



Katedra  
Informatyki i Automatyki  
Politechniki Rzeszowskiej

# Sprawozdanie końcowe

## Projekt Grzybek

Opracował(a): L05 156320

## Spis treści

|   |   |
|---|---|
| Praca w zespole .....   | 3 |
| Za co odpowiadałem w projekcie .....                          | 3 |
| jaki był mój okres aktywności w projekcie .....               | 3 |
| Co wykonałem w projekcie.....                                 | 3 |
| Zespół, a praca.....  | 7 |
| Jak przebiegała praca w zespole .....                         | 7 |
| za co w projekcie odpowiadali inni .....                      | 7 |
| za co w projekcie odpowiadali Project Manager'zy .....        | 7 |
| za co w projekcie powinni odpowiadać Project Manager'zy ..... | 7 |
| za co w projekcie odpowiadali interesariusze .....            | 8 |
| za co w projekcie powinni odpowiadać interesariusze .....     | 8 |

## Praca w zespole

### ZA CO ODPOWIADAŁEM W PROJEKCIE

W projekcie odpowiadałem za front-end m.in. stworzenie aktywności, fragmentów, ekranu tworzenia wydarzeń jak i kalendarza.

### JAKI BYŁ MÓJ OKRES AKTYWNOŚCI W PROJEKCIE

Nad projektem pracowałem intensywnie od 24.03.2019 do 21.04.2019. Ostatnie aktualizacje do projektu wniosłem 3.06.2019.

### CO WYKONAŁEM W PROJEKCIE

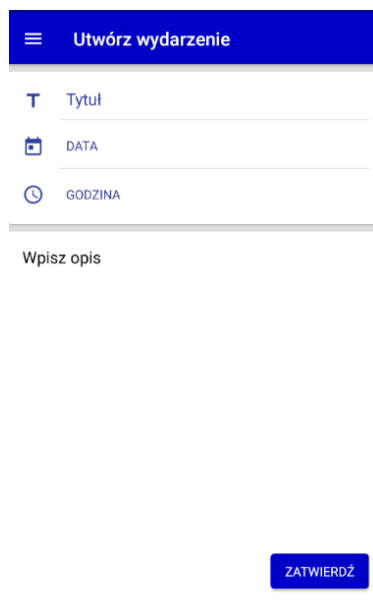
- Przycisk wyboru daty na ekranie „Utwórz wydarzenie”

```
datePickerButton.setOnClickListener{ it: View!
    if(!::date.isInitialized) {
        date = Calendar.getInstance()
    }
    var day: Int = date.get(Calendar.DAY_OF_MONTH)
    var month: Int = date.get(Calendar.MONTH)
    var year: Int = date.get(Calendar.YEAR)

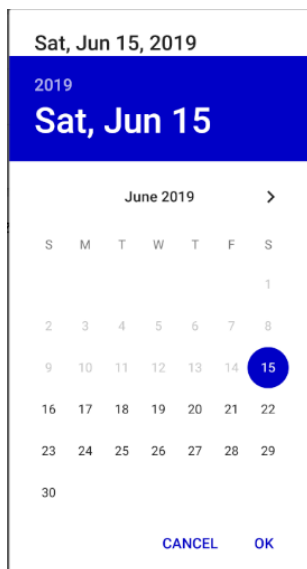
    dpd = DatePickerDialog(activity, DatePickerDialog.OnDateSetListener { view, nYear, nMonth, nDay ->
        date.set(nYear, nMonth, nDay)
        datePickerButton.setText(SimpleDateFormat("dd-MM-yyyy").format(date.time))
    }, year, month, day)
    dpd.datePicker.minDate = System.currentTimeMillis()
    dpd.show()
}
```

Zmienna *date* przechowuje komponent kalendarza, natomiast zmienna *dpd* przechowuje komponent okna wyboru daty. Data jest zapisywana do zmiennej *date*, a następnie wybrana przez nas data zostaje ustawiona jako tekst na przycisku.

