|  |
| --- |
| 2023-2024 1ste semester |
| Mobiele Apps 2: verslag project |
| Wijnhuis Ronny |

|  |
| --- |
| Calvin Liaci  3ICT |

Inhoudsopgave

[Verslag project: Wijnhuis Ronny 3](#_Toc154740626)

[1. Korte toelichting app vanuit het oogpunt van de gebruiker 3](#_Toc154740627)

[2. Icon en Splash Screen 4](#_Toc154740628)

[2.1 Visuele weergave en code 4](#_Toc154740629)

[2.1.1 Icon 4](#_Toc154740630)

[2.1.2 Splash screen 4](#_Toc154740631)

[2.2 Uitleg 4](#_Toc154740632)

[3. Navigatie 5](#_Toc154740633)

[3.1 Visuele weergave en code 5](#_Toc154740634)

[3.2 Uitleg 5](#_Toc154740635)

[4. Startpagina 6](#_Toc154740636)

[4.1 Visuele weergave 6](#_Toc154740637)

[4.2 HomePagePhotosRepository 7](#_Toc154740638)

[4.2.1 Code 7](#_Toc154740639)

[4.2.2 Uitleg code 7](#_Toc154740640)

[4.3 HomepageFragment 8](#_Toc154740641)

[4.3.1 Code 8](#_Toc154740642)

[4.3.2 Uitleg code 8](#_Toc154740643)

[4.4 HomepageFragmentViewModel 9](#_Toc154740644)

[4.4.1 Code 9](#_Toc154740645)

[4.4.2 Uitleg code 9](#_Toc154740646)

[4.5 Fragment\_homepage.xml 10](#_Toc154740647)

[4.5.1 Code 10](#_Toc154740648)

[4.5.2 Uitleg code 11](#_Toc154740649)

[5. Wijnen pagina 12](#_Toc154740650)

[5.1 Wijncategorieën pagina 12](#_Toc154740651)

[5.1.1 Visuele weergave 12](#_Toc154740652)

[5.1.2 WinesCategoryPhotosRepository 13](#_Toc154740653)

[5.1.2.1 Code 13](#_Toc154740654)

[5.1.2.2 Uitleg code 13](#_Toc154740655)

[5.1.3 WinesCategoryFragment 14](#_Toc154740656)

[5.1.3.1 Code 14](#_Toc154740657)

# Verslag project: Wijnhuis Ronny

## Korte toelichting app vanuit het oogpunt van de gebruiker

Welkom bij de Wijnhuis Ronny app, waar passie, wijn en kwaliteit samenkomen! Op onze startpagina wordt je begroet met onze krachtige slogan, die de essentie van ons wijnhuis weergeeft. Daaronder vind je waardevolle informatie over wie we zijn en wat onze passie voor wijn inhoudt.

Wanneer je de ontdekkingsreis door ons wijnassortiment begint, kom je terecht op de wijnenpagina. Hier word je verwelkomd door een uitgebreide selectie wijncategorieën, variërend van verfijnde witte wijnen tot rijke rode en verfrissende rosé wijnen. Na het maken van je keuze navigeer je naar de specifieke pagina van de geselecteerde wijnsoort en ontdek je een lijst met heerlijke wijnen die je direct aan je winkelwagen kunt toevoegen.

Bij de winkelwagen aangekomen, krijg je een overzicht van al de wijnen die je hebt geselecteerd, inclusief de hoeveelheden en prijzen. Hier heb je de mogelijkheid om je bestelling te plaatsen met een eenvoudige druk op de "Bestelling plaatsen"-knop. Dit opent een nieuwe pagina waar je persoonlijke gegevens kunt invoeren om de checkout te voltooien.

Eenmaal je gegevens zijn ingevoerd, klik je op "Bestelling afronden" en word je naadloos doorgestuurd naar je mail-app. Hier staat een automatisch gegenereerde e-mail klaar met al je gegevens en de details van je bestelde wijnen. Door op "Verzenden" te drukken, bevestig je officieel je wijnbestelling.

Naast het winkelen bieden we ook de mogelijkheid om je in te schrijven voor boeiende degustaties. Navigeer eenvoudig naar de degustatiepagina in de navigatiebalk, ontdek de komende evenementen en schrijf je in met slechts één klik. Hier ontvang je ook een automatisch opgestelde e-mail om je inschrijving te bevestigen en jezelf voor te bereiden op een smaakvolle ervaring.

## Icon en Splash Screen

### Visuele weergave en code

#### Icon

Afbeelding met tekst, schermopname, Lettertype, logo

Automatisch gegenereerde beschrijvingAfbeelding met elektronica, tekst, schermopname, multimedia

Automatisch gegenereerde beschrijving

#### Splash screen

Afbeelding met tekst, schermopname, Lettertype, ontwerp

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, visitekaartje, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met stoel, tekening, schets, ontwerp

