Paluch Bartłomiej
Nowoczesne Technologie Programistyczne

Projekt

Kalendarz/Organizer

Cel projektu: Celem projektu jest poznanie nowoczesnego, nieznanego nam wcześniej języka programowania oraz frameworku programowania aplikacji poprzez utworzenie działającej aplikacji w danym frameworku. W zgodzie z prowadzącym przedmiot zdecydowano o korzystaniu z języka Dart oraz Frameworku Flutter(żaden uczestnik zajęć nie posiadał doświadczenia z danym językiem oraz frameworkiem).

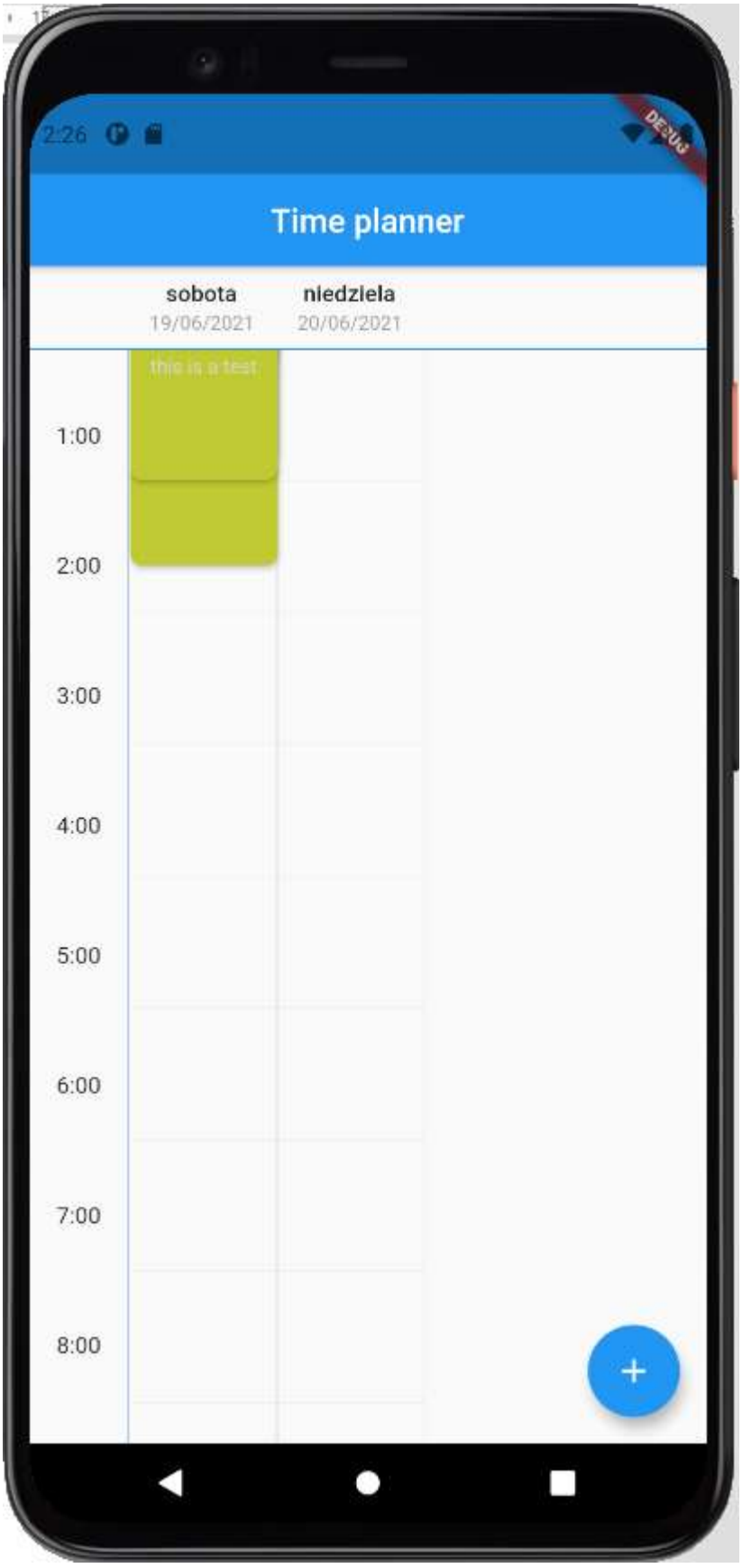
Przygotowania do projektu:
Przygotowano środowisko zgodnie ze sposobem zalecanym w dokumentacji.
https://flutter.dev/docs/get-started/install
Zapoznano się z podstawową dokumentacją frameworku Flutter.
https://flutter.dev/docs/reference/tutorials
Do tworzenia własnego kodu zastosowano jako referencję przykłady z cookbooka fluttera oraz przykłady zawarte w paczkach. https://flutter.dev/docs/cookbook źródła paczek podane w ich linkach.
Do rozwiązywania błędów wykorzystano dokumentację, stackoverflow, quora oraz
Caldav:
Zgodnie z zaleceniem prowadzącego, zapoznano się z protokołem CALDAV. Do utworzenia serwera wykorzystano oprogramowanie Baikal bazujące swoje działanie na języku programowania PHP oraz bazie danych Mysql. https://sabre.io/baikal/
Następnie skonfigurowano serwer i utworzono przykładowy kalendarz na serwerze. Przetestowano serwer z wykorzystaniem przykładowych zapytań przez klient Postman. Zapoznano się z tworzeniem klienta Caldav. Przeszukano bazę paczek flutter, okazuje się że nie istnieje implementacja klienta Caldav gotowa do użycia. Utworzenie klienta wymaga utworzenia zapytań:

- Pobrania obiektów
- Dodania obiektu
- Usunięcia obiektu
- Autoryzacji

Zapytania należy utworzyć poprzez zapytania html których zawartością oraz odpowiedzią jest dokument XML. Sposób implementacji zapytań znajduje się pod linkiem:
https://flutter.dev/docs/cookbook/networking/send-data
Do obiektu w kalendarzu należy utworzyć model obiektu. Ponieważ w zapytaniu należy przesłać obiekt XML do każdego zapytania należy utworzyć własny parser obiektu do formatu przyjmowanego przez protokół Caldav. Ze względu na poziom skomplikowania, oraz ograniczony czas, pomimo zainwestowania czasu w technologię zdecydowano się na rezygnację z tej technologii.

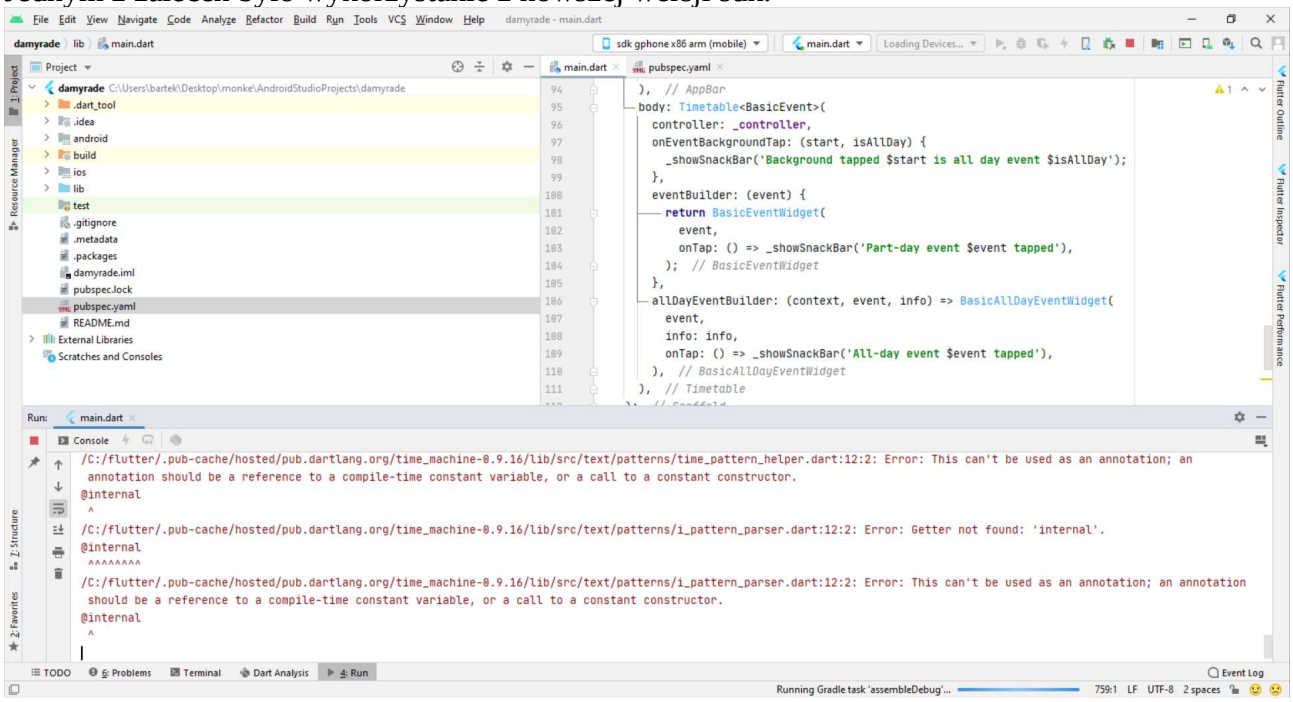
Interfejs:
Przeszukując gotowe rozwiązania zauważono, że utworzenie interfejsu kalendarza nie jest tak prostym zadaniem na jakie wygląda. Ze względu na konieczność uwzględnienia lat przestępnych, implementacji wielu widoków oraz mechanizmów tworzących kalendarz zalecane jest korzystanie z paczek. Ze względu na brak wcześniejszego doświadczenia z danym frameworkiem przy tworzeniu danego projektu ramy czasowe zmuszają mnie do wykorzystania z paczek. W poniższym projekcie wykorzystano przykłady gotowych paczek tworzących interfejs aplikacji do których należy wprowadzić odpowiednie, własne modyfikacje kodu źródłowego z zachowaniem wszelkich praw i obowiązków dostarczanych nam przez licencję oprogramowania.