Utworzony przeze mnie ekran ( samą stylizacją zajęła się inna osoba):



Kalendarz ukazujący się po wciśnięciu przycisku z napisem „DATA”:



Warto dodać, że dostępne dni zostały ograniczone jedynie do dnia dzisiejszego i następnych. Na komponencie można wybrać interesującą nas datę, a następnie po przyciśnięciu „OK” pojawi się ona jako tekst komponentu *button* w formacie „dd-MM-yyyy”.

|   |         |
|---|---------|
| T | Tytuł   |
| 📅 | DATA    |
| 🕒 | GODZINA |

|   |            |
|---|------------|
| T | Tytuł      |
| 📅 | 29-06-2019 |
| 🕒 | GODZINA    |

- Przycisk wyboru godziny na ekranie „Utwórz wydarzenie”

```
timePickerButton.setOnClickListener(object: View.OnClickListener {
    override fun onClick(v: View?) {
        if(!::date.isInitialized) {
            date = Calendar.getInstance()
        }
        var hour = date.get(Calendar.HOUR_OF_DAY)
        var minute = date.get(Calendar.MINUTE)

        tpd = TimePickerDialog(activity, TimePickerDialog.OnTimeSetListener{view, nHour, nMinute ->
            date.set(Calendar.HOUR_OF_DAY, nHour)
            date.set(Calendar.MINUTE, nMinute)
            timePickerButton.setText(SimpleDateFormat("HH:mm").format(date.time))
        }, hour, minute, is24HourView: true)
        tpd.show()
    }
})
```

Ten przycisk po wciśnięciu wyświetla okno wyboru godziny, które jest przechowywane w zmiennej *tpd*, następnie wybrana godzina jest zapisywana w zmiennej *date* oraz tekst przycisku zostaje podmieniony na nasz wybór w formacie „HH:mm”. Niestety tworzenie wydarzeń nie jest zabezpieczone przed wybraniem godziny z aktualnego dnia, ale wcześniejszej niż obecna.

|   |         |
|---|---------|
| T | Tytuł   |
| 📅 | DATA    |
| 🕒 | GODZINA |

|   |       |
|---|-------|
| T | Tytuł |
| 📅 | DATA  |
| 🕒 | 21:37 |

Dodatkowo aplikacja została zabezpieczona przed utworzeniem wydarzenia, które nie posiada tytułu, daty lub godziny, a po pozytywnym utworzeniu wydarzenia użytkownik zostaje o tym poinformowany stosownym komunikatem, przeniesiony do ekranu „Ekran główny” oraz utworzona zostaje aktywność wraz z informacjami dotyczącymi aktualnie stworzonego wydarzenia.

```
if(datePickerButton.text != resources.getString(R.string.default_date)
    && timePickerButton.text != resources.getString(R.string.default_time)
    && name != "") {
    confButton.isEnabled = false
}
```

Powyższy warunek działa na podstawie tekstu jaki zawierają przyciski daty, godziny oraz pole tekstowe dla tytułu.

Jeżeli, poniższe warunki nie są spełnione:

- Przycisk wyboru daty wyświetla napis „DATA”
- Przycisk wyboru godziny wyświetla „GODZINA”
- Pole tekstowe przeznaczone dla tytułu jest puste

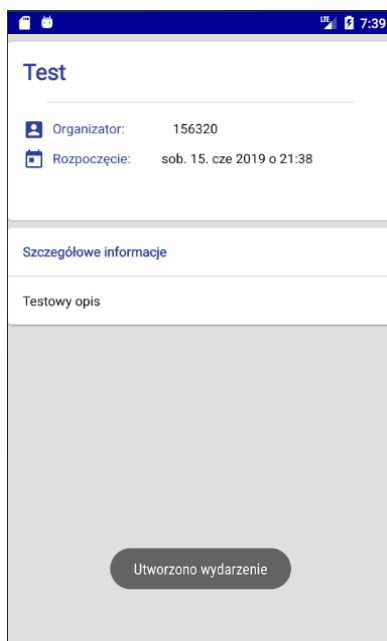
to zostanie utworzone wydarzenie.

```
Toast.makeText(activity, text: "Utworzono wydarzenie", Toast.LENGTH_LONG).show()
val fragment: Fragment? = FragStartScreen()
val man = activity?.supportFragmentManager
val trans = man?.beginTransaction()

if(fragment!=null) {
    trans?.replace(R.id.frameLay, fragment)
    trans?.commit()
}
(activity as MainScreen).switchFr(view)

val intent = Intent(activity, EventDetailsActivity::class.java)
    .putExtra( name: "event", DataForEvents( id: "", name, description, date, FirebaseAuth.getInstance().currentUser!!.uid)
startActivity(intent)
```

Powyższy kod obsługuje wyświetlenie komunikatu „Utworzono wydarzenie”, podmianę fragmentu, tak aby przejść do „Ekran główny” oraz przekazując odpowiednie dane uruchamia aktywność do prezentacji informacji o wydarzeniu. Funkcja `switchFr` ustawia „Ekran główny” w menu jako zaznaczony.



