



**Politechnika
Śląska**

PROJEKT INŻYNIERSKI

„System zarządzania lokalem gastronomicznym”

Jakub Górka
Nr albumu 295796

Kierunek: Informatyka
Specjalność: <wpisać właściwą>

PROWADZĄCY PRACĘ
dr hab. inż. Piotr Gaj
KATEDRA Systemów Rozproszonych i Urządzeń Informatyki
Wydział Automatyki, Elektroniki i Informatyki

GLIWICE Rok 2023

Tytuł pracy:

System zarządzania lokalem gastronomicznym

Streszczenie:

Praca koncentruje się na stworzeniu aplikacji mobilnej przeznaczonej do efektywnego zarządzania restauracją. Analiza dziedziny, a następnie opis wymagań funkcjonalnych oraz niefunkcjonalnych przygotowuje do implementacji aplikacji. Specyfikacja wewnętrzna oraz specyfikacja zewnętrzna przedstawiają sposób implementacji aplikacji oraz scenariusze korzystania z funkcjonalności. Projekt został zakończony testami, które opisane zostały w ostatnim rozdziale.

Słowa kluczowe:

Aplikacja mobilna, system zarządzania, gastronomia

Thesis title:

Restaurant management system

Abstract:

---- Uzupełnić po zatwierdzeniu polskiego streszczenia

(Thesis abstract – to be copied into an appropriate field during electronic submission, in English.)

Keywords:

Mobile application, management system, gastronomy

Spis treści

Spis treści

ROZDZIAŁ 1	WSTĘP	7
1.1	Cel pracy	7
1.2	Zakres pracy	7
1.3	Wkład autora	8
ROZDZIAŁ 2	ANALIZA DZIEDZINY	9
2.1	Kontekst problemu	9
2.2	Znane rozwiązania	9
ROZDZIAŁ 3	WYMAGANIA I NARZĘDZIA	13
3.1.	Wymagania funkcjonalne	13
3.2.	Wymagania нефункционалне	15
3.3.	Przypadki użycia	16
3.4.	Narzędzia	17
ROZDZIAŁ 4	SPECYFIKACJA ZEWNĘTRZNA	19
4.1	Wymagania sprzętowe	19
4.2	Sposób instalacji oraz aktywacji	19
4.3	Kategorie użytkowników	19
4.3.1	Gość	19
4.3.2	Użytkownik zalogowany	20
4.3.3	Administrator	20
4.4	Sposób obsługi	20
4.4.1	Logowanie	20
4.4.2	Utworzenie nowej restauracji	21
4.4.3	Dodanie wpisu do raportu z wybranego dnia	22
4.4.4	Rozpoczęcie oraz zakończenie dnia pracy	25
4.4.5	Sprawdzenie sumy przepracowanych godzin	27
4.4.6	Dodanie nowej faktury do Menadżera Faktur	28

ROZDZIAŁ 5 SPECYFIKACJA WEWNĘTRZNA	30
5.1 Przedstawienie Idei	30
5.2 Architektura systemu	30
5.2.1 Aplikacja mobilna	30
5.2.2 Baza danych	30
5.3 Opis struktury danych	31
ROZDZIAŁ 6 WERYFIKACJA I WALIDACJA	32
6.1 Testowanie	32
6.2 Walidacja	34
6.3 Wykryte i usunięte błędy	35
ROZDZIAŁ 7 PODSUMOWANIE I WNIOSKI	38
7.1 Możliwe kierunki rozwoju aplikacji	38
BIBLIOGRAFIA	40
SPIS SKRÓTÓW I SYMBOLI	44
ŹRÓDŁA	45
LISTA DODATKOWYCH PLIKÓW, UZUPEŁNIAJĄCYCH TEKST PRACY	46
SPIS RYSUNKÓW	47

