

Документација за проектната задача по

Мобилни информациски системи

FareShare

Изработиле: Димитар Пешевски, 206024 и Бојан Ристевски, 201177

Ментори: проф. д-р Петре Ламески и асс. Мила Додевска

Име на апликацијата: FareShare

Краток опис на апликацијата: FareShare претставува мобилна апликација за објавување на огласи за споделување на меѓуградски превоз. Конкретно, корисниците коишто патуваат на релација помеѓу 2 града може да објават оглас со детали за нивното патување (град на поаѓање и пристигнување, време на поаѓање и пристигнување, слободни патнички места, цена по патник и слично), додека корисниците кои се заинтересирани за патување на истата релација може да резервираат патно место за истиот оглас. На овој начин, апликацијата овозможува споделување на патничките трошоци за меѓуградски превоз.

Линк до изворниот код на апликацијата: <https://github.com/peshevskidimitar/fareshare>

Архитектура на апликацијата

Принципите на **Layered Architecture** предвидуваат раздвојувањето на апликацијата на различни слоеви, каде секој слој има свои одговорности и функции. Во контекстот на мобилната апликација FareShare, ова значи дека апликацијата е поделена на следниве основни слоеви:

- **Кориснички интерфејс (UI) слој:** Ова е слојот кој ги визуелизира податоците и овозможува интеракција со корисникот. Во FareShare, овој слој вклучува функционалности како најава, регистрација, прелистување на огласи, креирање на огласи, внес на податоци за резервација на патничко место итн.
- **Бизнис (сервис) слој:** Во овој слој се содржат бизнис правилата, процесите и функциите на апликацијата. Тука се обработуваат сите барања од корисниците, како и сите бизнис логички операции. Пример за функционалности во FareShare се проверка на слободни места за даден оглас и слично.
- **Податочен слој:** Овој слој се однесува на справувањето со податоците и пристапувањето до нив. Тука се вклучени операции како читање и запишување на податоци, извлекување на информации од база на податоци како и други извори на податоци. FareShare вклучува чување на информации за корисниците, огласите и резервациите.

- **Доменски слој:** Овој слој ги дефинира моделите коишто се користат во рамки на апликацијата. Конкретно, кај FareShare тоа се моделите за оглас, резервација и корисник.

Примената на Layered Architecture во мобилната апликација FareShare го олеснува одржувањето, проширувањето и тестирањето на апликацијата. Секој слој е независен и има јасно дефинирани граници на одговорност, што овозможува поединечно тестирање и заменување на компонентите без да се наруши функционалноста на останатите делови од апликацијата. Ова го прави процесот на развој и одржување на апликацијата помалку комплексен и поголемиот дел од работата се организира и изведува посебно за секој слој.

Веб сервиси

Во апликацијата FareShare, користењето на три веб сервиси од Firebase - **Firestore Database**, **Firebase Authentication** и **Firestore Storage** - игра клучна улога во обезбедувањето на функционалностите и услугите за корисниците.

- **Firestore Database:** Како NoSQL база на податоци, Firestore се користи за чување и менаџирање на податоците поврзани со огласите и резервациите во апликацијата FareShare. Ја избравме токму оваа база на податоци поради нејзината флексибилност и скалабилност, што овозможува лесно зачувување и пристап до различни типови на податоци како текст, слики, локации итн. Покрај тоа, Firestore обезбедува пристап во реално време и синхронизација на податоците, што е од суштинско значење за апликација како FareShare каде корисниците треба да бидат во тек со најновите информации за огласите и резервациите.
- **Firestore Authentication:** Овој сервис е одговорен за автентикацијата и авторизацијата на корисниците во апликацијата. Преку Firestore Authentication, корисниците на FareShare можат да се најават и да се регистрираат со свој кориснички профил, што обезбедува сигурен пристап до личните информации и функционалностите на апликацијата. Овој сервис обезбедува различни методи на автентикација, вклучувајќи најава преку е-пошта и лозинка и најава преку Google.
- **Firestore Storage:** Како складиште за кориснички генерирани содржини, Firestore Storage е одговорен за чување на слики, видеа и други медиумски датотеки кои се креирани или споделувани од страна на корисниците на FareShare и обезбедува брз и ефикасен начин за зачувување и пристап до содржини.