Automatisch gegenereerde beschrijving

### Uitleg

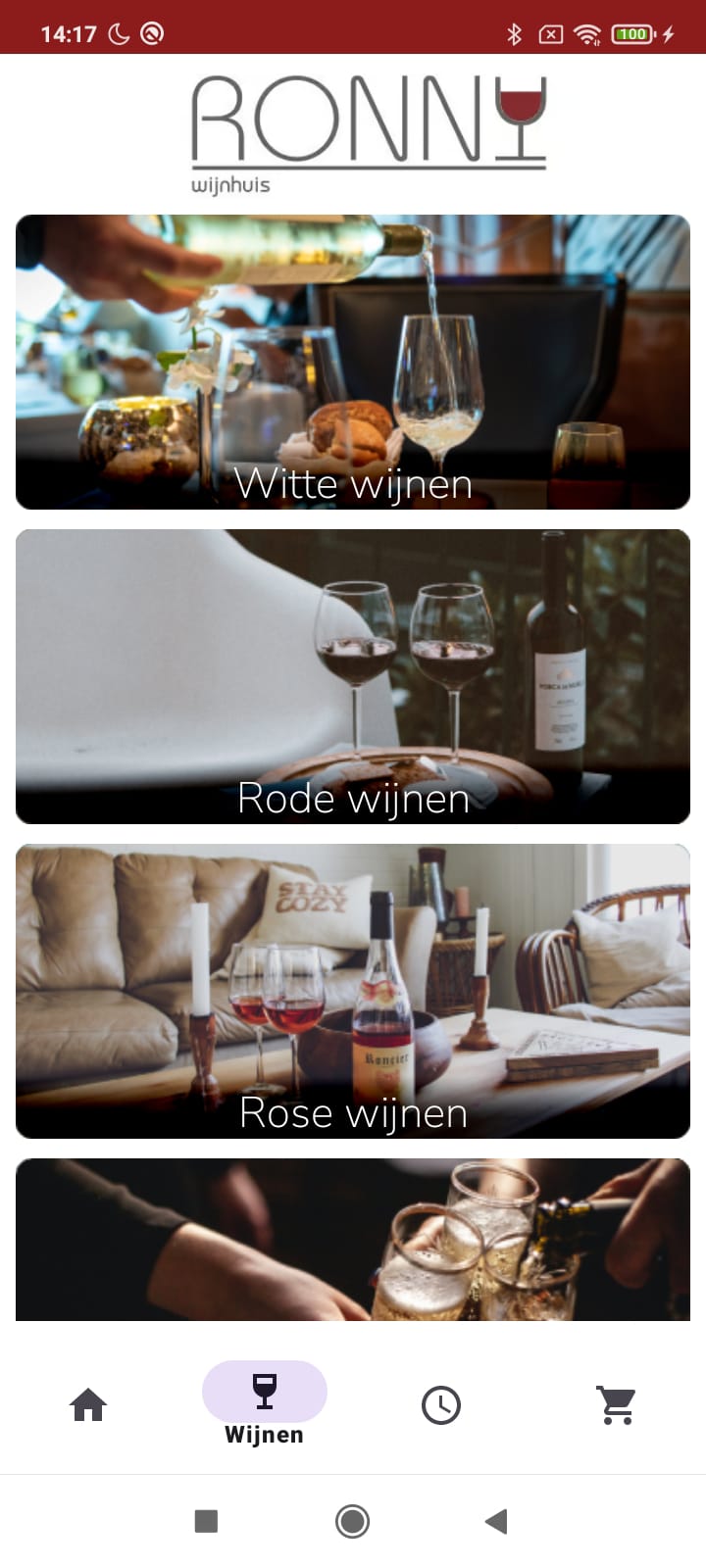
Om de Icon en Splash screen aan te passen in Android Studio, moest ik een paar wijzigingen aanbrengen in het AndroidManifest.xml-bestand. Voor het pictogram moest ik zelf een afbeelding toevoegen via `android:icon`. Wat betreft het opstartscherm, moest ik eerst een stijl maken in het styles.xml-bestand, genaamd `Theme.App.Splashscreen`. Deze stijl had een witte achtergrondkleur, een afbeelding met de naam "splashscreen" en een thema voor gebruik na de Splash screen. Daarna moest ik ook nog in het AndroidManifest-bestand het thema instellen als waarde binnen `android:theme`.