Rozdział 1

Wstęp

W obecnych czasach technologia jest wszechobecna, towarzyszy nam na każdym kroku i przenika każdy aspekt naszego życia. Nie inaczej jest w branży gastronomicznej, przedsiębiorcy stale rozglądają się za nowościami, które uprosczą pracę lub przełożą się na wydajność prowadzenia firmy. Praca poświęcona jest stworzeniu systemu zarządzania lokalem gastronomicznym, który w założeniu ma na usprawniać procesy związanych z prowadzeniem firmy – zbieraniem danych o zarobkach, wydatkach, zarządzaniem personelem. Aplikacja została zaprojektowana w taki sposób aby mogła być wykorzystywana w szeroko pojętej grupie HoReCa¹, sprawdzi się zarówno w małej gastronomii, dużych restauracjach, gastronomii sieciowej oraz w gastronomii, która jest częścią większego lokalu usługowego np. stacje paliw, kasyna. Zdalny dostęp do danych za pomocą aplikacji pozwala na zarządzanie wieloma lokalami jednocześnie na odległość.

1.1 Cel pracy

Celem pracy jest zaprojektowanie i implementacja aplikacji, która umożliwi efektywne zarządzanie lokalem gastronomicznym przez zbieranie a następnie prezentację danych związanych z utargami, wydatkami – codziennymi oraz zbieranie danych o fakturach, czasem pracy pracowników. Umożliwi to właścicielom oraz menadżerom zdalne zarządzanie i analizę działania firmy.

1.2 Zakres pracy

W ramach pracy zostaną dokładnie przeanalizowane wymagania funkcjonalne oraz niefunkcjonalne. Następnie przeprowadzone zostaną etapy projektowania, implementacji, a następnie testowania.

1.3 Wkład autora

Autor zaprojektował w pełni działającą aplikację przy użyciu języka Swift oraz SwiftUI. Aplikacja była jednocześnie testowana w rzeczywistej restauracji z 14 użytkownikami – pracownikami restauracji.

