







Programowanie urządzeń mobilnych

Projekt:
Aplikacja do śledzenia kursu kryptowalut

Wykonali:
Aleksander Jewula
Mateusz Jedziniak

Opis Projektu

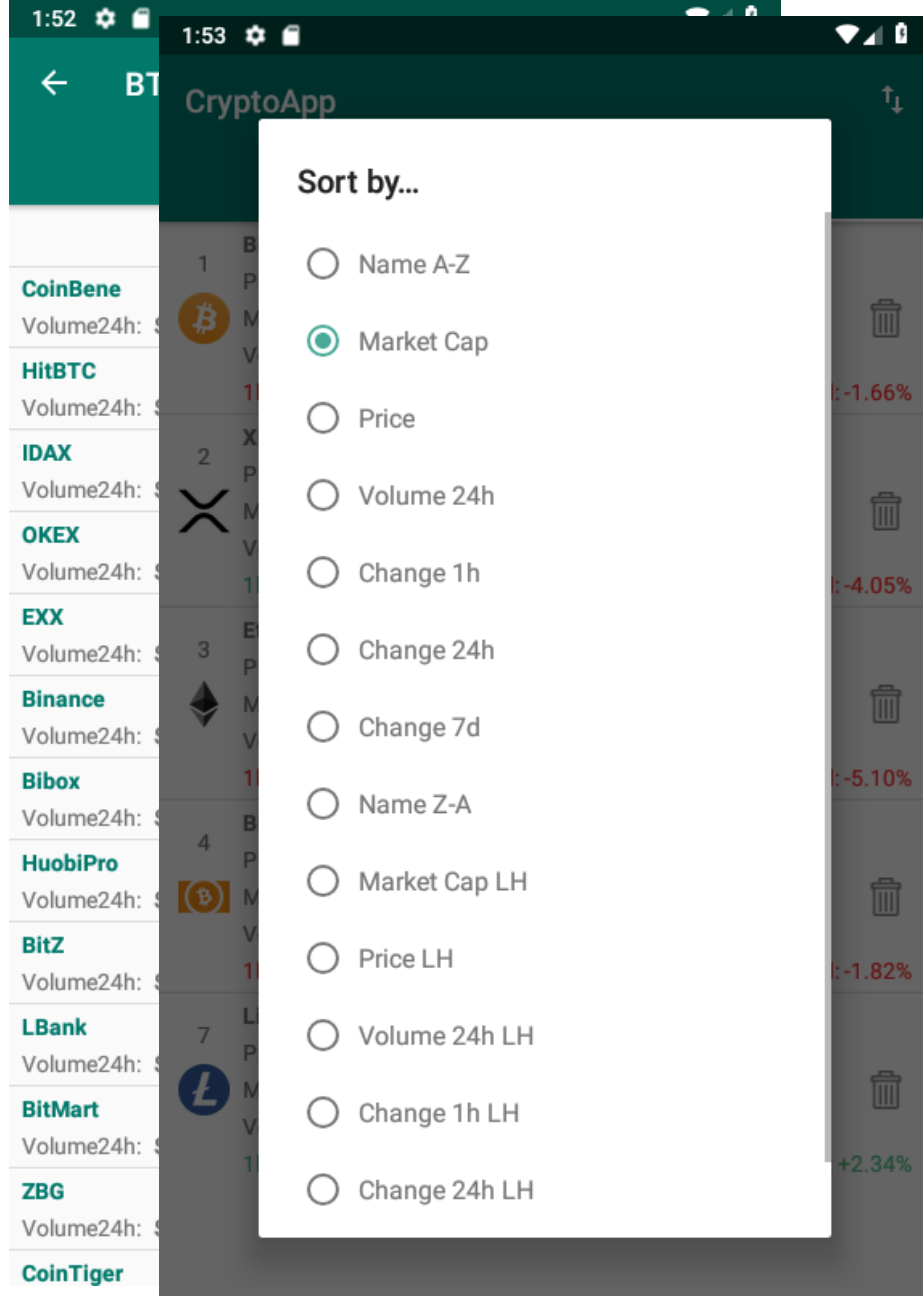
Aplikacja służy do monitorowania kursu wszystkich top 100 kryptowalut z coinmarketcap.com. Posiada możliwość nadawania wybranym walutom statusu ulubionych i kontrolowania ich w jednym miejscu. Dzięki zaimplementowanemu wykresowi każdej kryptowaluty możemy analizować jaki przebieg miała jej cena od chwili jej powstania do chwili obecnej a także zmiany wykresów w przedziałach czasowych. Przy wykresach widnieją informacje o kapitalizacji i wszystkich innych istotnych danych o walucie. O reszcie funkcjonalności poniżej.