Примената на овие веб сервиси од Firebase значително ги подобрува перформансите, сигурноста и скалабилноста на апликацијата FareShare, односно користењето на овие сервиси овозможува да се намали оптоварувањето што го има апликацијата за сметка на тоа што Firebase се грижи за инфраструктурата и управувањето со сервисите.

Шаблони за дизајн на софтвер

Во апликацијата FareShare, шаблоните за дизајн на софтвер играат клучна улога во организацијата на кодот, подобрувајќи ги читливоста, одржливоста и проширувањето на апликацијата. Во продолжение следува објаснување на секој од користените шаблони и нивната примена во рамки на FareShare.

- **Singleton:** Овој шаблон се користи за обезбедување на единствена инстанца од класата за пристап до сервисите на Firebase, како Firestore Database, Firebase Authentication и Firebase Storage. Singleton шаблонот обезбедува дека во апликацијата постои само една инстанца од класите за пристап до овие сервиси, што ги подобрува перформансите и управувањето со ресурсите.
- **Factory:** Factory шаблонот се користи за креирање на објекти од класите Post и Reservation на база на дадени податоци од DocumentSnapshot. Оваа апстракција на креирање на објекти овозможува флексибилност и лесно проширување на апликацијата, особено ако во иднина се додадат нови типови на податоци кои треба да се мапираат во објекти.
- **Dependency Injection (DI):** Оваа техника се користи за инјектирање на репозиториумите од податочниот слој во погорните слоеви на апликацијата. Преку Dependency Injection, се овозможува растеретување на класите од одговорноста за креирање на зависностите и олеснување на тестирањето преку инјектирање на симулирани објекти при тестирање.
- **Observer:** Овој шаблон се користи во податочниот слој за имплементација на методите за пристап до податоците. Секој метод враќа Stream, на кој другите компоненти во апликацијата може да се претплатат и да реагираат на промени на податоците. Ова овозможува реализација на реактивен програмски модел, каде промените на податоците автоматски се одразуваат во сите компоненти што се претплатени на Stream-овите.
- **BLoC (Business Logic Component):** Овој шаблон се користи во бизнис (сервис) слојот за организација на бизнис логиката и комуникацијата со податочниот и презентацискиот слој. Преку BLoC шаблонот, се одвојува бизнис логиката од корисничкиот интерфејс, што го прави кодот подобро организиран и проширлив, овозможува лесно тестирање и промена на функционалностите. BLoC се користи како посредник помеѓу различните делови од апликацијата кој ги обработува барањата од корисниците и враќа резултати како одговор.

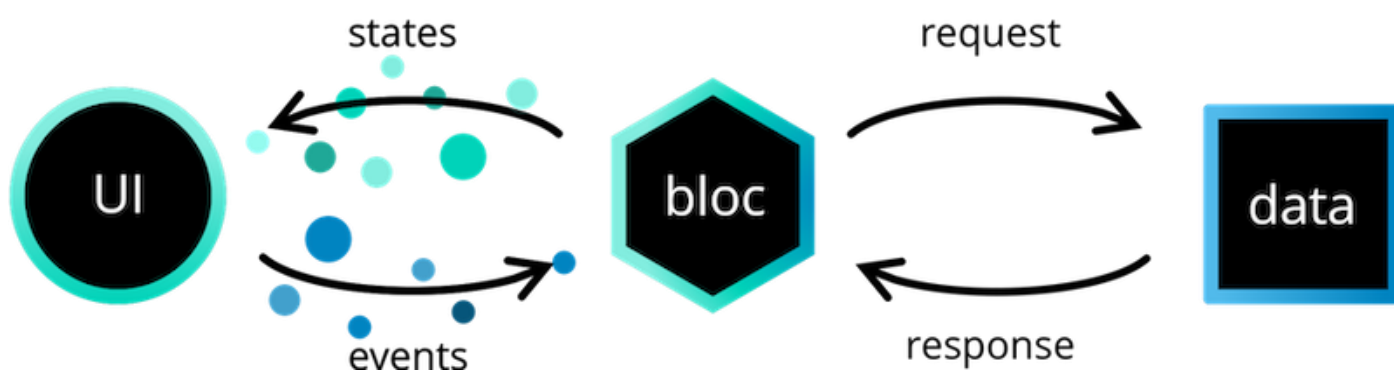
Со користење на овие шаблони, апликацијата FareShare постигнува подобро организиран и проширлив код, како и подобро управување со зависностите и одржливоста на кодот. Овие шаблони играат клучна улога во обезбедувањето на ефикасна и скалабилна архитектура на апликацијата.