Rozdział 2

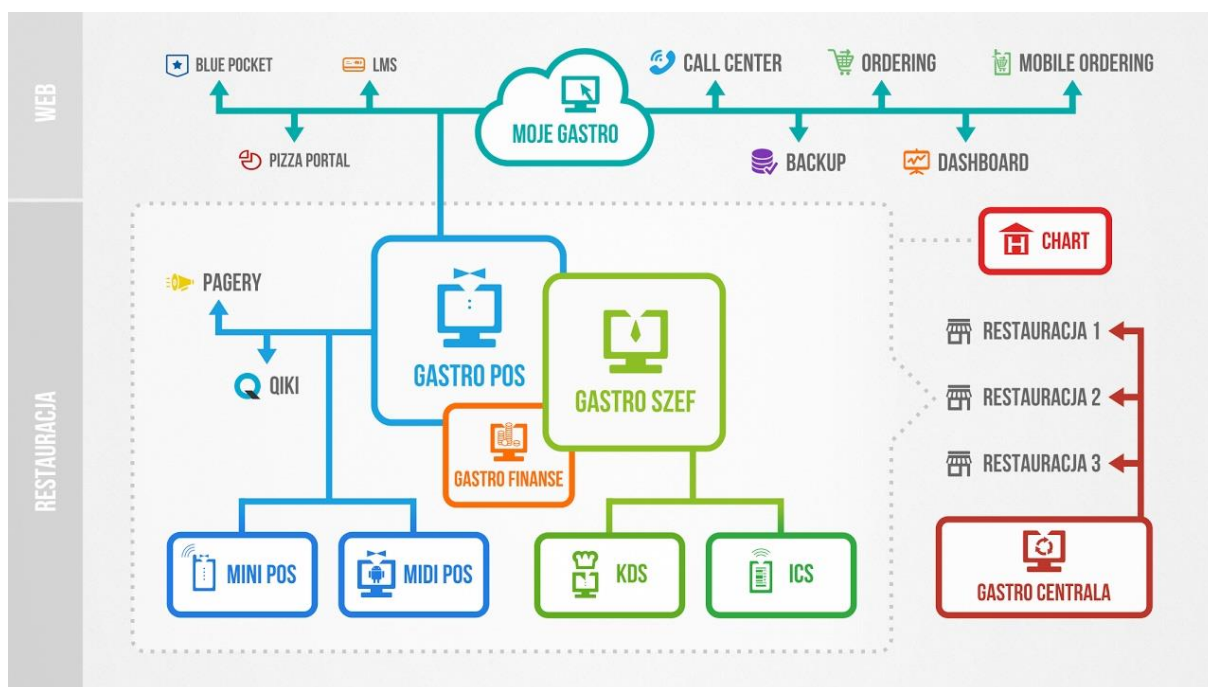
Analiza dziedziny

2.1 Kontekst problemu

Wprowadzanie do restauracji nowoczesnych narzędzi informatycznych, takich jak np. kioski zamówień obsługiwane przez samodzielnie przez klientów, pozwalają na zautomatyzowanie wielu procesów, co przekłada się na oszczędność czasu. W stale rozwijającej się branży gastronomicznej ważne jest aby maksymalizować wydajność, a dzięki temu minimalizować wydatki. Ograniczenie zbędnych kosztów pozwala na konkutowanie z innymi lokalami gastronomicznymi będącymi na rynku pod względem ceny jednocześnie utrzymując jakość produktów oraz usług na najwyższym poziomie. Przedstawiana aplikacja upraszcza kontrole wydatków przez właściciela lub menadżera, który w prosty i szybki sposób może zdalnie sprawdzić wydatki lokalu w danym dniu oraz czas pracy każdego pracownika. Faktury często przemieszczają się między lokalem, a firmą odpowiedzialną za księgowość w naszej firmie, przez co wiele dokumentów jest niepotrzebnie kserowanych. Aplikacja z menadżerem faktur, w którym możemy wprowadzić przychodzące do naszej firmy dokumenty pozwala na wgląd w nasze dokumenty w każdym momencie oraz łatwe zarządzanie płatnościami. Na rynku istnieje wiele podobnych rozwiązań, niestety wiele z nich jest niekompletna lub przestarzała. W celu uproszczenia zarządzania wymagane jest korzystanie z wielu produktów, które często nie są dostosowane do dzisiejszych realiów gastronomi, są nieintuicyjne w obsłudze, niedostosowane do urządzeń mobilnych, w dodatku drogie.

2.2 Znane rozwiązania

Najpopularniejszym rozwiązaniem na rynku jest rodzina produktów GASTRO od LSI Software, która jest na rynku od 1993 roku. „GASTRO POS”, czyli jeden z produktów rodziny GASTRO to pierwszy polski program gastronomiczny obsługiwany przez ekrany dotykowy. Firma podaje, że jest obecna w 18 tysiącach lokali gastronomicznych.



Rysunek 2.1: Grafika przedstawiająca rodzinę produktów GASTRO

GASTRO POS – to program sprzedaży, który umożliwia wybór pozycji towarowych, kompletowanie zamówień oraz realizację sprzedaży



Rysunek 2.2: Grafika przedstawia menu główne programu GASTRO POS

GASTRO SZEf – jest rozszerzeniem funkcjonalności programu GASTRO POS, na podstawie sprzedaży zrealizowanej na stanowiskach sprzedażowych umożliwia m.in. generowanie raportów utargów, wydatków, sprzedaży konkretnych produktów.



Rysunek 2.3: Grafika przedstawia wygląd programu GASTRO SZEf

mojeGASTRO – rozszerzenie produktu GASTRO SZEf, umożliwia zdalny dostęp do danych zebranych w GASTRO SZEf poprzez dedykowaną stronę internetową.



Rysunek 2.4: Grafika przedstawia wygląd strony internetowej mojeGASTRO

Zalety:

- bardzo rozbudowana lista produktów rozszerzająca możliwości systemu,
- producent zapewnia wsparcie oprogramowania od 30 lat.