CryptoApp		🔍	↑↓
ALL COINS		FAVORITES	
1	Bitcoin (BTC) Price: \$ 3581.46701102  MarketCap: \$ 62,690,757,832.00 Volume 24h: \$ 5,195,153,721.64 1h: -0.01% 24h: -1.18% 7d: -1.66%	★	
2	XRP (XRP) Price: \$ 0.3105045098  MarketCap: \$ 12,781,441,971.00 Volume 24h: \$ 350,838,040.02 1h: +0.34% 24h: -2.50% 7d: -4.05%	★	
3	Ethereum (ETH) Price: \$ 114.947680225  MarketCap: \$ 12,022,209,760.00 Volume 24h: \$ 2,560,955,055.34 1h: -0.15% 24h: -2.06% 7d: -5.10%	★	
4	Bitcoin Cash (BCH) Price: \$ 123.258050159  MarketCap: \$ 2,167,988,926.00 Volume 24h: \$ 180,218,787.16 1h: -0.26% 24h: -3.95% 7d: -1.82%	★	
5	EOS (EOS) Price: \$ 2.3900450372  MarketCap: \$ 2,165,966,646.00 Volume 24h: \$ 685,289,048.88 1h: -0.22% 24h: -2.88% 7d: -0.05%	☆	
6	Tether (USDT) Price: \$ 1.0103718386  MarketCap: \$ 2,038,173,770.00	☆	











W zakładce ALL COINS która wita nas po odpaleniu aplikacji widnieje lista Kryptowalut posortowana od tej z największą kapitalizacją, co można zmienić pod ikoną (2 strzałek) nadając inne filtry. Dodatkowo pod każdą walutą posiadamy krótkie informacje tj. Aktualną Cenę, sumie kapitalizacji, Volumen w ostatnich 24h a także procentowo jak zmienił się jej kurs w okresie 1h/24h/7dni. Ikona gwiazdki do której zaimplementowana jest krótka animacja dodaje daną kryptowalutę do zakładki FAVORITES gdzie w prosty sposób można zarządzać ulubionymi pozycjami w przejrzysty sposób.



Po kliknięciu w daną kryptowalutę dostajemy więcej informacji na jej temat łącznie z wykresem który możemy skalować w czasie 24 godzin, 7 dni, 1 miesiąca, 3 miesięcy, 1 roku lub całego okresu aktywności waluty. Dodatkowo posiadamy możliwość zmiany kursu FIAT po prawej stronie w liście wysuwanej. Dostęp z tego miejsca do zakładki Markets o której poniżej. Z informacji o kryptowalucie poza wcześniej wymienianymi z poziomu strony głównej można znaleźć: Cenę w BTC, Ilość wszystkich tokenów i ilość wszystkich dostępnych tokenów na rynku.



Zakładka MARKETS służy do wyboru marketu/gieldy z jakiej chcemy pobierać kurs i brać go pod uwagę. Przy nazwie giełdy posiadamy informację o jej Volumenie w ostatnich 24h, kurs jaki reprezentują na daną chwilę a także procentowo jak uległ on zmianie w tym okresie czasu.