## Afbeelding met tekst, schermopname Automatisch gegenereerde beschrijvingNavigatie

### Visuele weergave en code

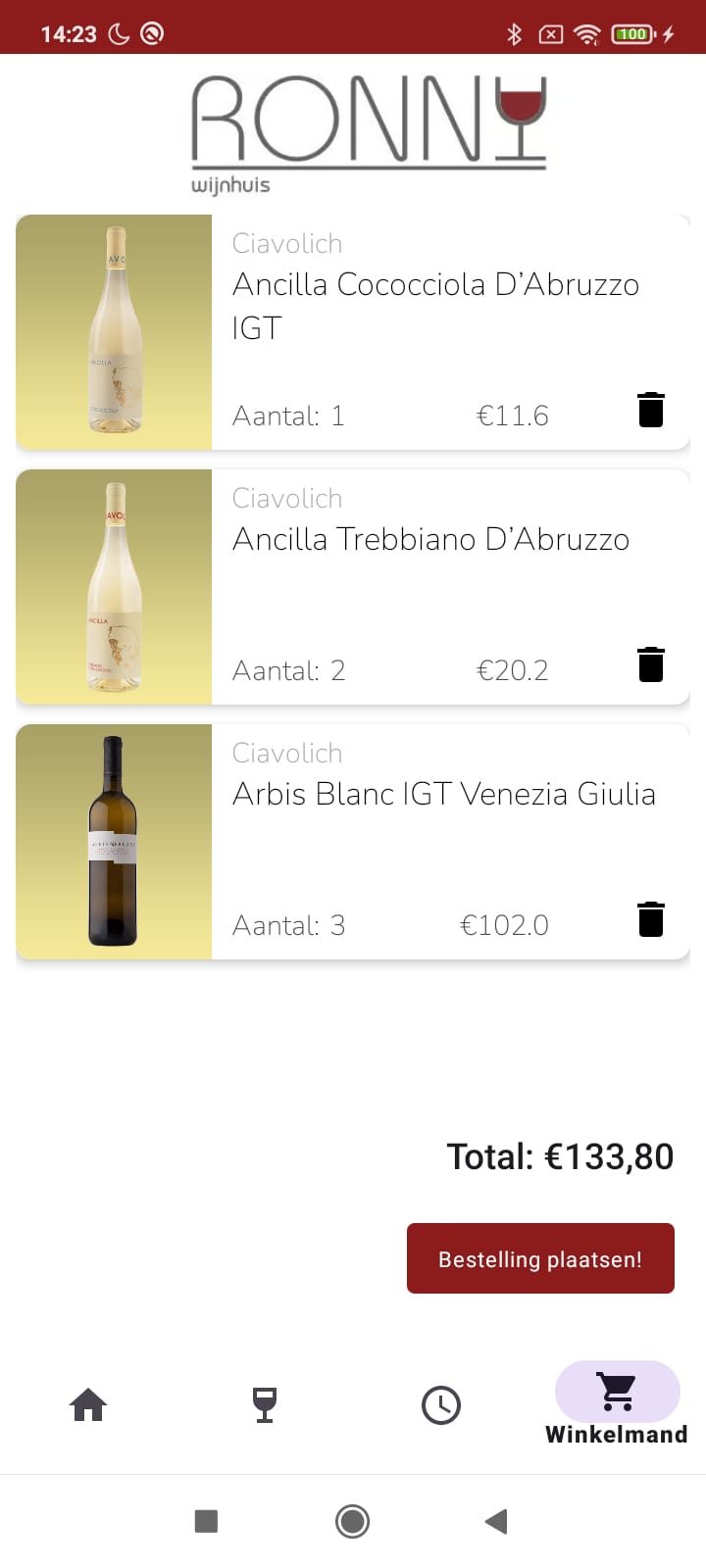
Afbeelding met tekst, schermopname, Menselijk gezicht, persoon