Wady:

- bardzo wysoki próg startowy przy chęci rozpoczęcia korzystania z oprogramowania GASTRO, do korzystania z systemu niezbędne zakupienie urządzeń – m.in. komputer, monitory dotykowe, drukarki zamówień,
- wysoki koszt zakupu oprogramowania (ceny aktualne na rok 2023) - odpowiednio dla produktów: GASTRO SZEFE około 2200zł, GASTRO POS około 2200zł,
- jeżeli istnieje zainteresowanie zdalnym dostępem za pomocą mojeGASTRO wymagana jest dodatkowa comiesięczna płatna subskrypcja,
- płatna aktualizacja oprogramowania, w celu aktualizacji niezbędne jest wezwanie informatyka firmy pośredniczącej w sprzedaży programów, które jest dodatkowym kosztem
- brak aplikacji mobilnej przeznaczonej do zarządzania lokalem gastronomicznym.

Rozdział 3

Wymagania i narzędzia

W niniejszym rozdziale przedstawione zostały wymagania funkcjonalne oraz niefunkcjonalne. Dodatkowo, omówiony został model przypadków użycia, ukazując scenariusze, w jakich użytkownicy mogą korzystać z aplikacji. Ostatecznie, opisane zostały narzędzia, jakie zostały wykorzystane podczas implementacji projektu.

3.1. Wymagania funkcjonalne

Wymagania funkcjonalne definiują konkretne funkcjonalności, jakie system musi spełniać, aby sprostać oczekiwaniom użytkowników i osiągnąć zamierzone cele projektowe. Kluczowe funkcjonalności projektu obejmują:

1) Obsługa użytkowników

a) Możliwość rejestracji użytkownika

System umożliwia potencjalnym użytkownikom dokonanie rejestracji, wymagane jest wprowadzenie: imienia, nazwiska, adresu email, hasła.

b) Możliwość zalogowania użytkownika

Zarejestrowani użytkownicy mają możliwość zalogowania się do swojego konta, aplikacja zapamiętuje dane logowania po zamknięciu aplikacji.

2) Zarządzanie restauracjami

a) Prezentacja listy restauracji, do których należy użytkownik

Użytkownikowi prezentowane jest menu z listą restauracji, do których użytkownik przynależy, wybiera restaurację, której szczegóły chce przeglądać.

b) Możliwość stworzenia nowej restauracji

Użytkownik tworzy nową restaurację o wybranej nazwie, po stworzeniu dostaje rolę administratora restauracji.

- c) Możliwość przyjęcia lub odrzucenia zaproszenia do dołączenia do restauracji
Na liście restauracji pojawia się zaproszenie do restauracji, użytkownik może zaproszenie przyjąć lub odrzucić.

3) Zarządzanie pracownikami

- a) Prezentacja listy pracowników
System umożliwia wyświetlenie listy wszystkich pracowników przypisanych do danej restauracji.
- b) Możliwość zaproszenia użytkownika do istniejącej restauracji
Administrator restauracji ma możliwość zaproszenia nowego użytkownika do restauracji.
- c) Wyświetlenie listy dni, w których wybranego miesiąca pracownik był w pracy
Użytkownik może sprawdzić historie obecności pracownika w pracy, z informacjami o czasie rozpoczęcia pracy, czasie zakończenia pracy, sumie przepracowanych godzin.
- d) Wyświetlenie sumy przepracowanych godzin w wybranym miesiącu
System podsumowuje liczbę przepracowanych godzin w wybranym miesiącu.

4) Raporty dzienne

- a) Wybranie dnia, z którego wyświetla się raport
Użytkownik ma możliwość wybrania dnia, z którego chce przejrzeć raport.
- b) Możliwość dodania informacji o aktualnym utargu
Użytkownik ma możliwość dodanie danych o aktualnym utargu, w systemie zapisywane jest kto dodał informacje, oraz o której godzinie.
- c) Wyświetlenie listy wprowadzonych danych o utargu
W aplikacji prezentowana jest lista przedstawiająca wprowadzone wcześniej dane
- d) Możliwość dodania nowego wydatku z kasy lub wpłaty do kasy
Użytkownik może wprowadzić dane o nowym wydatku lub wpłacie do kasy, wymagane podanie tytułu wpłaty/wypłaty, kwoty, możliwość dodania dodatkowego opisu, system zapisuje też przez kogo zostały wprowadzone dane oraz o której godzinie.
- e) Wyświetlenie listy wpłat oraz wydatków
W aplikacji prezentowana jest lista wprowadzonych wcześniej wpłat oraz wydatków
- f) Możliwość rozpoczęcia dnia pracy
Pracownik może rozpocząć swoją pracę, od momentu rozpoczęcia pracy liczony jest czas pracy. W celu rozpoczęcia pracy pracownik musi zrobić zdjęcie

przedstawiające swoją twarz, co umożliwia potwierdzenie obecności w miejscu pracy.