Jak widać na ekranie, po utworzeniu wydarzenia zostajemy przeniesieni do ekranu ze szczegółami o właśnie stworzonym wydarzeniu oraz wyświetlony zostaje komunikat.

W przypadku kiedy warunki potrzebne do stworzenia wydarzenia nie są spełnione, to wyświetla się następujący komunikat:

Utwórz wydarzenie

T

Test

DATA

19:42

Testowy opis

Nie wybrano daty, godziny lub nazwy

ZATWIERDŹ

- Stworzyłem kalendarz w ekranie „Kalendarz” ( tutaj stylizacją również zajmowała się inna osoba)

W tym celu wykorzystałem bibliotekę „CompactCalendarView” od SundeepK. Polegało to na wstawieniu gotowego komponentu do pliku xml właściwego fragmentu.



W dniach, w których są jakieś wydarzenia, wyświetlane są czerwone kropki odpowiadające ilości wydarzeń w tym dniu. Po wybraniu interesującego nas dnia, ukazuje się pod kalendarzem lista wydarzeń, naciskając na wyświetlone wydarzenie uruchamiamy aktywność wraz ze szczegółami na temat tego wydarzenia.

```
compactCalendarView!!.setListener(object : CompactCalendarView.CompactCalendarViewListener {
    override fun onDayClick(dateClicked: Date) {
        bookingsFromMap = compactCalendarView!!.getEvents(dateClicked)
        if (bookingsFromMap != null) {
            mutableBookings.clear()
            for (booking in bookingsFromMap!!) {
                val dt = booking.data as DataForEvents?
                val message = dt!!.getHour() + " " + dt.nameOfEvent
                mutableBookings.add(message)
            }
            adapter.notifyDataSetChanged()
        }
    }

    override fun onMonthScroll(firstDayOfNewMonth: Date) {
        showMonthYear!!.text = dateFormatForMonth.format(compactCalendarView!!.firstDayOfCurrentMonth)
        firstDayOfMonth.time = compactCalendarView!!.firstDayOfCurrentMonth
        getEvents(firstDayOfMonth)
    }
})
```

Przy wybieraniu dnia w kalendarzu wywoływana jest funkcja *onDayClick*, która wyświetla listę wszystkich wydarzeń z owego dnia. Informacje o wydarzeniach przechowywane są w liście *bookingFromMap*, skąd są pobierane dane, zapisywane do zmiennej *message* i dodawane do listy *mutableBookings*.

Kod obsługujący utworzenie aktywności z informacjami po wybraniu wydarzenia z listy:

```
bookingsListView.setOnItemClickListener { AdapterView.OnItemClickListener { parent, view, position, id ->
    val evClicked = bookingsFromMap!![position]
    val data = evClicked.data as DataForEvents?
    val intent = Intent(activity, EventDetailsActivity::class.java)
    intent.putExtra("name", data)
    startActivity(intent)
}
```

## Zespół, a praca

### JAK PRZEBIEGAŁA PRACA W ZESPOLE

Każdy w zespole miał wyznaczone zadania, które powinien wykonać w możliwie krótkim czasie. Nie ograniczaliśmy się tylko do własnych zadań, jeśli ktoś miał jakieś sugestie odnośnie czyjegoś kodu lub znalazł błąd, to się tym dzielił z innymi.

### ZA CO W PROJEKCIE ODPOWIADALI INNI

Inni w projekcie zajmowali się back-endem, bazą danych, stylizacją aplikacji, utworzeniem ikon. Była osoba odpowiedzialna za ustalanie spotkań oraz komunikację z Project managerami

### ZA CO W PROJEKCIE ODPOWIADALI PROJECT MANAGER'ZY

Project managerzy odpowiadali za obmyślenie funkcji jakie powinna spełniać ta aplikacja. Stworzenie harmonogramu oraz sprawdzanie postępów w projekcie.

### ZA CO W PROJEKCIE POWINNI ODPOWIADAĆ PROJECT MANAGER'ZY

Project managerzy powinni być również odpowiedzialni za dostarczenie nam makiet, aby wzorować na nich wygląd aplikacji. Stworzenie diagramów, które znacznie ułatwiłyby proces tworzenia.

#### ZA CO W PROJEKCIE ODPOWIADALI INTERESARIUSZE

Interesariusze odpowiedzialni byli za ogólną opiekę nad projektem, oceniali czy pomysły studentów są sensowne, jak i pomagali w rozwiązywaniu problemów w aplikacji.

#### ZA CO W PROJEKCIE POWINNI ODPOWIADAĆ INTERESARIUSZE

Nie mam żadnych zastrzeżeń odnośnie interesariuszy.