

**Akademia Górniczo-Hutnicza w Krakowie**

**Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej**

Inżynieria oprogramowania

Aplikacja do planowania czasu

Autorzy: Waldemar Świder, Adrianna Gnacek

Kierunek: Informatyka

Opiekun: dr inż. Marek Zachara

Kraków, 2020

**Spis treści**

1. **Opis systemu**
2. **Diagram przypadków użycia**
   1. Aktorzy
   2. System aplikacji
   3. Scenariusze przypadków użycia
3. **Wymagania niefunkcjonalne**
   1. Wymagania produktowe
   2. Wymagania organizacyjne
   3. Wymagania zewnętrzne
4. **Architektura systemu**
5. **Opis interfejsów**
6. **Stos technologiczny**
7. **Projekt testów**
8. **Analiza ryzyka**
9. **Lista narzędzi**

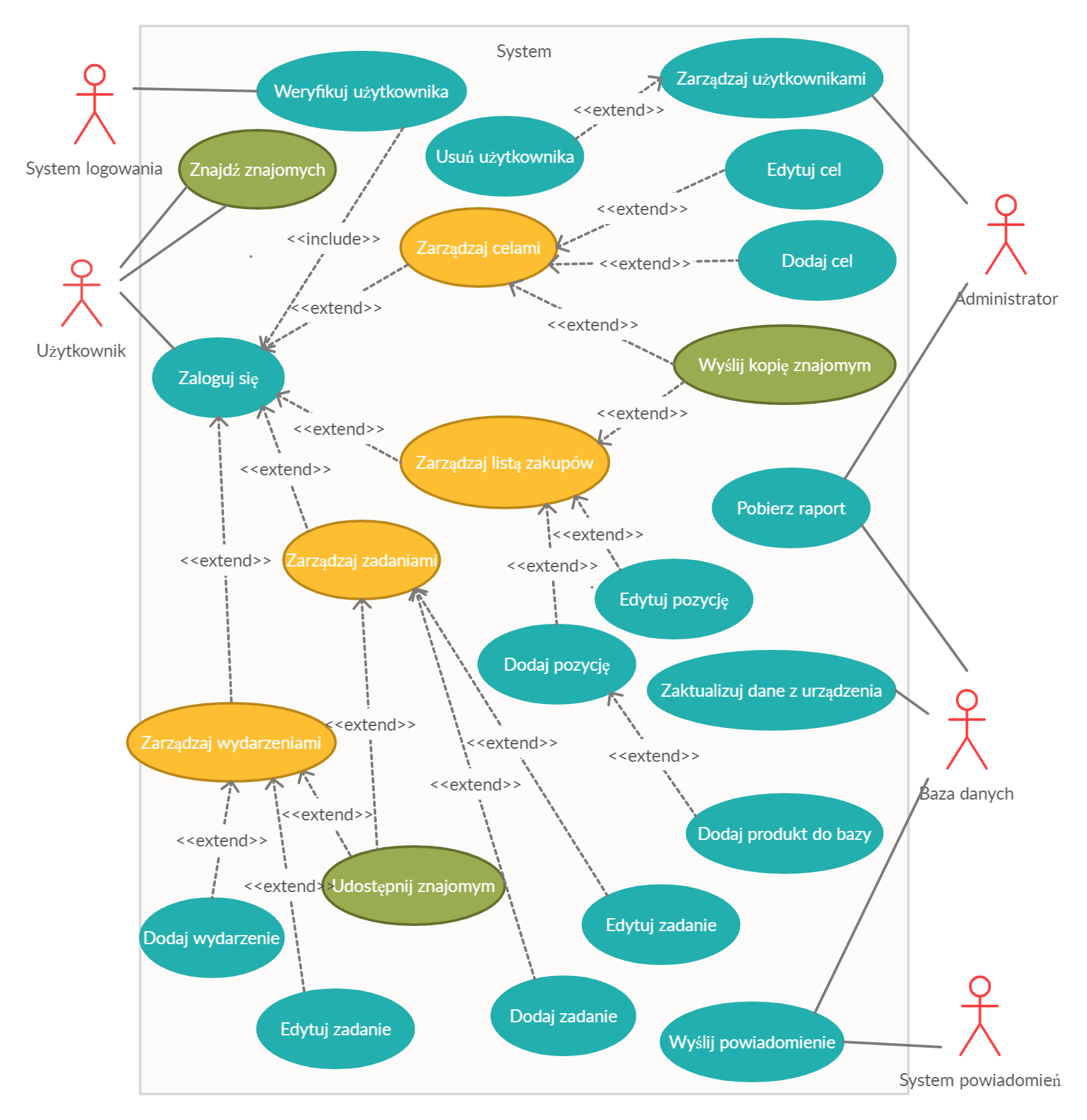
**1. Opis systemu**

Aplikacja mobilna na system Android nad którą pracujemy ma w założeniu służyć do pomocy w organizacji czasu użytkownika. Można w niej tworzyć wydarzenia i zadania oraz dowolnie edytować szczegóły planów (data, miejsce, powtarzalność, ew. linki, kategoria). Każde wydarzenie bądź zadanie może zostać podpięte pod system przypomnień urządzenia wysyłający przypomnienie z odpowiednim wyprzedzeniem. System posiada również możliwość ustalenia alarmu/budzika bez konieczności tworzenia wydarzenia.

Kolejną funkcjonalnością systemu jest możliwość ustalenia celów w postaci dobrych nawyków, które użytkownik chce zaadaptować np. czytanie 3x w tygodniu. Aplikacja zapewnia możliwość monitorowania postępów w celach i wysyła powiadomienie w razie ich niewypełniania. Dodatkowo każdy użytkownik posiada prywatną listę zakupów, dzięki której może na bieżąco wprowadzać do niej swoje potrzeby i odznaczać produkty kupione. Ponadto, dzięki możliwości wprowadzenia potrzebnej ilości i orientacyjnej kwoty produktu użytkownik uzyskuje informacje o szacowanym koszcie planowanych zakupów.