- g) Wyświetlenie listy pracowników danego dnia

Wyświetlana jest lista prezentująca którzy pracownicy danego dnia byli w pracy

5) Menadżer faktur

- a) Wyświetlanie listy przyjętych w wybranym miesiącu faktur z podziałem na firmy.

Prezentacja listy faktur przyjętych w wybranym miesiącu, z podziałem na firmy.

O każdej fakturze prezentowana jest informacja o dacie wystawienia, dacie płatności, kwocie, zdjęcie faktury

- b) Dodanie nowej faktury do listy

Użytkownik ma możliwość wprowadzenia nowej faktury, wymagane są informacje o dacie wystawienia faktury, dacie płatności, kwocie, dodatkowo możliwość wprowadzenia dodatkowego opisu oraz zrobienia zdjęcia faktury.

3.2. Wymagania niefunkcjonalne

Wymagania niefunkcjonalne to cechy i ograniczenia systemu, które nie dotyczą bezpośrednio jego funkcjonalności, ale wpływają na jakość, wydajność, niezawodność, bezpieczeństwo itp. Opisują one, w jaki sposób system działa lub jakie posiada cechy.

1) Wydajność bazy danych

System powinien optymalizować zapytania do bazy danych w celu nieprzekraczania założonego limitu 50 tysięcy zapytań dziennie.

System może obsługiwać 200 użytkowników jednocześnie.

2) Optymalizacja zapytań

Kod aplikacji powinien być zoptymalizowany pod kątem minimalizacji wysyłanych zapytań do bazy danych, w celu otrzymania wyników w najkrótszym możliwie czasie.

3) Możliwość rozszerzenia

W przypadku rozwoju aplikacji usługa bazy danych powinna być gotowa do łatwego rozszerzenia w celu obsługi większej ilości zapytań