CryptoApp			
ALL COINS		FAVORITES	
1	Bitcoin (BTC) Price: \$ 3581.46701102  MarketCap: \$ 62,690,757,832.00 Volume 24h: \$ 5,195,153,721.64 1h: -0.01% 24h: -1.18% 7d: -1.66%		
2	XRP (XRP) Price: \$ 0.3105045098  MarketCap: \$ 12,781,441,971.00 Volume 24h: \$ 350,838,040.02 1h: +0.34% 24h: -2.50% 7d: -4.05%		
3	Ethereum (ETH) Price: \$ 114.947680225  MarketCap: \$ 12,022,209,760.00 Volume 24h: \$ 2,560,955,055.34 1h: -0.15% 24h: -2.06% 7d: -5.10%		
4	Bitcoin Cash (BCH) Price: \$ 123.258050159  MarketCap: \$ 2,167,988,926.00 Volume 24h: \$ 180,218,787.16 1h: -0.26% 24h: -3.95% 7d: -1.82%		
7	Litecoin (LTC) Price: \$ 32.6267266293  MarketCap: \$ 1,964,303,501.00 Volume 24h: \$ 621,532,113.85 1h: +0.02% 24h: -3.26% 7d: +2.34%		

Sortowanie ekranu głównego i dostępne jego opcje. Dopisek „LH” oznacza sortowanie odwrotne dając możliwość prostego wyświetlania największych i jednym kliknięciem najmniejszych zmian na rynku w danym zakresie.

Zakładka ulubionych walut które można w prosty sposób usunąć kliknięciem w ikonkę kosza. Z tego miejsca mamy pełną funkcjonalność bez żadnych ograniczeń, jedyną zmianą jest brak nieinteresujących pozycji. Daje nam to możliwość prostego śledzenia walut z dalszych miejsc MarketCap (np.85) bez zbędnego przewijania do dołu listy.

Opis kodu źródłowego

Użyte technologie:

EasyRest: ta biblioteka jest szeroko stosowana do wszystkich połączeń sieciowych w aplikacji.

Dbą o buforowanie żądań, wielowątkowość i synchronizację.

Material-dialogs: Biblioteka używana do wyświetlania okna dialogowego sortowania.

Customtabs: Biblioteka używana w przeglądarce internetowej z integracjami Chrome

ToggleButtonGroup: biblioteka używana dla przycisków, które pozwalają użytkownikom przełączać zakres dat na wykresie.

MaterialFavoriteButton: Biblioteka używana do animowanych ikon w aplikacji.

MPAndroidChart: Służy do wyświetlania wykresu ceny w czasie.

GSON: Biblioteka służąca do serializowania tekstu z bazy danych do rzeczywistych obiektów Java i odwrotnie.

```
public class CoinMarketCapService {  
  
    public static final String COIN_MARKETCAP_ALL_COINS_URL = "https://api.coinmarketcap.com/v1/ticker/?limit=0";  
    public static final String COIN_MARKETCAP_CHART_URL_ALL_DATA = "https://graphs2.coinmarketcap.com/currencies/%s/";  
    public static final String COIN_MARKETCAP_CHART_URL_WINDOW = "https://graphs2.coinmarketcap.com/currencies/%s/%s/%s/";  
    public static final String COIN_MARKETCAP_QUICK_SEARCH_URL = "https://s2.coinmarketcap.com/generated/search/quick_search.json";  
  
    public static void getAllCoins(Context context, afterTaskCompletion<CMCCoin[]> taskCompletion, afterTaskFailure failure, boolean async) {...}  
  
    public static void getCMCChartData(Context context, String coinID, afterTaskCompletion<CMCChartData> taskCompletion, afterTaskFailure failure, boolean async) {...}  
  
    public static void getCMCQuickSearch(Context context, afterTaskCompletion<CMCQuickSearch[]> taskCompletion, afterTaskFailure failure, boolean async) {...}  
}
```