Każdy użytkownik systemu zostaje nim po uwierzytelnieniu przez zewnętrzny system logowania powiązany z kontem na innej platformie (np. Google). Każdy użytkownik posiada listę znajomych, których może zaprosić za pośrednictwem aplikacji. Wydarzenia i zadania można udostępniać znajomym do wglądu. Figurują one wtedy w Kalendarzu znajomych, również wszystkie zmiany dokonane przez autora wydarzenia/zadania. Cele i zakupy można natomiast wysłać dowolnemu znajomemu, który otrzyma ich kopię i będzie je edytować niezależnie od autora, gdyż przyjmujemy, że każdy wykonuje zakupy bądź cele sam. W przypadku wydarzeń i zadań przyjmujemy, iż osoba która je utworzyła jest w pewnym sensie przełożonym użytkowników, którym zleca zadania bądź uczestnictwo w wydarzeniach, więc to ona ma pełną kontrolę nad ich szczegółami.

Wszystkie informacje dotyczące danych użytkowników są przechowywane w zewnętrznej bazie danych i regularnie aktualizowane. Użytkownik posiada możliwość dostępu online do danych zapisanych w pamięci urządzenia, a wszelkie edycje zostaną zapisane zaraz po ich wprowadzeniu. W razie braku dostępu do Internetu można korzystać w trybie offline, jednak informacje zostaną zsynchronizowane po uzyskaniu połączenia Internetowego.

**2. Diagram przypadków użycia**

2.1 Aktorzy

* Administrator – osoba mająca największe uprawnienia w systemie, upoważniona do wglądu i edycji wszystkich elementów aplikacji, Waldemar. Może on dodawać/modyfikować/usuwać użytkowników, przeglądać raporty i inne statystyki.
* Użytkownik - osoba zainteresowana korzystaniem z aplikacji z poziomu użytkownika, może dowolnie modyfikować swoje wydarzenia, zadania, cele i listę zakupów. Posiada listę przyjaciół. Może udostępniać dowolnym przyjaciołom wydarzenia i zadania oraz wysyłać im kopie swoich zakupów i celów.
* System logowania – zewnętrzny system uwierzytelniający umożliwiający użytkownikowi zalogowanie się do aplikacji powiązując dane z kontem na innej platformie np. Google.
* Baza danych – serwer na którym znajdują się wszystkie informacje użytkowników umożliwiające korzystanie z konta z dowolnego urządzenia.
* System powiadomień – API Android która wysyła notyfikacje na pasek powiadomień.

