];
    await favorites.get().then((snapshot) {
        for (var doc in snapshot.docs) {
            favIds.add(int.parse(doc.id));
        }
    });

    if (response.statusCode == 200) {
        // If the server did return a 200 OK response then parse the JSON.
        List<Coin> result = [];
        for( var i = 0 ; i < 20; i++ ) {
            result.add(Coin.fromJson(jsonDecode(response.body) ['data'] [i]));
            if(favIds.contains(result[i].id)) {
                result[i].changeFavorite();
            }
        }
        return result;
    } else {
        // If the server did not return a 200 OK response then throw an
        exception.
        throw Exception('Failed to load Coins');
    }
}
```

## Detailscreen

Der Detailscreen ist insofern besonders, als dass Daten aus zwei API-Endpunkten in einer Klasse gespeichert werden. Die Daten werden folgendermaßen abgefragt:

```
// fetchCoinData() is called with a specific coinId to request coin
// specific data from the CoinMarketCap-API
// data is saved in a Coin object
// param "coinId": is provided by every route that leads to this page
Future<Coin> fetchCoinData(final String coinId) async {

    // API URLs
    // API endpoint "info" provides meta data like a coinname, a coin id
    String urlInfo = 'https://pro-
api.coinmarketcap.com/v1/cryptocurrency/info?id=' + coinId;
    // API endpoint "quotes" provides specific data like the current price of
    // a crypto currency
    String urlQuotes = 'https://pro-
api.coinmarketcap.com/v1/cryptocurrency/quotes/latest?id=' + coinId;

    // API key
    final Map<String, String> tokenData = {
        "X-CMC_PRO_API_KEY": "195a8398-cf16-44bd-8e63-cf59d9670dfa",
    };

    // API-Call
    final responseInfo = await http.get(Uri.parse(urlInfo), headers:
tokenData);
    final responseQuotes = await http.get(Uri.parse(urlQuotes), headers:
tokenData);
```

Unter der Bedingung, dass beide API-Endpunkte den HTTP-Antwortcode 200 ausliefern, wird ein neues Coin-Objekt erzeugt und die Daten werden in die Klassenvariablen geschrieben:

```
return Coin(
    id: jsonInfo['id'],
    name: (jsonInfo['name'] == null) ? noData : jsonInfo['name'],
    symbol: (jsonInfo['symbol'] == null) ? noData : jsonInfo['symbol'],
    logo: (jsonInfo['logo'] == null) ? noData : jsonInfo['logo'],
    website: (jsonInfo['urls']['website'][0] == null) ? noData :
jsonInfo['urls']['website'][0],
    price: (jsonQuotes['quote']['USD']['price'] == null) ? noData :
jsonQuotes['quote']['USD']['price'].toString(),
    circSupply: (jsonQuotes['circulating_supply'] == null) ? noData :
jsonQuotes['circulating_supply'].toString(),
    maxSupply: (jsonQuotes['max_supply'] == null) ? noData :
jsonQuotes['max_supply'].toString(),
    marketCap: (jsonQuotes['quote']['USD']['market_cap'] == null) ? noData
: jsonQuotes['quote']['USD']['market_cap'].toString(),
    changeDay: (jsonQuotes['quote']['USD']['percent_change_24h'] == null) ?
noData : jsonQuotes['quote']['USD']['percent_change_24h'].toString(),
    changeWeek: (jsonQuotes['quote']['USD']['percent_change_7d'] == null) ?
noData : jsonQuotes['quote']['USD']['percent_change_7d'].toString()
);
```

Über einen `FutureBuilder` werden diese Daten zugänglich gemacht, sodass sie unter anderem auf dem Bildschirm ausgegeben werden können. Zum Beispiel:

```
Text(  
  "There is a total supply of " + snapshot.data!.maxSupply + " " +  
  snapshot.data!.symbol + ". Out of this total supply there is a circulating  
  supply of " + snapshot.data!.circSupply + " " + snapshot.data!.symbol +  
  ". ",  
  textAlign: TextAlign.justify,  
)
```