Automatisch gegenereerde beschrijving



Afbeelding met tekst, schermopname, fles

Automatisch gegenereerde beschrijving

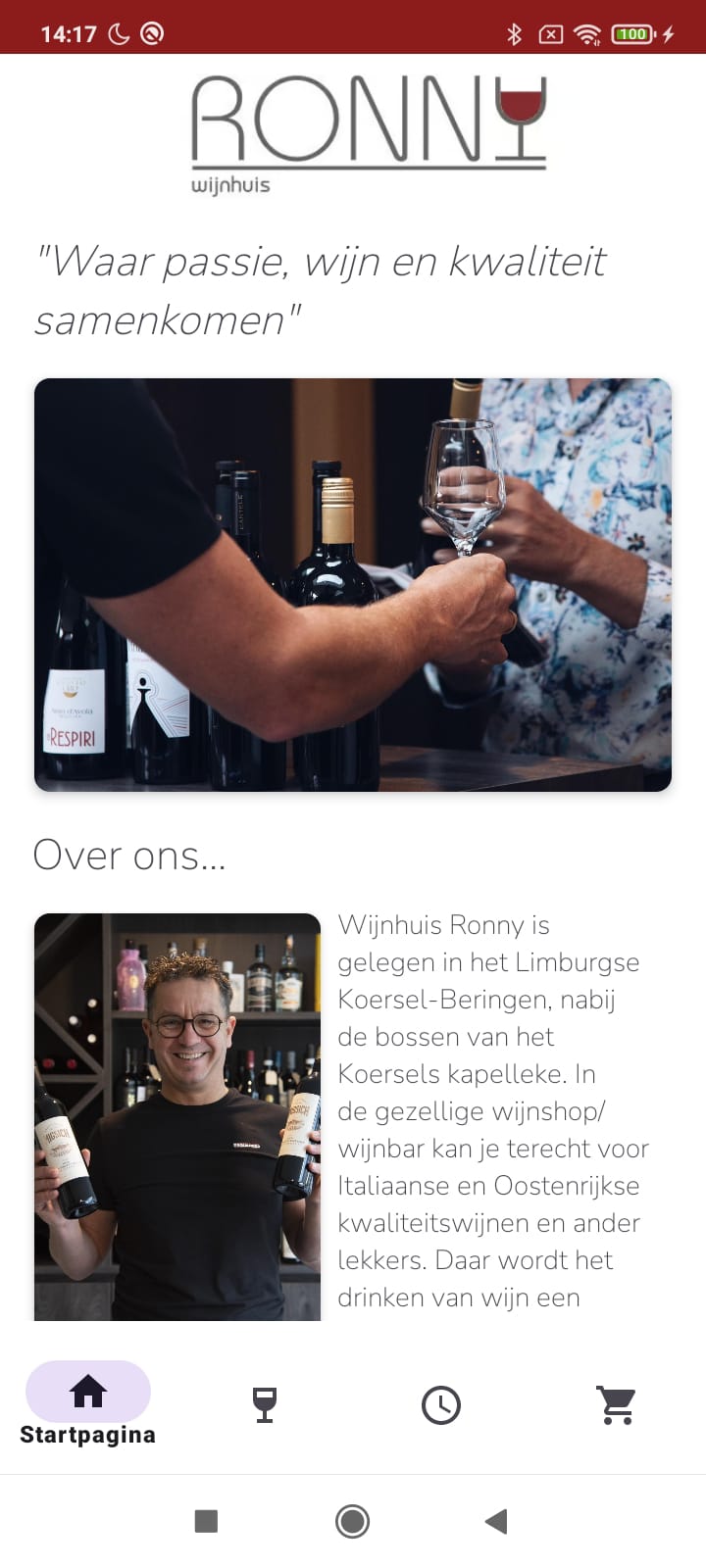


### Uitleg

Deze `bottom\_nav.xml` definieert het menu voor een Bottom Navigation View in een Android-applicatie. Het menu bevat vier items, elk geassocieerd met een specifiek fragment in de app. Deze XML-configuratie wordt gebruikt in combinatie met een Bottom Navigation View in de `activity\_main.xml`, waardoor een navigatiebalk ontstaat met de genoemde items. Elk item vertegenwoordigt een specifiek deel van de app en wordt geassocieerd met een overeenkomstig fragment, waardoor gebruikers gemakkelijk kunnen schakelen tussen verschillende delen van de app. Het gebruik van pictogrammen en duidelijke titels draagt bij aan een simpele gebruikerservaring.

## Startpagina

### Visuele weergave



### HomePagePhotosRepository

#### Code

Afbeelding met tekst, elektronica, schermopname, computer

Automatisch gegenereerde beschrijving

#### Uitleg code

Bovenstaande Kotlin-code vertegenwoordigt een klasse genaamd HomePagePhotosRepository, bedoeld voor het ophalen van foto-URL's uit een Firebase Realtime Database.

De klasse begint met het initialiseren van een databaseReference die verwijst naar het gedeelte "Startpagina" in de Firebase-database. Dit wordt gedaan in de constructor, waarbij ook een logboekmelding wordt gegenereerd om te melden dat de database-referentie is geïnitialiseerd.

Vervolgens wordt een companion object gebruikt om een Singleton-patroon te implementeren. Dit zorgt ervoor dat er slechts één exemplaar van HomePagePhotosRepository wordt gemaakt en gebruikt.

De getInstance-methode in het companion object zorgt voor het ophalen van deze enkele instantie. Als er nog geen instantie bestaat, wordt er één gemaakt.

De hoofdfunctionaliteit van de klasse is te vinden in de getPhotoUrls-methode. Deze methode accepteert een callback-functie die een lijst met foto-URL's als parameter heeft. Binnen de methode wordt een listener toegevoegd aan de database-referentie om eenmalig gegevens op te halen. Bij succes worden de foto-URL's uit de database gehaald en doorgegeven aan de callback. Als er zich een fout voordoet, wordt een foutmelding gegenereerd en wordt een lege lijst naar de callback gestuurd.

In essentie biedt deze klasse een gestructureerde manier om foto-URL's op te halen vanuit de database, waarbij het Singleton-patroon ervoor zorgt dat er geen onnodige instanties worden gemaakt.

### HomepageFragment

#### Code

Afbeelding met tekst, elektronica, schermopname, computer

Automatisch gegenereerde beschrijving

#### Uitleg code

De `HomepageFragment`-klasse is een subclass van `Fragment`. Binnen deze klasse wordt de binding met de bijbehorende lay-out (`FragmentHomepageBinding`) geïmplementeerd via het Android View Binding-systeem.

Het fragment heeft een `ViewModel` genaamd `HomepageFragmentViewModel`, die verantwoordelijk is voor het beheren van gegevens en logica die specifiek zijn voor dit fragment.

In de `onCreateView`-methode, die wordt aangeroepen bij het maken van de fragment, wordt de binding geïnitialiseerd, de ViewModel geassocieerd en wordt de observer ingesteld om wijzigingen in de lijst met foto-URL's te volgen. Wanneer er veranderingen optreden, worden de ImageViews bijgewerkt met behulp van de Picasso-bibliotheek om afbeeldingen van de ontvangen URL's weer te geven. Vervolgens wordt de methode `loadPhotoUrls` van de ViewModel aangeroepen om de foto-URL's te laden.

De `onDestroyView`-methode wordt gebruikt om het `\_binding`-object naar `null` te zetten, waardoor geheugenlekken worden voorkomen.

Kort samengevat biedt dit fragment een gestructureerde manier om de lay-out te initialiseren, de ViewModel te associëren en dynamisch afbeeldingen bij te werken op basis van wijzigingen in de lijst met foto-URL's. De `ViewModel`-laag is verantwoordelijk voor het beheren van de gegevenslogica, terwijl het fragment zich richt op het verzorgen van de gebruikersinterface-interactie.

