Car Pulse code documentation

**1.My Garage – main screen**

На почетна страна корисникот се запознава со Car Pulse. Најнапред дефинираме празна листа во која ќе ги зачувуваме сите наши внесени автомобили *(слика 1)*

A computer screen shot of code

Description automatically generated

*Слика 1*

Дефинираме метод initState(), кој ја повикува функцијата Future<void> loadCars(), која ги вчитува автомобилите преку CarService и ја апдејтира нивната состојба *(слика 2)*

A computer screen shot of a program code

Description automatically generated

*Слика 2*

На најгорниот дел од екранот ние сме пречекани со App Bar. На App Bar-от од левата страна го поставивме текстот “My Garage” со кој го дефинираме самиот почеток на нашата апликација, додека на десната страна ставивме иконка од облаче. Со клик на облачето ние ќе сме редиректирани на страна за приказ на временска прогноза или Weather Screen (која ќе ја обработиме во точка 2). Ова можеме да го погледнеме на *слика 3*.

A computer screen shot of text

Description automatically generated

*Слика 3*

Под нашиот App Bar создаваме List View и со помош на методот \_buildCarCard создаваме листа од така наречени картички во кои ги вметнуваме сите наши автомобили. Секој наш автомобил добива по една картичка соодвнетно. Прикажано на *слика 4*. Методот \_buildCarCard() ги создава самите картички и прикажува информации за возилата.

A computer screen shot of a program code

Description automatically generated

*Слика 4*

Внесот на секој автомобил го правиме со помош на floatingActionButton() односно зеленото кругче со плус знакче кое е лоцирано на најдолниот десен агол на екранот. При клик на ова копче корисникот добива можност да пополни мала форма со фирмата и моделот на неговиот прв или наредно автомобил (прикажано на слика 5).

A computer screen shot of text

Description automatically generated

*Слика 5*

Под полињата за внес на фирма и модел на автомобилот стојат опцијии за cancel, опција за да го поништиме внесот на автомобилот и опција add, која дозволува внес на самиот автомобил. Ова го правиме преку функцијата \_addNewCar(), која проверува дали сме ја пополниле фирмата и моделот на автомобилот и доколку се е исполнето го реализира самиот метод. Слика 6 и слика 7.

A computer code on a black background

Description automatically generated

*Слика 6*

A computer screen shot of a program code

Description automatically generated

*Слика 7*

Методот Future<void> \_onCarTap дозволува кога на почетниот екран ќе кликнеме на некој автомобил, да не пренасочи кон неговата соодветна страна за детали, односно CarDetailScreen (обработен во точка 3). Дополнително во самиот метод е повикан методот onPhotoChanged() кој ја апдејтира фотографијата на самиот автомобил доколку е променета претходно од друг екран. *Слика 6.*

A computer screen shot of a program code

Description automatically generated

*Слика 6*

**2. WeatherScreen – временска прогноза**

Целта на овој екран е да добиеме информации за временската прогноза во наредните 3 денови и да добиеме информација дали е погодно времето за да го испереме нашиот автомобил. Оваа функционалност ние ја имплементираме користејќи веб сервис, односно Weather API и локациски сервиси со што овозможуваме приказ на временската прогноза (доколку корисникот даде дозвола за пристап до неговата локација). Со методите getWeatherData(), \_requestLocationPermission() и \_getCurrentLocation() бараме дозвола за локација на корисникот и доколку ја добиеме ја споредуваме со соодветниот status code за дополнителна модификација на методите. Доколку корисникот дозволи користење на локација, функцијата враќа одговор, а наспроти враќа error message.

Потоа поставуваме App Bar на екранот со наслов ‘Weather’. Дополнително имаме билдер кој ја гради страницата во зависност од connection state *(слика 1)*. Билдерот ја враќа максималната и минималната температура во тој период, како и состојбата – дали е погодно да го переме нашето возило или не со соодветните пораки: ’You can wash the car! There is no rain expected in the next 3 days’ или ‘Caution: Rain or snow is expected in the next 3 days.’

A screen shot of a computer code

Description automatically generated

A computer screen shot of a computer screen

Description automatically generated

*Слика 1 и 2*

**3. CarDetailsScreen – детали за селектираниот автомобил**

На овој екран корисникот добива детали за селектираниот автомобил. На горниот дел од екранот стои AppBar-от кој испишува текст за фирмата и моделот на селектираниот автомобил. *(слика 1)*

A computer code with text

Description automatically generated with medium confidence

*Слика 1*

Директно под AppBar-от стои празна слика од нашиот автомобил со центриран текст под неа. Со клик на празното поле за слика, ние добиваме можност за внес на слика. На *слика 2* повикуваме \_showImagePickerDialog(), функција која ни ги овозможува следните две опции: првата опција е избор на слика директно од нашата галерија, додека втората опција е внес на слика преку нашата камера (сензор). Дополнително креираме функција за избор на фотографијата и зачувување на фотографијата која сме ја одбрале или фотографирале. *(слика 3)*