BLoC (Business Logic Component)

Шаблонот за дизајн на софтвер BLoC (Business Logic Component) е шаблон за управување со состојбата кој обично се користи во Flutter апликации. Тој помага во управувањето со состојбата на апликацијата и ја олеснува изградбата на сложени апликации. Шаблонот BLoC се базира на концептот на реактивно програмирање (reactive programming), што овозможува декларативен пристап кон управувањето со промени во состојбата.

BLoC е дизајниран за да ги одвои бизнис логиката на апликацијата од корисничкиот интерфејс. Ова го прави полесно одржувањето и менувањето на кодот, бидејќи промените во бизнис логиката не влијаат на UI слојот. Оваа поделба на задачите исто така го прави полесно тестирањето на бизнис логиката на апликацијата, бидејќи е одделена од UI слојот.

Во BLoC, бизнис логиката на апликацијата се енкапсулира во BLoC класа. BLoC класата како влез прима настани (events), а како излез враќа состојби (states). Настаните претставуваат промени во состојбата на апликацијата кои треба да бидат обработени, додека состојбите ја претставуваат моменталната состојба на апликацијата.



Слика 1 „Приказ на BLoC архитектурата“

UI слојот на апликацијата комуницира со BLoC класата за да добие информациите за промените кои се случиле во состојбата на апликацијата. Оваа интеракција обично се прави користејќи го widget-от BlocBuilder од пакетот flutter_bloc. Widget-от BlocBuilder ја известува UI компонентата за промените во состојбата на апликацијата.

Шаблонот за дизајн на софтвер BLoC е еден од неколкуте шаблони за управување со состојбите кои можат да се користат во Flutter. Другите популарни шаблони за управување со состојбите вклучуваат Provider, Redux и MobX. Секој шаблон има свои предности и недостатоци, а изборот за кој шаблон да се користи зависи од конкретните потреби на апликацијата.

Локациски сервиси

Во рамки на апликацијата FareShare се интегрирани два локациски сервиси и тоа: Google Maps и Geolocator. Со нивна помош се овозможува при резервацијата на патничко место патникот да ја означи локацијата од каде што треба да биде подигнат што обезбедува прецизна локација и ја олеснува комуникацијата помеѓу корисникот и возачот. Конкретно, во рамки на формата за резервација преку библиотеката `google_maps_flutter` се прикажува мапа на којашто корисникот може да ја означи својата локација. Исто така при преглед на сопствените огласи корисникот може да ги види локациите на патниците кои направиле резервации.

Камера

Интегрирањето на камерата во рамки на апликацијата FareShare му овозможува на огласувачот да постави слика од своето возило којашто може да ја избере од галеријата или пак на лице место да го слика своето возило користејќи ја камерата. На овој начин секој патник што ќе сака да резервира место ќе може да има во предвид во какво возило ќе се вози.