### HomepageFragmentViewModel

#### Code

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

#### Uitleg code

De `HomepageFragmentViewModel`-klasse is een subclass van `ViewModel`. Binnen deze klasse wordt er een instantie van `HomePagePhotosRepository` aangemaakt via de methode `getInstance()`.

De klasse bevat een `MutableLiveData` met de naam `photoUrlsLiveData`, die fungeert als een waarnemerbare gegevensbron voor de lijst met foto-URL's

De `loadPhotoUrls`-methode in deze klasse wordt aangeroepen om foto-URL's op te halen vanuit de `HomePagePhotosRepository`. Hierbij wordt een callback gebruikt om de verkregen foto-URL's door te geven aan `photoUrlsLiveData`. Het gebruik van `postValue` zorgt ervoor dat deze bewerking veilig wordt uitgevoerd op de hoofdthread.

In wezen fungeert deze ViewModel als een tussenschakel tussen de `HomePagePhotosRepository` en het bijbehorende fragment (`HomepageFragment`). Het initieert het ophalen van foto-URL's en zorgt ervoor dat wijzigingen in deze gegevens worden gecommuniceerd naar de observerende componenten, zoals de ImageViews in het bijbehorende fragment.

### Fragment\_homepage.xml

#### Code

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

Afbeelding met tekst, menu, schermopname

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, menu

Automatisch gegenereerde beschrijving

#### Uitleg code

Dit XML-bestand definieert de lay-out voor het fragment met de naam `HomepageFragment`. De lay-out begint met de specificatie van de variabele `homepageFragmentViewModel` in het `<data>`-element, waardoor gegevensuitwisseling tussen de lay-out en de ViewModel mogelijk is.

Binnen de `ScrollView` bevindt zich een `ConstraintLayout`, dat de hoofdindeling van de UI-elementen beheert. Er zijn meerdere `TextView`-elementen voor het weergeven van tekst, waaronder de slogan, een over-ons-kop, en alinea's tekst. De teksteigenschappen zoals lettergrootte en lettertype zijn aangepast voor elk tekstelement.

Daarnaast zijn er twee `CardView`-elementen, `photoCardView` en `overOnsCardView`, die fungeren als containers voor de `ImageViews`. Deze cards hebben afgeronde hoeken en een subtiele schaduw voor visuele aantrekkelijkheid.

Binnen de `CardView`-elementen bevinden zich `ImageView`-elementen (`imageView2` en `overOnsImageView`) voor het weergeven van afbeeldingen. Momenteel is de bron van de afbeeldingen ingesteld op een standaardafbeelding genaamd "homepagephoto", maar deze zullen later dynamisch worden vervangen door afbeeldingen die worden opgehaald vanuit de ViewModel.

Verder zijn er `TextView`-elementen specifiek voor het gedeelte "Over Ons", waaronder een subkop, tekstalinea's en informatie over Ronny.

## Wijnen pagina

### Wijncategorieën pagina

#### Visuele weergave

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, voedsel

Automatisch gegenereerde beschrijving

#### WinesCategoryPhotosRepository

##### Code

Afbeelding met tekst, elektronica, schermopname, computer

Automatisch gegenereerde beschrijving

##### Uitleg code

Deze Kotlin-code vertegenwoordigt een repositoryklasse genaamd `WinesCategoryPhotosRepository`, ontworpen om foto-URL's op te halen vanuit een Firebase Realtime Database voor het gedeelte "Categorieën".

De klasse begint met het initialiseren van een `databaseReference` naar het "Categorieen"-gedeelte in de Firebase Realtime Database. Deze verwijzing wordt gemaakt in de constructor van de klasse, waarbij ook een logboekmelding wordt gegenereerd om aan te geven dat de database-referentie is geïnitialiseerd.

Daarnaast maakt de klasse gebruik van het Singleton-patroon, waardoor slechts één instantie van `WinesCategoryPhotosRepository` kan bestaan. Dit wordt bereikt door middel van het `companion object` en de `getInstance`-methode.