4) Użyteczność

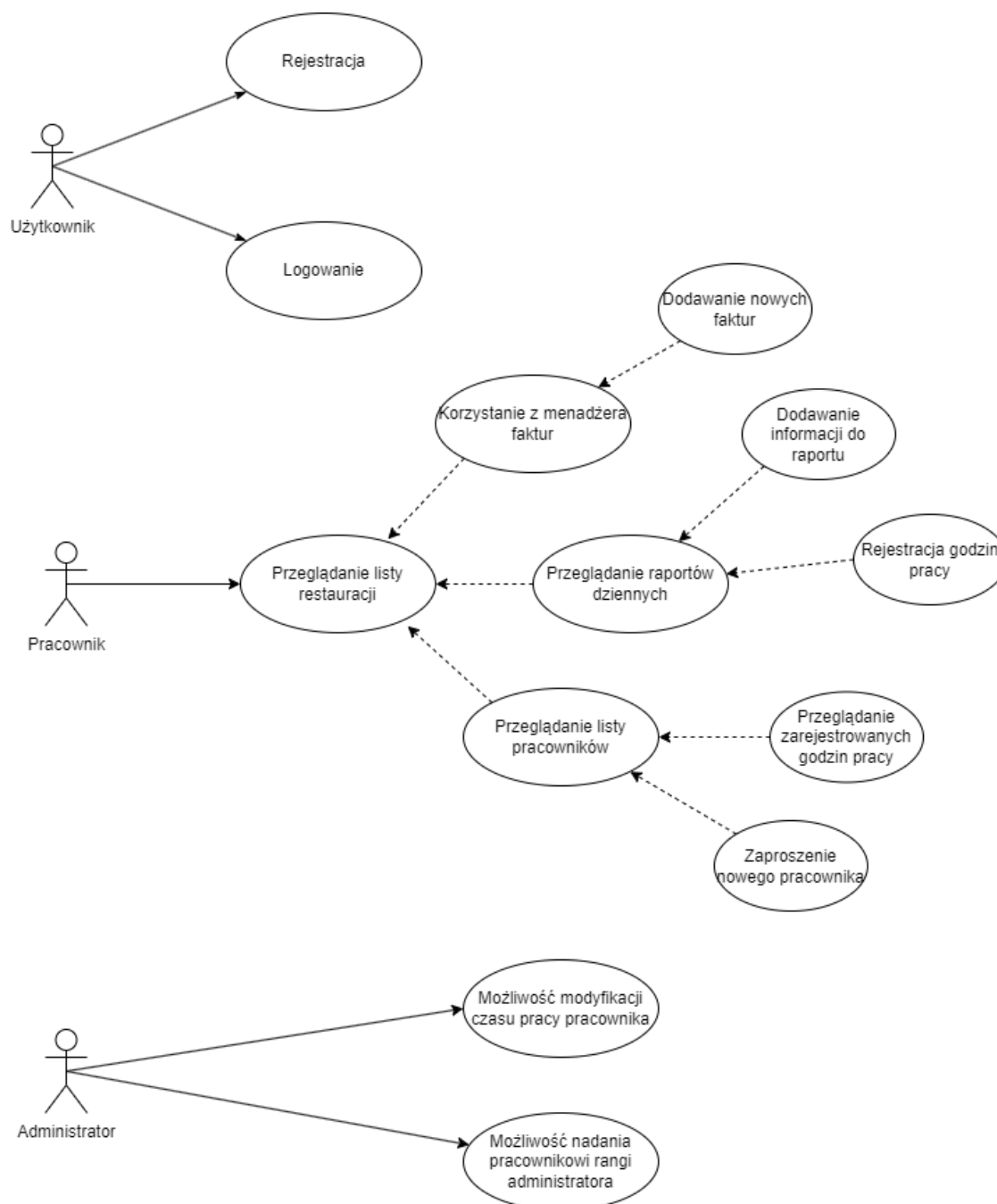
Interfejs użytkownika powinien być prosty oraz intuicyjny do obsługi dla każdego użytkownika bez potrzeby przeprowadzania szkolenia z przedstawiającego działanie aplikacji.

5) Dostępność usług Firebase

System powinien uwzględniać dostępność usług Firebase (takich jak Firestore) i w przypadku ewentualnych przerw w świadczeniu tych usług, informować użytkowników o możliwych zakłóceniach w korzystaniu z aplikacji.

3.3. Przypadki użycia

Diagram przypadków użycia to narzędzie, które przedstawia interakcje pomiędzy różnymi aktorami a systemem. Jest używany do zidentyfikowania, zrozumienia i opisania funkcji oraz interakcji, jakie zachodzą między różnymi uczestnikami, a systemem. Ten rodzaj diagramu pomaga w definiowaniu funkcji systemu z perspektywy użytkownika, ukazując, jak system będzie używany w praktyce.



Rysunek 3.1: Diagram przypadków użycia

3.4. Narzędzia

W trakcie implementacji projektu wykorzystano szereg narzędzi i technologii, które wspierały proces tworzenia oraz testowania aplikacji. Poniżej przedstawiono główne narzędzia użyte w ramach pracy nad projektem:

1) Xcode

Xcode to oficjalnie środowisko programistyczne Apple Inc., przeznaczone do tworzenia aplikacji na systemy iOS, iPadOS, macOS, watchOS, tvOS. Środowisko oferuje kompleksowe wsparcie dla Swift oraz SwiftUI.

2) Swift

Swift to potężny i nowoczesny język programowania stworzony przez Apple, który zdobył popularność dzięki swojej przejrzystej i czytelnej składni.

Wprowadzony w 2014 roku jako następca Objective-C, Swift jest zoptymalizowany pod kątem wydajności, co sprawia, że jest efektywny zarówno dla tworzenia aplikacji na platformy iOS, iPadOS, macOS, watchOS i tvOS.