Najważniejszą informacją jaką mamy w naszym programie są to wartości kryptowalut. Napisany service pomaga nam pobrać dane.

```
1 package com.cryptoapp.chuddyni.cryptoapp.models;
2
3 import com.fasterxml.jackson.annotation.JsonProperty;
4
5 public class MarketNode {
6     @JsonProperty("MARKET")
7     private String market;
8     @JsonProperty("FROMSYMBOL")
9     private String fromSymbol;
10    @JsonProperty("TOSYMBOL")
11    private String toSymbol;
12    @JsonProperty("PRICE")
13    private float price;
14    @JsonProperty("VOLUME24HOUR")
15    private float volume24h;
16    @JsonProperty("CHANGEPCT24HOUR")
17    private float changePct24h;
18    @JsonProperty("CHANGE24HOUR")
19    private float change24h;
20
21    public String getMarket() {
22        return market;
23    }
24
25    public String getFromSymbol() {
26        return fromSymbol;
27    }
28
29    public String getToSymbol() {
30        return toSymbol;
31    }
32
33    public float getPrice() {
34        return price;
35    }
36
37    public float getVolume24h() {
38        return volume24h;
39    }
40
41    public float getChangePct24h() {
42        return changePct24h;
43    }
44
45    public float getChange24h() {
46        return change24h;
47    }
48
49    public void setMarket(String market) {
50        this.market = market;
```

Pobrane dane pozwalają nam na operację z nimi. Gettery, setery, adnotacje.

```

109 Bundle args = new Bundle();
110 args.putString(ARG_SYMBOL, symbol);
111 args.putString(ARG_ID, id);
112 fragment.setArguments(args);
113 return fragment;
114 }
115
116 public void setColors(float percentChange) {
117     if (percentChange >= 0) {
118         chartFillColor = ResourcesCompat.getColor(getActivity().getResources(), R.color.materialLightGreen, null);
119         chartBorderColor = ResourcesCompat.getColor(getActivity().getResources(), R.color.darkGreen, null);
120         percentageColor = ResourcesCompat.getColor(getActivity().getResources(), R.color.percentPositiveGreen, null);
121     }
122     else {
123         chartFillColor = ResourcesCompat.getColor(getActivity().getResources(), R.color.materialLightRed, null);
124         chartBorderColor = ResourcesCompat.getColor(getActivity().getResources(), R.color.darkRed, null);
125         percentageColor = ResourcesCompat.getColor(getActivity().getResources(), R.color.percentNegativeRed, null);
126     }
127 }
128
129 public void setUpChart() {
130     XAxis xAxis = lineChart.getXAxis();
131     xAxis.setDrawAxisLine(true);
132     xAxis.setPosition(XAxis.XAxisPosition.BOTTOM_INSIDE);
133     xAxis.setAvoidFirstLastClipping(true);
134     lineChart.getAxisLeft().setEnabled(true);
135     lineChart.getAxisLeft().setDrawGridLines(false);
136     lineChart.getAxis().setDrawGridLines(false);
137     lineChart.getAxisRight().setEnabled(false);
138     lineChart.getLegend().setEnabled(false);
139     lineChart.setDoubleTapToZoomEnabled(false);
140     lineChart.setScaleEnabled(false);
141     lineChart.setDescription("");
142     lineChart.setContentDescription("");
143     lineChart.setData(getString(R.string.noChartDataString));
144     lineChart.setDataTextColor(R.color.darkRed);
145     lineChart.setOnChartValueSelectedListener(this);
146     lineChart.setOnChartGestureListener(new OnChartGestureListener() {
147         @Override
148         public void onChartGestureStart(MotionEvent me, ChartTouchListener.ChartGesture lastPerformedGesture) {
149             YAxis yAxis = lineChart.getAxisLeft();
150             if (me.getX() > yAxis.getLongestLabel().length() * yAxis.getTextSize() &&
151                 me.getX() < displayWidth - lineChart.getViewPortHandler().offsetRight()) {
152                 viewPager.setPagingEnabled(false);
153                 nestedScrollView.setScrollingEnabled(false);
154             }
155         }
156     });
157
158     @Override
159     public void onChartGestureEnd(MotionEvent me, ChartTouchListener.ChartGesture lastPerformedGesture) {