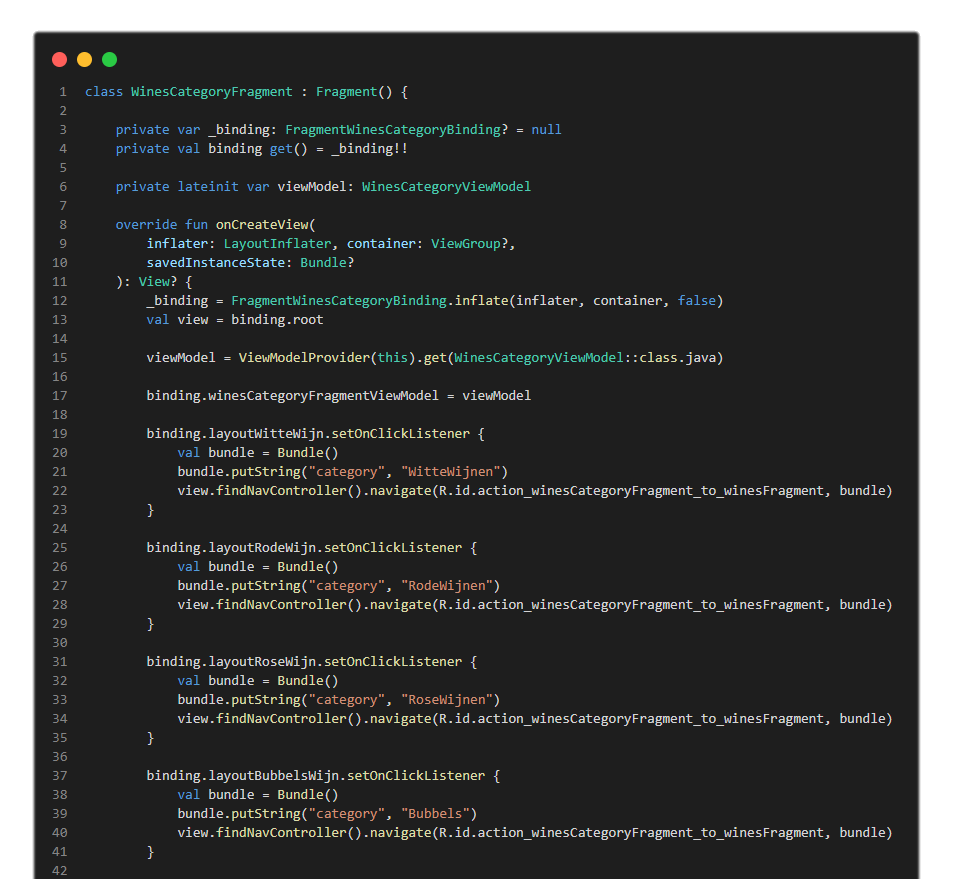
De `getPhotoUrls`-methode wordt gebruikt om foto-URL's op te halen vanuit de database. Hiervoor wordt een `ValueEventListener` aan de database-referentie toegevoegd om eenmalig gegevens op te halen. In het geval van een succesvolle gegevenswijziging worden de foto-URL's uit de database gehaald en doorgegeven aan een callback-functie.

De callback-functie wordt gedefinieerd als een lambda-uitdrukking met het type `(List<String>) -> Unit`. Binnen deze callback worden de foto-URL's verwerkt en doorgegeven aan de externe componenten die deze repository gebruiken. In het geval van een fout bij het ophalen van de gegevens, wordt een foutmelding gegenereerd en wordt een lege lijst naar de callback gestuurd.

Kort samengevat biedt deze klasse een gestructureerde manier om foto-URL's op te halen vanuit een Firebase-database met de mogelijkheid van Singleton-gebruik, waardoor efficiënt beheer van deze gegevens mogelijk is.

#### WinesCategoryFragment

##### Code



Afbeelding met tekst, elektronica, schermopname, scherm

Automatisch gegenereerde beschrijving

##### Uitleg code

Het fragment begint met de initialisatie van het View Binding-systeem, waarbij de bijbehorende binding variabele `\_binding` wordt gemaakt voor het bijhouden van verwijzingen naar de lay-outelementen. Dit voorkomt geheugenlekken door de binding in de `onDestroyView`-methode naar `null` te zetten.

De ViewModel, genaamd `WinesCategoryViewModel`, wordt geïnitialiseerd om de logica en gegevens voor dit fragment te beheren. De binding van de ViewModel met de lay-out wordt tot stand gebracht.

Vervolgens wordt voor elk type wijn (bijv. Witte wijnen, Rode wijnen) een zogenaamd "layout-click listener" toegevoegd. Als een bepaald wijntype wordt geselecteerd, wordt het fragment genavigeerd naar een ander fragment (`winesFragment`) met de bijbehorende wijncategorie.

Daarnaast wordt een observer ingesteld op de `photoUrlsLiveData` van de ViewModel. Deze observeert wijzigingen in de lijst met foto-URL's en bij wijzigingen worden de afbeeldingen dynamisch bijgewerkt met behulp van de Picasso-bibliotheek.

Ten slotte wordt de `loadPhotoUrls`-methode aangeroepen om de foto-URL's vanuit de ViewModel op te halen en te observeren.

In essentie biedt dit fragment een gestructureerde gebruikersinterface voor het selecteren van wijncategorieën, waarbij de ViewModel zorgt voor de afhandeling van gegevens en logica, en het View Binding-systeem de interactie tussen de lay-out en de code faciliteert.

#### WinesCategoryViewModel

##### Code

Afbeelding met tekst, schermopname, software, multimedia

Automatisch gegenereerde beschrijving

##### Uitleg code

In deze klasse wordt een instantie van `WinesCategoryPhotosRepository` gecreëerd met behulp van het Singleton-patroon. Dit zorgt ervoor dat gedurende de levensduur van de applicatie slechts één exemplaar van het repository bestaat. Het repository is verantwoordelijk voor het ophalen van foto-URL’s uit de Firebase Realtime Database.

De klasse maakt gebruik van het LiveData-mechanisme om wijzigingen in de lijst met foto-URL's te volgen. Er wordt een `MutableLiveData`-instantie genaamd `\_photoUrlsLiveData` geïnitialiseerd, terwijl een publieke `LiveData`-instantie genaamd `photoUrlsLiveData` wordt aangeboden voor externe observatie.