2.3 Scenariusze przypadków użycia

|  |  |
| --- | --- |
| Nazwa | Zaloguj się |
| Opis | Pozwala na uzyskanie dostępu do menu głównego aplikacji. |
| Aktorzy | Użytkownik, system logowania |
| Podstawowy ciąg zdarzeń | 1. System wyświetla ekran logowania 2. Użytkownik, korzystając z UI, wybiera "third party app" za pośrednictwem którego potwierdzona zostanie tożsamość 3. Wysłane jest żądanie o autentyfikację do ww. pośrednika 4. Sygnał zwrotny potwierdza autentyfikację, użytkownik przenoszony jest do menu głównego |
| Alternatywy | * Sygnał zwrotny odmawia autentyfikacji, użytkownik pozostaje w menu logowania i wyświetlany jest komunikat o nieudanej autentyfikacji |

|  |  |
| --- | --- |
| Nazwa | Znajdź znajomych |
| Opis | Pozwala na zaproszenie innych użytkowników do przyjaciół |
| Aktorzy | Użytkownicy |
| Podstawowy ciąg zdarzeń | 1. Użytkownik przechodzi do ekranu zaproszeń 2. Użytkownik wysyła żądanie zaproszenia osoby poprzez jej identyfikator 3. Jeśli identyfikator istnieje, użytkownik otrzymuje powiadomienie o treści "zaproszenie wysłane" |
| Alternatywy | * Jeśli identyfikator nie istnieje, użytkownik otrzymuje powiadomienie o treści "nie znaleziono użytkownika" |

|  |  |
| --- | --- |
| Nazwa | Zarządzaj celami/wydarzeniami/zadaniami/zakupami |
| Opis | Umożliwia wgląd w plany i ich modyfikację |
| Aktorzy | Użytkownik |
| Podstawowy ciąg zdarzeń | 1. Użytkownik przechodzi do części aplikacji odpowiedzialnej za zarządzanie celami 2. System wyświetla wszystkie rekordy z danej kategorii przypisane do danego użytkownika. 3. Użytkownik wysyła żądanie akcji jaką chce przeprowadzić: może to być usunięcie/edycja wybranej pozycji bądź utworzenie nowej 4. Aplikacja oczekuje na przetworzenie żądania 5. Informacje na ekranie są aktualizowane |
| Alternatywy | * W przypadku błędnego żądania, podawany jest stosowny komunikat oraz wyświetlona informacja o źródle błędu |

|  |  |
| --- | --- |
| Nazwa | Wyślij/Udostępnij znajomym |
| Opis | Pozwala na wgląd znajomego dla udostępnienia oraz pobranie kopii dla wysłania. |
| Aktorzy | Użytkownicy |
| Podstawowy ciąg zdarzeń | 1. Użytkownik wybiera odpowiednią opcję w panelu zarządzania 2. Użytkownik wybiera rekord który chciałby przekazać, a następnie jest przenoszony do ekranu wysyłania 3. Użytkownik podaje identyfikator odbiorcy i wysyła żądanie 4. Jeśli identyfikator istnieje, użytkownik otrzymuje powiadomienie o treści "treść przekazana" 5. Użytkownik jest przenoszony do poprzedniego ekranu 6. Odbiorca otrzymuje powiadomienie o możliwości odbioru treści 7. W przypadku zatwierdzenie żądania, treści odbiorcy są odświeżane 8. Nadawca otrzymuje stosowne powiadomienie, jego treści również są odświeżane |
| Alternatywy | * Odmowa odbiorcy: nadawca zostaje powiadomiony przy najbliższym zalogowaniu * Identyfikator nie istnieje bądź odbiorca nie jest znajomym nadawcy i nie wyraził zgody na otrzymywanie udostępnień od obcych: nadawca otrzymuje stosowny komunikat, powrót do pkt 3 |