Паметење состојба

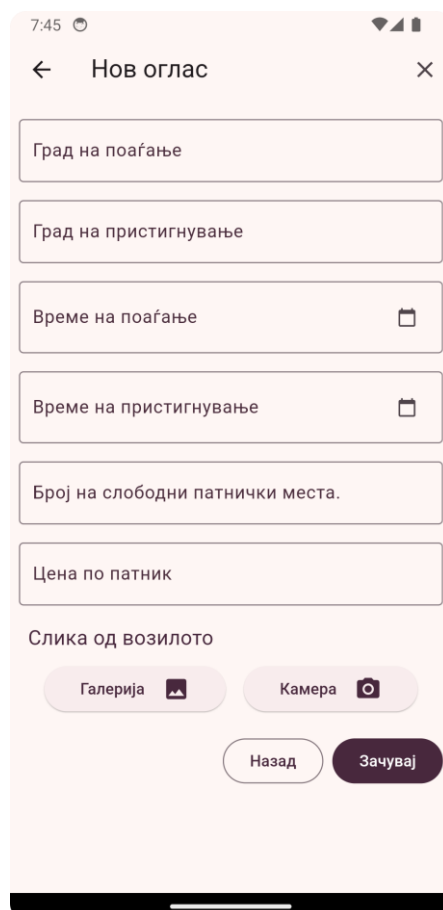
Паметењето состојба во апликацијата FareShare е овозможено преку `stateful widgets`, како и преку шаблонот за дизајн на софтвер BLoC кој овозможува ефикасно и скалабилно справување со состојбата. Речиси сите компоненти во рамки на апликацијата паметат состојба.

Custom UI елементи

За градење на презентацискиот слој го користиме Material Widget каталогот кој го нуди самата рамка за програмирање Flutter. Со помош на овој каталог градиме и custom UI елементи кои ги задоволуваат барањата на нашата апликација. Во прилог следуваат слики од екраните на апликацијата на кои се претставени компонентите од кои е изграден презентацискиот слој.