De methode `loadPhotoUrls` binnen deze klasse initieert het proces van het ophalen van foto-URL's vanuit het repository. Het resultaat, de lijst met foto-URL's, wordt vervolgens doorgegeven aan de `\_photoUrlsLiveData` met behulp van de methode `postValue`. Hierdoor kunnen externe componenten, zoals de XML-laag, dynamisch reageren op wijzigingen in de beschikbare foto-URL's.

#### Fragment\_wines\_category.xml

##### Code

Afbeelding met tekst, schermopname, menu

Automatisch gegenereerde beschrijving

Afbeelding met tekst, menu, schermopname

Automatisch gegenereerde beschrijving

Afbeelding met tekst, menu, schermopname

Automatisch gegenereerde beschrijving

Afbeelding met tekst, menu, schermopname

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, menu

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, menu

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, menu

Automatisch gegenereerde beschrijving

##### Uitleg code

De bovenstaande XML-code beschrijft de opmaak van een lay-outbestand binnen een Android-applicatie, specifiek voor het fragment met de naam "WinesCategoryFragment."

De gegevensbindingen worden gespecificeerd onder het "data" element, waarbij de variabele "winesCategoryFragmentViewModel" wordt gedeclareerd als van het type "com.example.appwijnhuisronny.WinesCategoryViewModel."

Binnen de lay-out is er een `ScrollView` die de mogelijkheid biedt om door een lijst met weergave-elementen te scrollen. Binnen deze `ScrollView` is een `LinearLayout` geplaatst met een verticale oriëntatie. Deze `LinearLayout` bevat verschillende `ConstraintLayout`-elementen, elk vertegenwoordigend een wijncategorie.

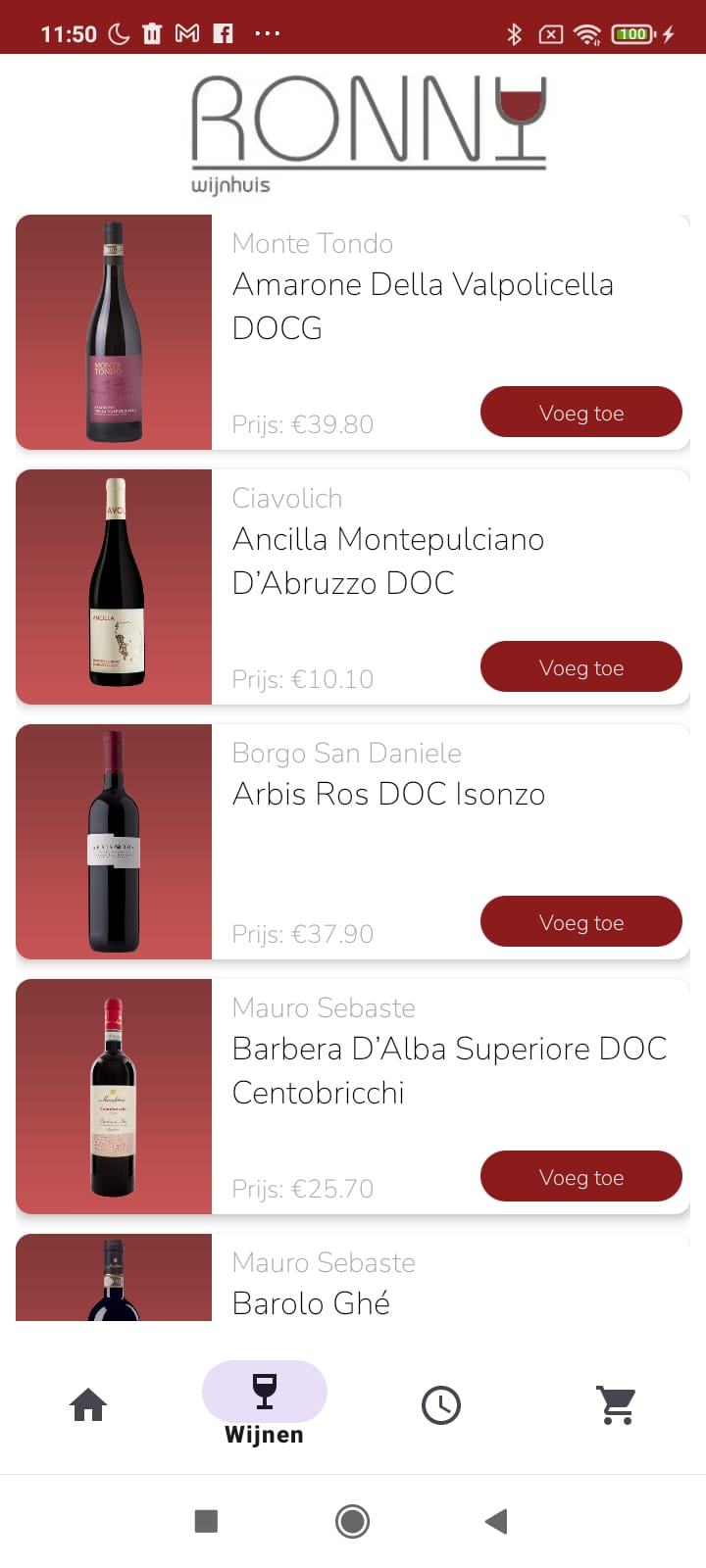
Elk `ConstraintLayout`-element bestaat uit een achtergrondafbeelding (`ImageView`) en een tekstlabel (`TextView`). De afbeeldingen worden gespecificeerd met unieke identificatoren zoals "image\_witte\_wijn," "image\_rode\_wijn," enzovoort, elk met een bijbehorende achtergrondafbeelding en tekstlabel voor de betreffende wijncategorie. De tekstlabels worden dynamisch geplaatst ten opzichte van de afbeeldingen met behulp van het ConstraintLayout.