|  |  |
| --- | --- |
| Nazwa | Zaktualizuj dane z urządzenia |
| Opis | Zapewnia stałą obecność najnowszej wersji danych w chmurze |
| Aktorzy | Baza danych |
| Podstawowy ciąg zdarzeń | 1. Dopóki urządzenie nie ma dostępu do Internetu, system czeka 2. System porównuje różnice w bazie, oraz urządzeniu 3. System aktualizuje informacje w urządzeniu, według zmian w bazie danych 4. System aktualizuje informacje w bazie danych, według zmian w urządzeniu |
| Alternatywy |  |

|  |  |
| --- | --- |
| Nazwa | Wyświetl powiadomienie |
| Opis | Pozwala na otrzymywanie notyfikacji w terminie wybranych wydarzeń |
| Aktorzy | System powiadomień |
| Podstawowy ciąg zdarzeń | 1. System sprawdza dane na temat możliwych powiadomień 2. Jeśli informacje do wyświetlenia są dostępne, stwarza kanał powiadomień 3. Informacje do wyświetlenia zostaje przetworzone i stworzone zostaje odpowiedni obiekt notyfikacji 4. "Wypchany" zostaje obiekt przy użyciu kanału powiadomień 5. Po odpowiednim interwale proces się powtarza |
| Alternatywy | * Brak informacji do wyświetlenia: system czeka kolejny interwał i próbuje ponownie |

**3. Wymagania niefunkcjonalne**

3.1 Wymagania produktowe

* Intuicyjny interfejs korzystający z gestów **[multitouch]**
* Dostęp do kompletnych danych w trybie offline **[nie występujące braki w danych, możliwość wglądu w 100% danych dostępnych podczas ostatniej wizyty online]**
* Płynne **[<1s]** przechodzenie między ekranami i dynamiczne **[<2s]** ładowanie w nich danych z bazy
* Brak ekranów ładowania **[z pominięciem startu aplikacji]**
* Niezawodny system powiadomień **[w przypadku błędu systemu/restartu, proces powiadomień samodzielnie przywróci się w 100% przypadków]**
* Prosta obsługa aplikacji **[instrukcja mieszcząca się na 1 stronie dokumentu]**