Projekty w Github:
1. cal
https://pub.dev/packages/time_planner

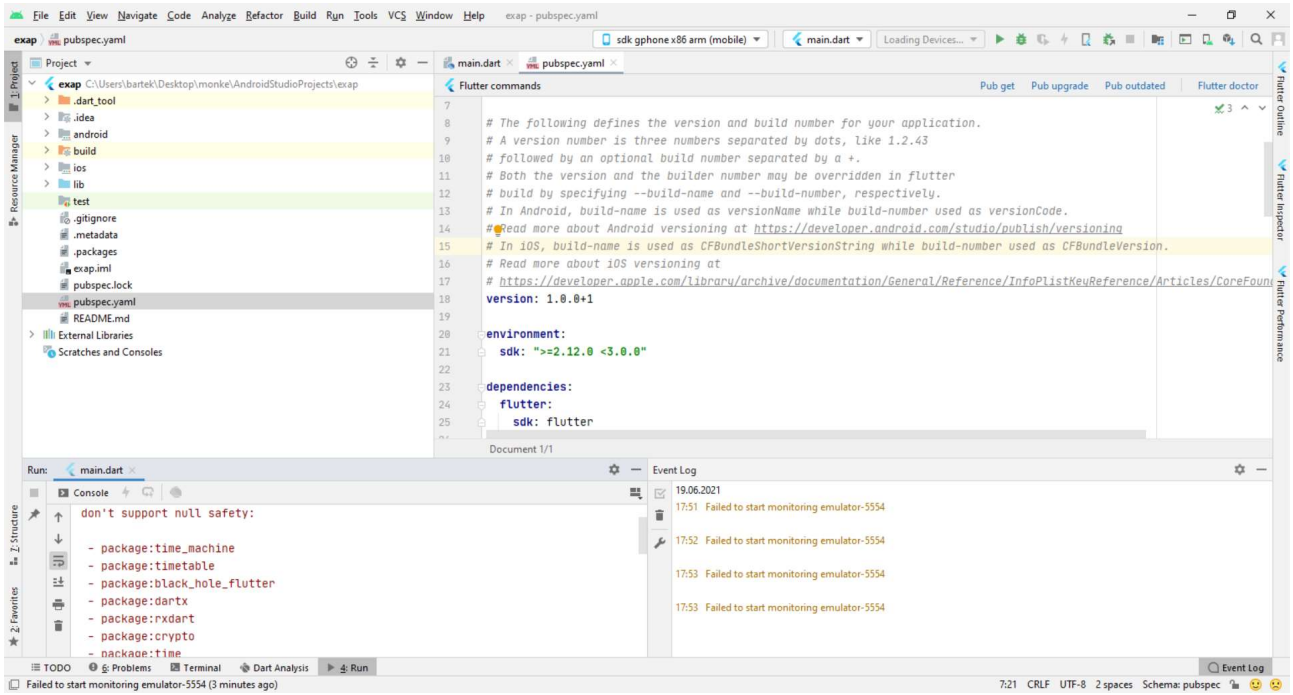


Aplikacja została utworzona na podstawie przykładu dostępnego na stronie źródłowej paczki. Podczas testowania paczki, przy tworzeniu przykładowego, mockowanego obiektu zauważono błąd pozwalający na utworzenie wydarzenia które wystaje poza ramy czasowe wydarzenia(czas ujemny). Dodatkowo, paczka pozwala na utworzenie odliczania czasu najwcześniej od 1:00 do 24:00. Próby zmiany godzin na inne niż w dokumentacji(od 0-24) nie pozwalała na uruchomienie działającej aplikacji. Paczka nie zawiera widoku tworzenia wydarzenia, należałoby ją zaimplementować. Po analizie kodu zrezygnowano z wykorzystywania paczki, konieczność gruntownej zmiany działania i analiza całości kodu powoduje, że czas dostępny na utworzenie aplikacji jest zbyt mały aby na podstawie tej paczki jedna osoba utworzyła w pełni funkcjonalną aplikację w dostępnym czasie. Przykładowa aplikacja z wykorzystaniem tej paczki znajduje się pod adresem:
https://jamalianpour.github.io/time_planner_demo/#/

2. damyrade
sdk: **">=2.7.0 <3.0.0"**
https://pub.dev/packages/timetable/example
Przykładowy kod paczki timetable 0.29 posiada problem z kompilacją domyślnego przykładu. Przegląd stacku wywołań oraz przeszukiwanie rozwiązania błędu nie dało żadnych efektów. Jednym z zaleceń było wykorzystanie z nowszej wersji sdk.



3. exap
sdk: **">=2.12.0 <3.0.0"**
Dla tego samego przykładu spróbowano wykorzystać inną wersję SDK fluttera, problemem okazał się być Non Null support dodane w nowszej wersji języka dart. Ze względu na dużą liczbę oprogramowania posiadającego