Elk `ConstraintLayout`-element heeft een gewicht toegewezen, wat betekent dat ze gelijkmatig verdeeld worden in de beschikbare ruimte, waardoor ze elk een gelijk deel van het beschikbare schermgebied innemen.

Deze lay-out lijkt ontworpen te zijn voor een selectiescherm van wijncategorieën, waarbij elke categorie een achtergrondafbeelding en een bijbehorende tekst heeft. De `onClick`-attributen zijn geconfigureerd om een gemeenschappelijke klikafhandelingsmethode genaamd "onLayoutClick" aan te roepen wanneer een gebruiker op een bepaalde categorie klikt.

### Wijnen pagina

#### Visuele weergave

Afbeelding met tekst, fles, schermopname, ontwerp

Automatisch gegenereerde beschrijving

#### Wine Model

##### Code

Afbeelding met tekst, schermopname, scherm, multimedia

Automatisch gegenereerde beschrijving

##### Uitleg code

De `Wine`-klasse is gemarkeerd met het sleutelwoord `data`, wat aangeeft dat deze klasse primair bedoeld is voor het vastleggen en overbrengen van gegevens. De klasse heeft verschillende eigenschappen (properties) die de kenmerken van een wijn representeren:

- `BackgroundImage`: Een optionele String die de achtergrondafbeelding van de wijn vertegenwoordigt.

- `Image`: Een optionele String die de afbeelding van de wijn vertegenwoordigt.

- `Druif`: Een optionele String die de druivensoort van de wijn vertegenwoordigt.

- `Info`: Een optionele String die aanvullende informatie over de wijn bevat.

- `Inhoud`: Een optionele String die de inhoud van de wijn (bijv. 750 ml) vertegenwoordigt.

- `Naam`: Een optionele String die de naam van de wijn vertegenwoordigt.

- `Prijs`: Een optionele String die de prijs van de wijn vertegenwoordigt.

- `Regio`: Een optionele String die de regio van herkomst van de wijn vertegenwoordigt.

- `Wijndomein`: Een optionele String die het wijndomein van de wijn vertegenwoordigt.

- `Aantal`: Een optionele integer die het aantal flessen van de wijn vertegenwoordigt (standaard ingesteld op 1).

- `TotalPrice`: Een berekende eigenschap die de totale prijs van de wijn vertegenwoordigt, berekend als het product van de prijs en het aantal flessen.

Deze klasse biedt een gestructureerde manier om informatie over een wijn te organiseren en door te geven binnen mijn project. Het gebruik van optionele parameters (met een standaardwaarde van `null`) maakt het mogelijk om slechts een subset van eigenschappen in te vullen als dat nodig is, afhankelijk van de beschikbare informatie over een specifieke wijn.

#### WinesRepository

##### Code

Afbeelding met tekst, elektronica, schermopname, computer

Automatisch gegenereerde beschrijving

##### Uitleg code

De `WinesRepository`-klasse is verantwoordelijk voor de toegang tot en het ophalen van wijngerelateerde gegevens uit een Firebase Realtime Database.

De klasse heeft een private eigenschap `databaseReference` van het type `DatabaseReference`, die is geïnitialiseerd met de referentie naar de "Wijnen" knoop in de Firebase-database. Dit wordt gedaan in de `init`-blok, dat wordt uitgevoerd wanneer een instantie van de klasse wordt gemaakt. Er wordt een logboekmelding gegenereerd om aan te geven dat de database-referentie is geïnitialiseerd.

De klasse bevat een companion object met een `getInstance`-functie die verantwoordelijk is voor het ophalen van een instantie van de `WinesRepository`. De functie gebruikt een dubbele-check-vergrendeling (double-checked locking) om ervoor te zorgen dat er slechts één instantie van de repository wordt gemaakt.

De `loadWines`-functie is bedoeld voor het ophalen van wijnen uit de database op basis van een opgegeven categorie. Deze functie accepteert een categorie (bijvoorbeeld "RodeWijnen") en een `MutableLiveData`-object (`winesList`) dat de lijst met wijnen zal bevatten. Het maakt gebruik van een `ValueEventListener` om wijzigingen in de database bij te houden.

Binnen de `onDataChange`-methode wordt de snapshot van de database verwerkt. Voor elke kind-snapshot wordt een `Wine`-object gemaakt met behulp van de gegevens uit de database. Deze `Wine`-objecten worden verzameld in een lijst (`\_winesList`), die vervolgens wordt gepost naar het `winesList`-object.

Mocht er een fout optreden tijdens het lezen van de database, dan worden bijbehorende logboekmeldingen gegenereerd om mogelijke problemen te signaleren. Deze repository biedt dus een gestructureerde manier om wijnen vanuit de database op te halen en te leveren aan de UI-laag van een Android-applicatie.