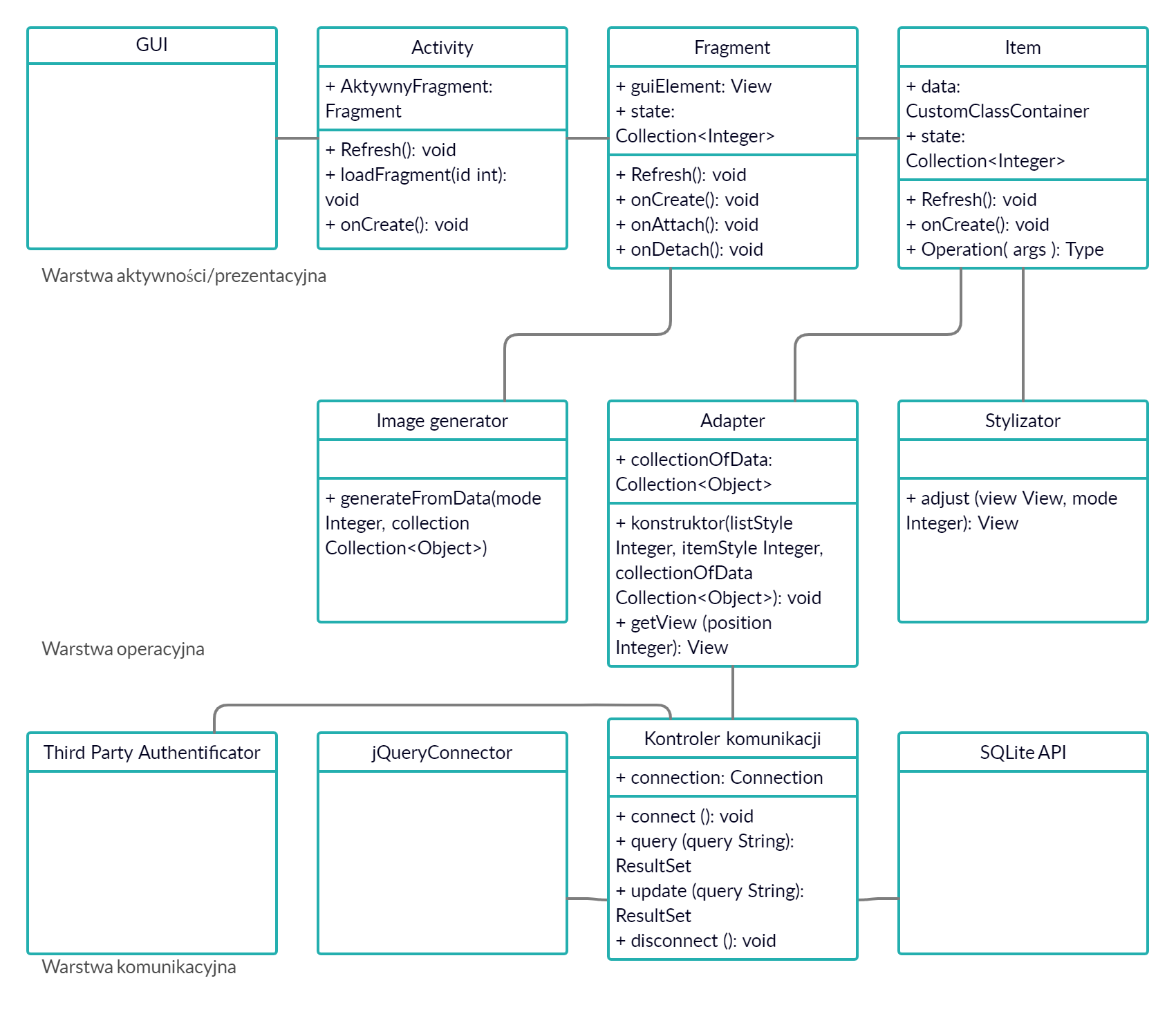
3.2 Wymagania organizacyjne

* Aplikacja zaimplementowana w języku **Java**, oraz **XML**
* Baza danych działać będzie w technologii SQL, korzystając z systemu **PostgresSQL**
* Aplikacja będzie kompatybilna z wersjami systemu operacyjnego **Android ≥ 6.0**

3.3 Wymagania zewnętrzne

* Proste i przejrzyste API bazy danych **[jasno określone polecenia, uniwersalna technologia, możliwość implementacji zewnętrznych dodatków]**

**4. Architektura systemu**

****

**5. Opis interfejsów**

1. **Połączenie systemu z zewnętrzną bazą danych** - realizowane za pośrednictwem konektora ‘jQuerryConnector’, używane w przekazywaniu danych użytkownika pomiędzy chmurą a systemem. Przekazuje informacje w formie rekordów bazodanowych.
2. **Połączenie systemu z pamięcią wewnętrzną urządzenia** - realizowane za pośrednictwem wbudowanego API Android, korzystając m.in. z klasy ‘SQLiteDatabase’, używane w przechowywaniu danych wprowadzonych offline, oraz odczyt danych, w przypadku braku połączenia z zewnętrzną bazą. Przekazuje informacje w formie rekordów bazodanowych.
3. **GUI** - Umożliwia użytkownikowi komunikację z systemem, a w dalszej kolejności, komunikację z bazami danych, zewnętrzną, oraz lokalną. Jest bezpośrednim połączeniem klienta z danymi i pozwala na dwukierunkowe przekazywanie informacji
4. **Third party authentificator** – potwierdza tożsamość użytkownika, odpowiada na żądanie logowania
5. **Activity v fragment** – aktywność przydziela kontekst oraz przechowuje mechanizm przejścia fragmentów
6. **Fragment v item** – fragment inicjuje kontenery zarządzające itemami oraz przechowuje logikę związaną z adapterami.
7. **Fragment v image generator** – fragment inicjuje odpowiedni obiekt image generatora oraz przydziela mu zadania
8. **Item v stylizator** – item wnioskuje o edycję atrybutów w zależności od zadanego parametru
9. **Item v adapter** – adapter korzystając z danych uzyskanych przez kontroler komunikacji przydziela dane odpowiedniemu child itemowi
10. **Adapter v kontroler komunikacji** – adapter przekazuje treść komendy oraz rodzaj żądania, które ma być wykonane na bazie danych