4. kal Zapoznanie się z modułem firebase – zalecanym sposobem połączenia zewnętrzną bazą danych znajdującą się w chmurze Google jest paczka Firebase. Paczka pozwala na połączenie z bazą danych poprzez API aplikacji. Do przekazania dostępu do API aplikacji należy skonfigurować plik google-services w katalogu android → app → src informacjami dostępnymi w projekcie utworzonym na stronie developera google. Dodatkowo, plik build.gradle musi zawierać deklarację wykorzystania usług google. Plik google.services pozwala na korzystanie z wielu różnych api google, nie tylko z Firebase. Flutter pozwala na konfigurację usług zarówno dla aplikacji android, jak i aplikacji pod system apple IOS. Aby utworzyć aplikację pod IOS należy posiadać urządzenie z systemem macOS. Konieczność połączenia internetowego aby korzystać z danej usługi konieczne jest utworzenie lokalnej bazy danych dla aplikacji. Zrezygnowano z technologii Firebase.
Do testowania konfiguracji wykorzystywano domyślną aplikację okienkową.

5. kalendarz – zawiera jeden z pierwszych kodów testowanych podczas tworzenia oprogramowania. Kod zawiera kalendarz nie bazujący na żadnej paczce. Po wielokrotnych modyfikacjach pozostawiono samą logikę kalendarza.

6. kalendarzyk – zawiera kod, testujący działanie no null w nowszej wersji fluttera. Wykorzystano tą samą aplikację co w kal.

7. tymrazemsieuda – zawiera przykład wykorzystania paczki tablecalendar 3.0. Paczkę wybrano jako docelowe narzędzie na którym będzie opierać się aplikacja.

8. inny – zawiera projekt nad którym początkowo były wykonywane. Bazuje na przykładowym kodzie z tymrazemsieuda dodając nowe elementy. Element dodany to dodawanie własnych elementów oraz usuwanie ich. Trwają pracę nad obsługą bazy danych(aktualnie dane są lokalne dla instancji aplikacji).
Baza danych: Paczka https://moor.simonbinder.eu/docs/getting-started/

9. mainbranch – zawiera aktualny stan projektu. Zaimplementowano: Obsługę bazy danych(zapisuje wydarzenia w bazie danych, ale po ponownym uruchomieniu nie uzupełnia Events. FutureBuilder ukazuje wszystkie obiekty, nie tylko obiekty z danego dnia. Kliknięcie na obiekt usuwa rekord ale tylko gdy jesteśmy na określonym dniu). Nie byłem w stanie poradzić sobie z jednokrotnym wykonaniem operacji na pierwszej odpowiedzi zapytania get. Nie byłem w stanie dodać do FutureBuilder warunkowego wyświetlania/dodawania obiektów.

Dokumentacja mainbranch:
Plik database.dart zawiera model bazy danych. Klasa appointment opisuje właściwości tabeli wydarzenia. Klasa główna bazy danych MyDatabase zawiera definicję bazy danych, połączenia z nią oraz definicję zapytań do bazy danych pobierających dane, dodających dane do tabeli i usuwające dane z tabeli. Funkcja _openConnection opisuje połączenie z bazą danych. W projekcie zastosowano bazę sqlite.
Plik database.g.dart zawiera wygenerowany przez paczkę kod implementujący połączenie z bazą danych oraz implementację zapytań opisaną przez przez database.dart.
Plik utils.dart zawiera dodatkowe narzędzia, funkcje i zmienne wykorzystywane w main.dart.
Plik main.dart zawiera główną logikę programu. Plik main.dart tworzy interfejs użytkownika poprzez zastosowanie odpowiednich widgetów. Zapytanie select zwraca obiekt Future, który jest obsługiwany przez FutureBuilder – klasę tworzącą obiekt w chwili gdy zawartość zapytania zostanie zmieniona. Aplikacja składa się z trzech głównych obiektów, Tablecalendar który implementuje widok kalendarza na podstawie paczki Tablecalendar. Sizedbox zawierający Futurebuilder zawierający listę obiektów znajdujących się w bazie danych. Ostatnim elementem jest FloatingObjectButton implementujący dodawanie obiektu do bazy danych.