Слика 2 „Приказ на листата со огласи“



Слика 3 „Приказ на формата за нов оглас“


7:44

← Најава

Е-пошта

Лозинка

Најави се

 Најави се со Google

Регистрирај се

Слика 4 „Приказ на формата за најава“

7:44

← Регистрација

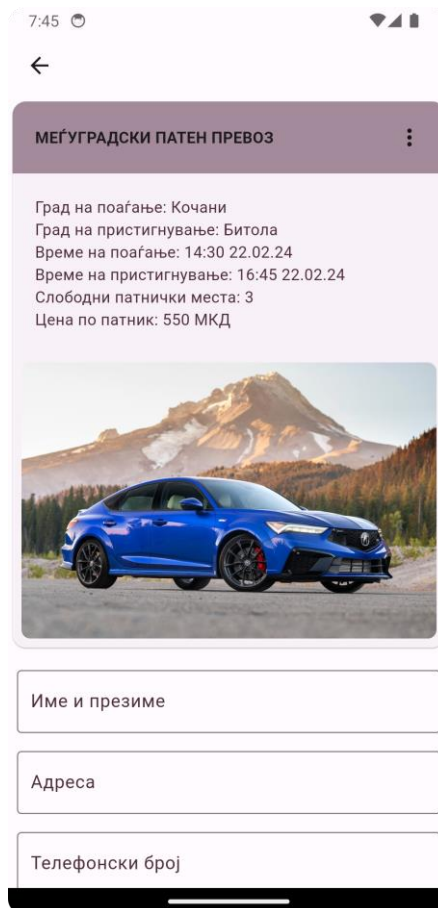
Е-пошта

Лозинка

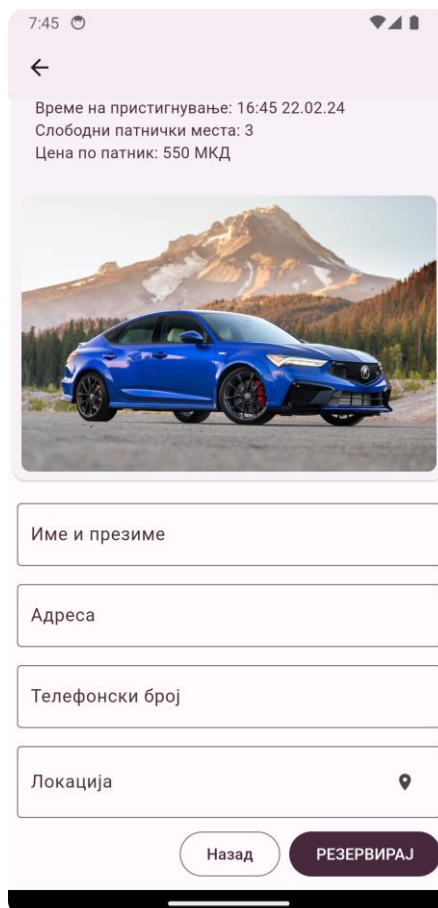
Потврди лозинка

Регистрирај се

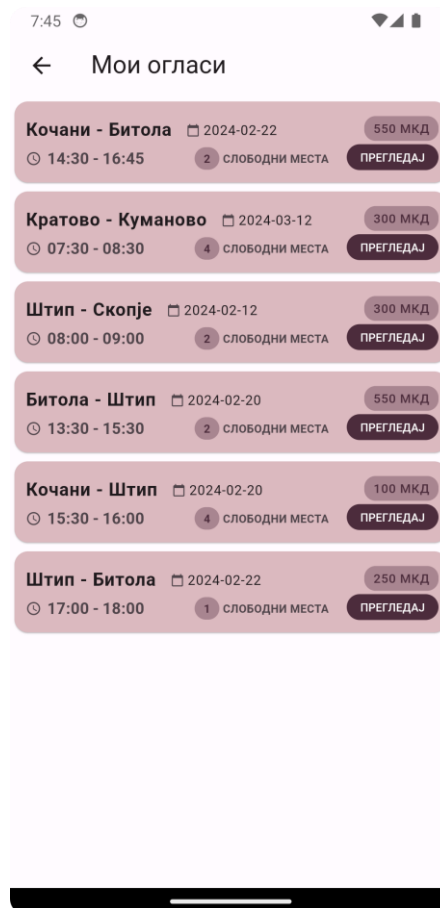
Слика 5 „Приказ на формата за регистрација“



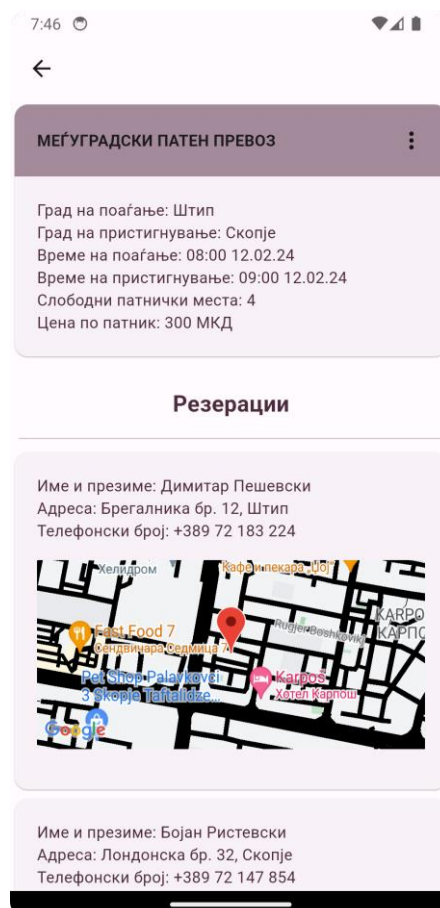
Слика 6 „Приказ на даден оглас и формата за резервација“