A computer screen shot of a program code

Description automatically generated

*Слика 2*

A screen shot of a computer program

Description automatically generated

*Слика 3 – избор за внес на слика*

*A computer screen shot of code

Description automatically generated*

*Слика 4 – функциите за избор и внес на слика*

Под сликата корисникот е пречекан со следните 4 функционалности:

* Поглед на сервисната историја на автомобилот, кој има можност самостојно да ја внесе
* Персонализиран планер за идни модификации кои корисникот може да ги испланира преку овој екран
* Поглед за сите идни сервиси кои корисникот треба да ги направи во иднина
* Поглед за преглед и внес за сите детални информации поврзани со неговиот автомобил

Овие функционалности ние ги вметнуваме преку clickableSquareBlocks на кои имплементираме соодветни иконки од патеката /car\_pulse/icons (custom иконки). Соодветните пренасочувања и самото градење на clickableSquareBlocks гo правиме преку методот \_buildSquareBlocks, кој зема аргументи за насловот и сликата соодвнетно. О*вој метод е прикажан на слика 5*. Дополнително имаме променлива selectedCar на која ја задаваме вредноста од автомобилот кој сме го селектирале

**A computer screen shot of a program code

Description automatically generated**

*Слика 5*

**4. Servis History -екран за преглед на вашите сервиси**

Градиме конструктор за ServiceHistoryScreen со потребниот аргумент selectedCar, кој ја чува вредноста на нашиот селектиран автомобил. Најпрвин дефинираме App Bar со текст соодветен на страната, односно ‘Service Book’ и arrow back иконка, со која се враќаме на претходната страна *(слика 1)*

A screen shot of a computer program

Description automatically generated

*Слика 1*

Користиме ListView за приказ на сликата и соодветниот текст за селектираниот автомобил. Креираме DataTable во која ги чуваме информациите за сервисите, поле за преглед на сервисот и копче за бришење на истиот тој сервис. Состојбата на сервисот ја мапираме во променлива serviceRecord која наследува од /model/ServiceInfo каде се чуваат сите соодветни променливи и методи. *Ова е прикажано во слика 2.*

A screen shot of a computer program

Description automatically generated

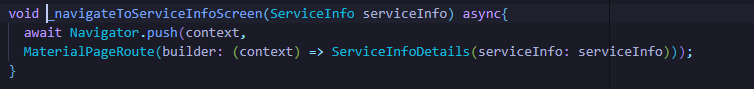
*Слика 2*

Со користење на ElevatedButton креираме копче со наслов “Show”, односно прикажи го сервисот. Со плик на копчето го повикуваме методот \_navigateToServiceInfoScreen(serviceRecord) со кој сме редиректирани до страната ServiceInfoDetails *(слика 3 и слика 4)*

A computer screen shot of a program code

Description automatically generated

*Слика 3 – креирање на копчето “show”*



*Слика 4 – редирекција кон страната ServiceInfoDetails*

На новата страна сме пречекани со App Bar кој содржи наслов “Service Details” и иконка со стрелка која ни овозможува враќање на претходната страна.Дополнително имаме пополнета DataTable организирана во две колони, односно Attribute – за насловот и Attribute – за внесената вредност (слика 5)

A computer screen shot of a program code

Description automatically generated

*Слика 5*

По прегледот на сервис, како што спомнавме имаме копче за бришење на даден сервис. Оваа функција ни е обележена со црвена канта за отпадоци како симбол за бришење на информации. Ова поле *(прикажано на слика 6)* повикува метод \_deleteServiceInfo(serviceRecord) кој го брише сервисот и го апдејтира carService *(слика 7)*.

A computer screen shot of a program

Description automatically generated

*Слика 6*

A computer screen shot of a service

Description automatically generated

*Слика 7*

Кога ќе погледнете вон табелата, во долниот десен агол имаме повторно floatingActionButton во форма на зелено кругче со плус иконка, која ни дозволува да додадеме нов сервис. При клик на копчето, ние сме пренасочени кон AddServiceScreen – односно екранот за внес на нашиот сервис (слика 8). Дополнително користиме и соодветни променливи како newService за зачувување на самиот сервис

A computer screen shot of a program code

Description automatically generated

*Слика 8*

На новиот екран ние градиме контролери за сите податоци кои ги зачуваваме *(слика 9)*. Потоа креираме App Bar со наслов “Add Service” и стрелка назад, доколку се предомислиме за додавање на нов сервис. Внесот на податоци е повторно од типот DataTable и во секој DataRow ја внесуваме соодветната информација *(слика 10)*

A screen shot of a computer program

Description automatically generated

*Слика 9 – сите контролери за внес на информации*

A screen shot of a computer program

Description automatically generated

*Слика 10 – пример за еден од повеќето DataRows*

Кога ќе завршиме со внесувањето на информации имаме зелено копче “Save Service” *(слика 11)*, кое го повикува методот \_saveServiceInfо(), метод кој не прашува дали сме сигурни, дали сакаме да го зачуваме внесениот сервис *(слика 12)*. Во тој метод, доколку притиснеме дека сакаме да го зачуваме сервисот повикуваме уште еден метод \_saveAndRedirect() (слика13) во кој во променливата newService ги зачувуваме состојбите на сите променливи за кои сме ги внеле посакуваните вредности. Потоа тој не враќа кон првичната страна и ја апдејтира табелата со нашиот нов внесен сервис.