```

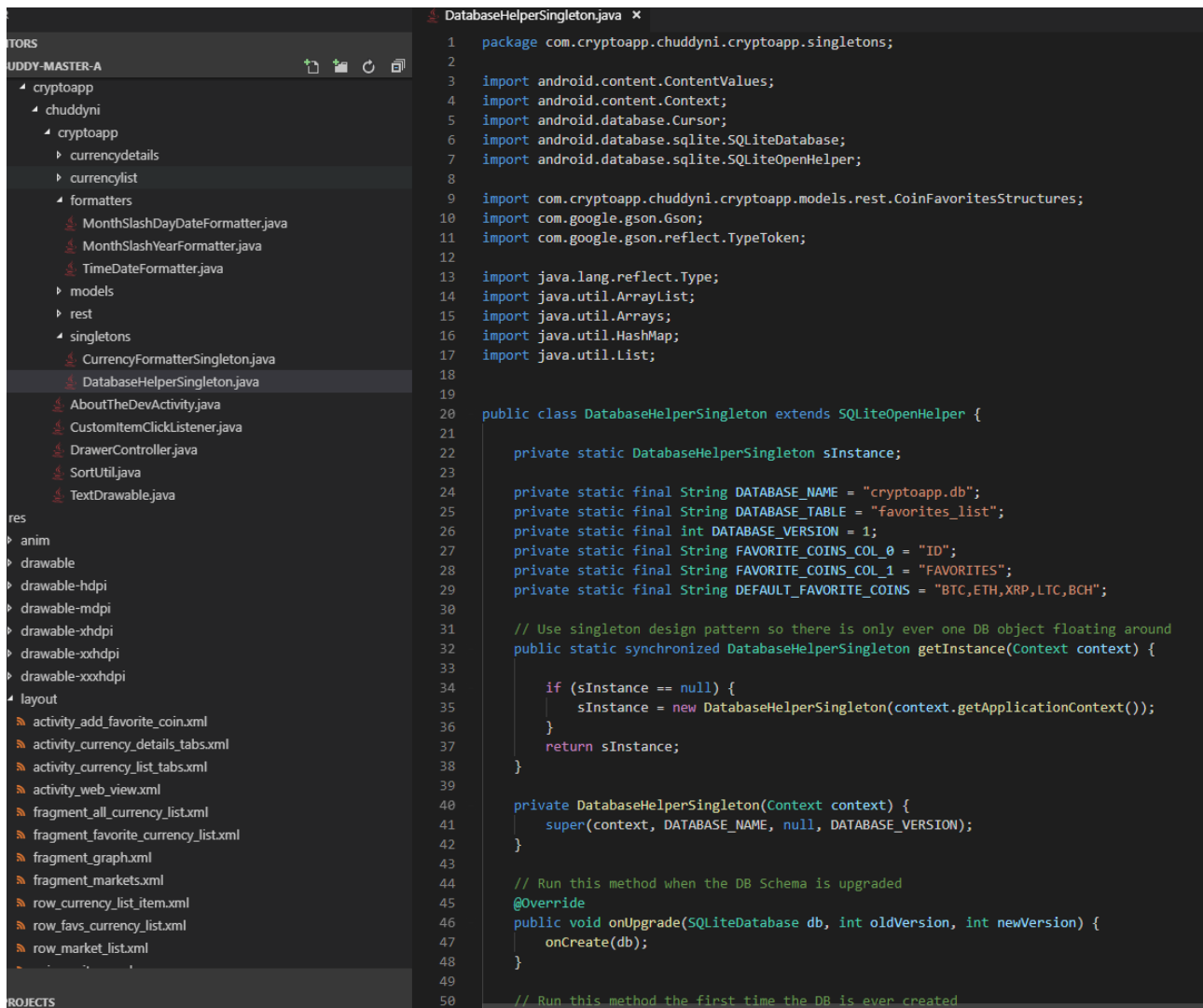
Klasa odpowiedzialna za odpowiednie formatowanie informacji. Kolor czerwony dla kursu waluty który jest poniżej 0, kurs sprzed dnia, miesiąca które są pobrane z API oraz reszta informacji w widoku aplikacji.

```

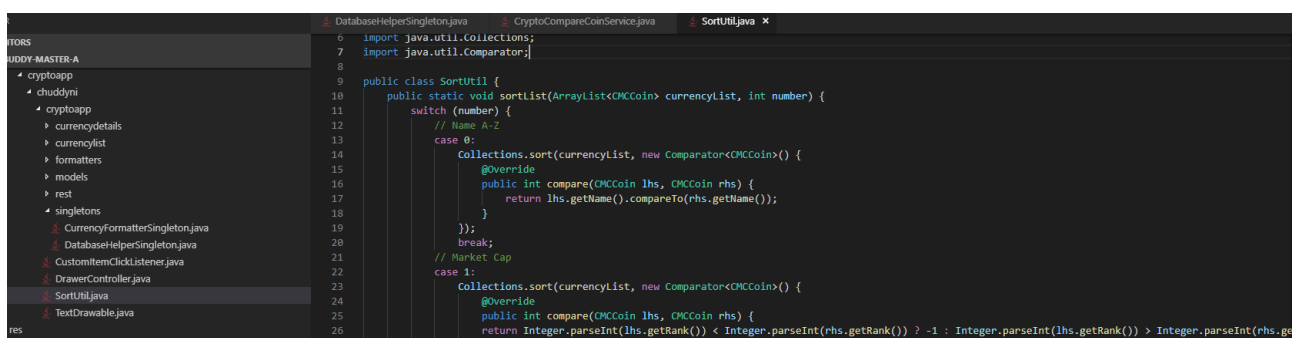
58 public void setFavoriteButtonClickListener(final AllCurrencyListAdapter.ViewHolder holder, final int position) {
59     holder.favButton.setOnClickListener(new View.OnClickListener() {
60         @Override
61         public void onClick(View v) {
62             CoinFavoritesStructures favs = dbRef.get().getFavorites();
63             CMCCoin item = currencyList.get(position);
64             if (favs.favoritesMap.get(item.getSymbol()) == null) { // Coin is not a favorite yet. Add it.
65                 favs.favoritesMap.put(item.getSymbol(), item.getSymbol());
66                 favs.favoriteList.add(item.getSymbol());
67                 holder.favButton.setFavorite(true, true);
68                 favsUpdateCallbackRef.get().addFavorite(item);
69             } else { // Coin is already a favorite, remove it
70                 favs.favoritesMap.remove(item.getSymbol());
71                 favs.favoriteList.remove(item.getSymbol());
72                 holder.favButton.setFavorite(false, true);
73                 favsUpdateCallbackRef.get().removeFavorite(item);
74             }
75             dbRef.get().saveCoinFavorites(favs);
76         }
77     });
78 }

```

W naszej aplikacji jest opcja zaznaczenia przy danej walucie opcji „Favorites”. Pozwala ona na uporządkowanie ulubionych walut w jedno miejsce.



Zgodnie z podstawowym wzorcem projektowym jakim jest Singleton, utworzyliśmy klasę pomocniczą do naszej opcji Favorites. Tworzy ona bazę i zapisuje w niej informację o zmianie ulubionej waluty oraz usuwa ją po interakcji użytkownika.



Za sortowanie danych po wybraniu opcji odpowiada klasa SortUtil. Po wybraniu odpowiedniego działania, kolekcja jest sortowana.