Слика 7 „Приказ на даден оглас и формата за резервација“



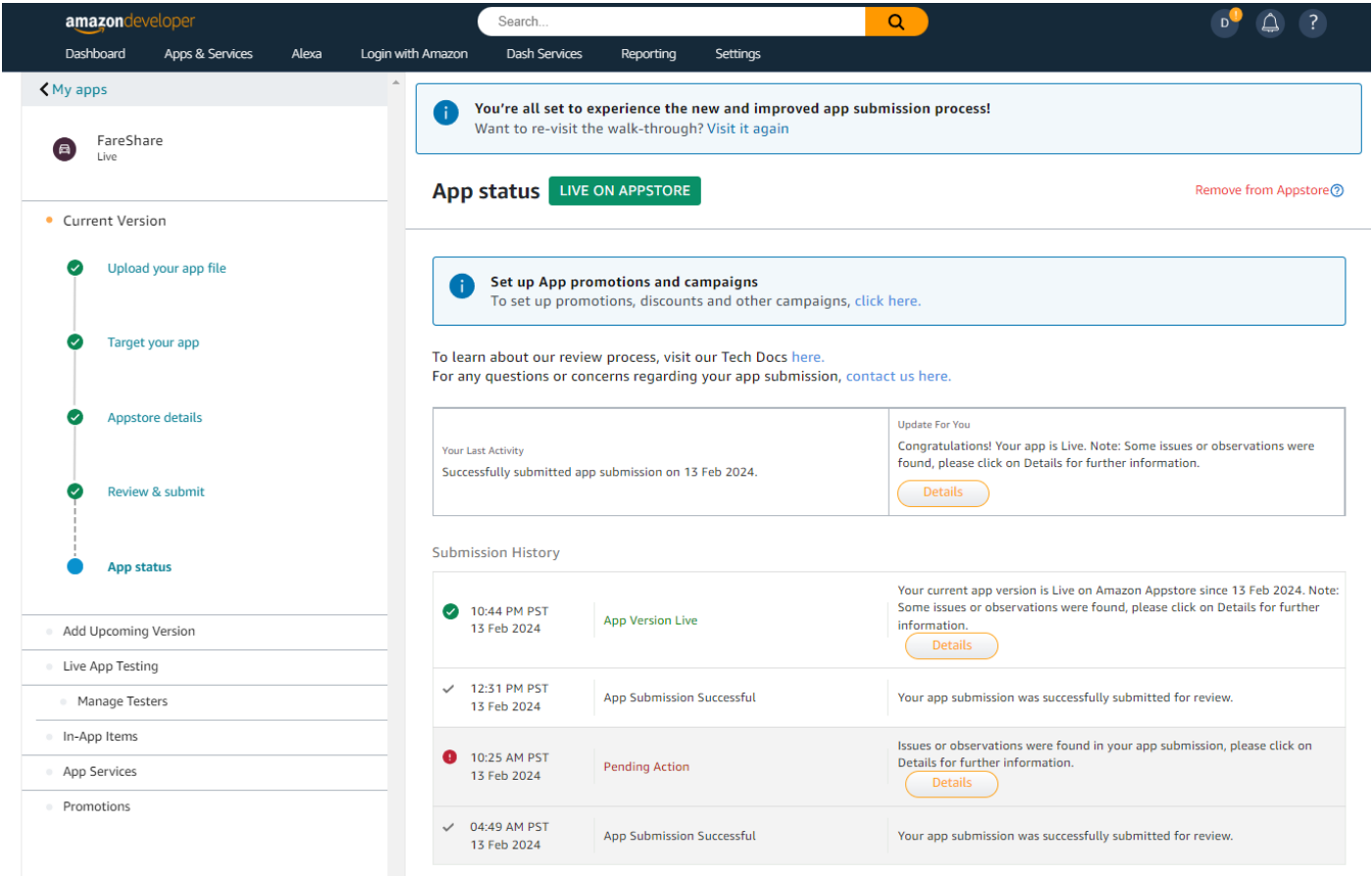
Слика 8 „Приказ листата од огласи на најавениот корисник“



Слика 9 „Приказ на листата со резервации“

Поставување на маркет

Нашата апликација FareShare ја поставивме на Amazon Appstore for Android. Во прилог следува слика за успешното поставување и дека апликацијата е live на Amazon Appstore.



Слика 10 „Поставување на маркет“