Wykorzystane Biblioteki:

- FirebaseDatabase – biblioteka zawierająca funkcje umożliwiające współpracę z bazą danych na platformie Firebase
- Firebase Auth – biblioteka służąca do obsługi autentykacji użytkowników, obsługa rejestracji oraz logowania.

3) SwiftUI

SwiftUI to framework do tworzenia interfejsów użytkownika UI stworzony przez Apple Inc. w celu tworzenia aplikacji na platformy Apple (iOS, iPadOS, macOS, watchOS, tvOS). Został zaprezentowany w 2019 roku i od tego czasu jest stale rozwijany. SwiftUI wyróżnia się tym, że framework zajmuje się implementacją interfejsu, który został przez dewelopera opisany za pomocą kodu.

4) Firebase

Firebase to produkt, który powstał w 2011 roku jako platforma do tworzenia aplikacji mobilnych, w 2014 roku przejęty przez firmę Google, stale rozwijany skupiony na integracji w innych produktach Google.

a. Firestore Database

Usługa bazy danych, przechowuje dane w formacie przypominającym JSON, dzięki czemu jest prosty do zrozumienia. Obsługuje wiele platform dzięki czemu umożliwia tworzenia produktów pod wiele systemów operacyjnych.

b. Firebase Authentication

Usługa autentykacji wprowadzona przez Firebase, udostępnia wiele metod logowania m.in. logowanie za pomocą konta Google, Apple, Facebook, Github i wiele innych. Zapewnia możliwości takie jak weryfikacja e-mail, resetowanie hasła oraz zapobieganie nieautoryzowanemu dostępowi.

5) Apple App Store Connect

Platforma firmy Apple, służy do zarządzania procesem przesyłania stworzonej aplikacji do sklepu App Store. Umożliwia monitorowanie statystyk, interakcje z użytkownikami oceniającymi aplikacje w sklepie oraz udostępnianie wersji beta.

6) Apple Testflight

Umożliwia deweloperom przeprowadzenie testów beta swoich aplikacji przed oficjalnym wypuszczeniem do sklepu App Store. Platforma jest zintegrowana z App Store Connect, pozwala udostępniać wybraną wersję do testów dla konkretnych użytkowników. Testerzy mają możliwość przesyłania opinii oraz zgłaszania błędów, system zbiera dane o awariach aplikacji.

7) Github

GitHub został założony w 2008 roku przez Chrisa Wanstratha i Toma Prestona. Od tego czasu stał się jedną z najpopularniejszych platform do hostowania projektów opartych na systemie kontroli wersji Git.

Rozdział 4

Specyfikacja zewnętrzna

W rozdziale przedstawiono wymagania sprzętowe niezbędne do uruchomienia aplikacji oraz sposób instalacji aplikacji. Opisane zostały również kategorie użytkowników występujących w systemie oraz sposób obsługi.

4.1 Wymagania sprzętowe

Aplikacja przeznaczona jest dla smartfonów firmy Apple Inc.

W celu pobrania oraz uruchomienia aplikacji niezbędne jest posiadanie urządzenia z oprogramowaniem iOS 16.0 lub nowszym oraz posiadanie konta Apple ID w celu zalogowania się z sklepie systemowym App Store.

Do korzystania z aplikacji wymagane jest stabilne połączenie z internetem.

4.2 Sposób instalacji oraz aktywacji

Aplikację można pobrać i zainstalować z systemowego sklepu Apple App Store. Po zainstalowaniu w celu uzyskania dostępu do funkcjonalności wymagane jest zalogowanie się lub utworzenie nowego konta.

4.3 Kategorie użytkowników

Aplikacja wyróżnia trzy kategorie użytkowników, którzy mają dostęp do różnych funkcjonalności aplikacji

4.3.1 Gość

Osoba, która nie zalogowała się, ma dostęp do funkcjonalności rejestracji oraz logowania.

4.3.2 Użytkownik zalogowany

Zalogowani użytkownicy posiadają uprawnienia do większości funkcjonalności zapewnianych przez aplikację m.in. dodawanie raportów dziennych, przeglądanie przepracowanych godzin, dodawanie faktur do menadżera faktur.