**6. Lista technologii**

* **Git** - Najpopularniejsza technologia używana w tworzeniu repozytoriów, w której zespół ma największe doświadczenie
* **Java 11** - W przeciwieństwie do odpowiednika “Kotlin”, jest to znana zespołowi technologia
* **XML** - Używany w tworzeniu aplikacji Android, służy do prostej stylizacji obiektów, jedna z podstawowych funkcjonalności programowania Android
* **Biblioteki Android** - Niezbędne w tworzeniu natywnej aplikacji dla ww. systemu
* **jQueryConnector** - Najpopularniejsza biblioteka używana w komunikacji z zewnętrznymi bazami danych, bez innych dobrze udokumentowanych alternatyw
* **PSQL** - Język obsługi zewnętrznej bazy danych, najlepiej znana zespołowi technologia w danej dziedzinie
* **SQLite** - Język obsługi lokalnej bazy danych, najpopularniejsza i najlepiej udokumentowana technologia dla bazy lokalnej

**7. Projekt testów**

* **jUnit** - testy jednostkowe na bazach danych
* **testowanie manualne** - testy na UI, wydajności, oraz funkcjonalności
* **Kraken mobile** – użyjemy go do testów typu end to end
* **Kobiton** – performance test

**8. Analiza ryzyka**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Zaniedbywalne | Marginalne | Krytyczne | Katastrofalne |
| Pewne | Podwyższenie cen usług dostawcy | Downtime zewnętrznej bazy |  |  |
| Prawdopodobne | Powolne działanie urządzenia | Kłótnie w zespole |  |  |
| Możliwe | Niewygodny interface użytkownika | Brak komunikacji systemu z ekranem | Błędy w logice biznesowej systemu | Ataki DDOS na bazę danych |
| Mało prawdopodobne |  | Utrata wsparcia jednego z narzędzi | Choroba kluczowego członka zespołu | Konflikt oprogramowania |
| Prawie niemożliwe |  |  | Bus risk | Wyciek danych |

* **Podwyższenie cen usług dostawcy** – prowadzenie analiz rynku i dobieranie najlepszych ofert
* **Downtime zewnętrznej bazy danych** – rozważenie zmiany dostawcy usług, jeżeli sytuacja będzie się powtarzać lub trwać zbyt długo
* **Powolne działanie urządzenia** – wprowadzenie optymalizacji
* **Kłótnie w zespole** – rozstrzyganie sporów na drodze mediacji
* **Niewygodny interface użytkownika** – zastosowanie się do sugestii użytkowników
* **Brak komunikacji systemu z ekranem** – analiza logów w celu ustalenia przyczyny błędu i jej rozwiązania
* **Błędy w logice biznesowej systemu** – debuggowanie kodu
* **Ataki DDOS na bazę danych –** przejście na architekturę rozproszoną
* **Utrata wsparcia jednego z narzędzi** – zawsze możemy znaleźć inną, wciąż wspieraną alternatywę
* **Choroba kluczowego członka zespołu** – dobra dokumentacja, komentarze umożliwiające przejęcie pracy
* **Konflikt oprogramowania** – zrezygnowanie z jednej z konfliktujących technologii oraz odpowiedni dobór zamiennika
* **Bus risk** – prowadzenie odpowiedniej dokumentacji
* **Wyciek danych** - istotne wzmocnienie zabezpieczeń, rebranding

**9. Lista narzędzi**

* Repozytorium zdalne w systemie “GitHub”,
* Serwis hostingowy “Gearhost.com”,
* Środowisko programistyczne “Android Studio”,
* Telefon z systemem Android, wraz z odpowiednim okablowaniem służący w testowaniu aplikacji,
* Wbudowany emulator Android studio
* Aplikacja “Trello”