A screen shot of a computer program

Description automatically generated

A computer screen shot of code

Description automatically generated

A screenshot of a computer program

Description automatically generated

*Слика 9, 10 и 11*

**5. Modification Planner – планирање на модификации**

Оваа страна е наменета за приказ на сите наши идни модификации кои планираме да ги вградиме на нашиот автомобил. Дефинираме App Bar со соодветен текст на страната, односно ‘Modification Planner’ и arrow back иконка, со која се враќаме на претходната страна *(слика 1)*

A screen shot of a computer program

Description automatically generated

*Слика 1*

Како и претходниот екран исто во моделот на ListView чуваме контејнер каде ја поставуваме сликата на автомобилот и текст за соодветната марка и модел. Под ова креираме повторно DataTable со полиња за модификацијата, преглед на модификација и поле за бришење на истата таа модификација. Дополнително ова се го мапираме во променливата modificationRecords *(слика 2)*

A screen shot of a computer program

Description automatically generated

*Слика 2*

Копчињата за приказ на модификацијата и бришење на модификација соодветствуваат со копчињата на ServisHistory (точка 4). Дополнително креираме и floatingActionButton сличен на тој во ServisHistory, но со редиректирање кон страницата за додавање на модификација *(слика 3)*

A screen shot of a computer program

Description automatically generated

*Слика 3*

На сличен начин е создаден и овој екран, најпрвин креираме контролери за потребните променливи (слика 4), потоа креираме метод \_saveModificationInfo() кој ни прикажува дали сакаме да ја зачуваме модификацијата и го повикува методот \_saveAndRedirect() кој креира newModification каде ја сместуваме новата модификација *(слика 5 и слика 6)*

A screenshot of a computer screen

Description automatically generated

*Слика 4*

A screen shot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

*Слика 5 и 6*

**6. Edit stats – информации за нашиот автомобил**

Овој екран е наменет за приказ на перформансите на нашето возило, како и дополнителни информации. Дефинираме App Bar со текст соодветен на страната тоест ‘Stats’ и arrow back иконка, со која се враќаме на претходната страна *(слика 1)*

A screen shot of a computer program

Description automatically generated

*Слика 1*

Повторно креираме DataTable каде на ист начин ги зачувуваме податоците. Повторно создаваме floatingActionButton кој не редиректира на страницата AddEditScreen. Креираме контролери за сите потребни параметри *(слика 2)*

A screen shot of a computer program

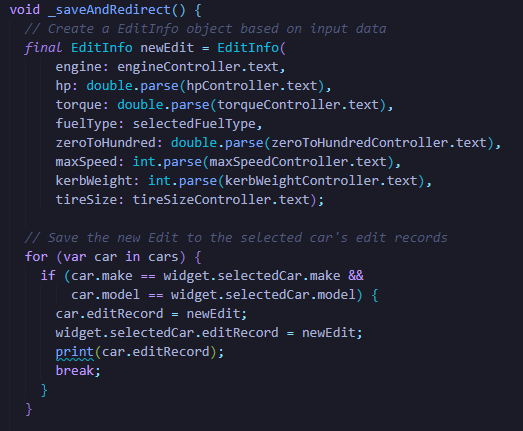
Description automatically generated

*Слика 2*

И нормално креираме метод \_saveEditInfo() и \_saveAndRedirect() кои ни го зачувуваат внесот на податоци и не враќаат кон првичната страна каде ја пополнуваме табелата со дадените информации *(слика 2 и слика 3)*

A screen shot of a computer program

Description automatically generated

**

*Слика 2 и 3*

7. Upcoming Service – преглед на идни сервиси

Овој екран е наменет за дефинирање на нашите сервиси. Дополнително и можност за потсетник на нашите сервиси кои ни се ближат (користиме awesome notifications – flutter). Создаваме App Bar со текст ‘Upcoming service” и arrow back иконка, со која се враќаме на претходната страна. На корисникот му даваме можност да го одбере датумот на неговиот претходен сервис и со помош на тој датум автоматски го пресметуваме датумот на идниот сервис за автомобилот.

**\* Design patterns**

Како design patterns имаме искористено:  
-singleton design pattern

A screen shot of a computer program

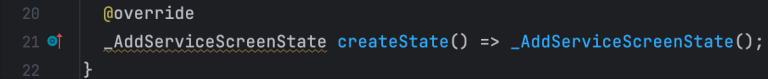
Description automatically generated

-builder pattern

A screen shot of a computer program

Description automatically generated

-state pattern



A computer screen with text

Description automatically generated

-factory pattern (код од car.dart)

A computer screen shot of a program code

Description automatically generated

-observer pattern (код од car\_details\_screen.dart)

A black screen with white text

Description automatically generated

-callback pattern (код од addServiceInfo.dart)

A screen shot of a computer

Description automatically generated