4.3.3 Administrator

Konto administratora rozszerza funkcjonalności konta zalogowanego użytkownika. Administrator dodatkowo ma możliwość m.in. dodawania użytkowników do restauracji, modyfikacji czasu pracy pracownika w danym dniu.

4.4 Sposób obsługi

W niniejszym podrozdziale przedstawione zostaną kluczowe procesy obsługi restauracji.

4.4.1 Logowanie

Po uruchomieniu aplikacji użytkownikowi ukazuje się ekran logowania (rys 4.1), w celu zalogowania wymagane jest uzupełnienie adresu email, hasła (rys 4.2), a następnie zalogowanie zielonym przyciskiem. Po pomyślnym zalogowaniu ukazuje się lista restauracji.



Rysunek 4.1: Ekran logowania widoczny po uruchomieniu aplikacji



Rysunek 4.2: Ekran logowania po uzupełnieniu danych logowania

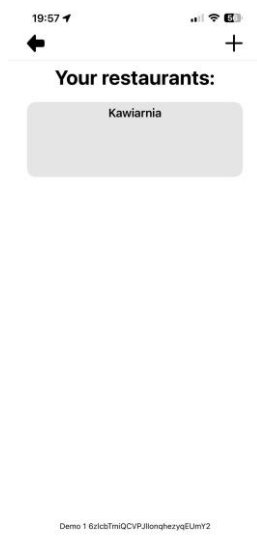


Rysunek 4.3: Ekran listy restauracji widoczny po pomyślnym zalogowaniu

Aplikacja zapamiętuje dane logowania, jeżeli użytkownik nie wyloguje się ręcznie, po ponownym uruchomieniu aplikacji system automatycznie zaloguje się na poprzednio używane konto.

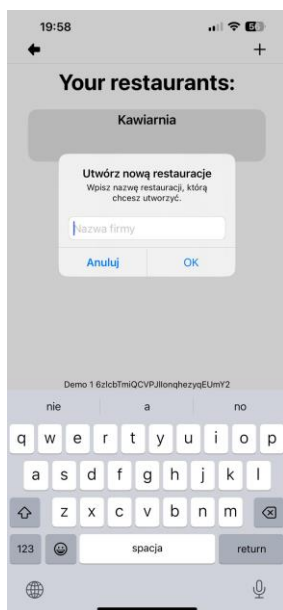
4.4.2 Utworzenie nowej restauracji

W celu dodania nowej restauracji użytkownik musi nacisnąć przycisk „plus” w prawym górnym rogu ekranu nad listą restauracji (rys 4.4).

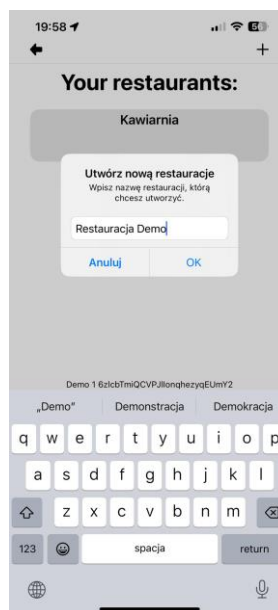


Rysunek 4.4: Widok listy restauracji – przycisk dodania restauracji w prawym górnym rogu

Następnie należy wpisać nazwę nowej restauracji (rys 4.5 oraz rys 4.6).

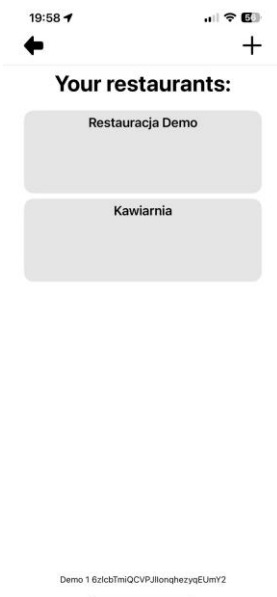


Rysunek 4.5: Okienko dodawania restauracji – wymagane uzupełnienie nazwy



Rysunek 4.6: Okienko dodawania restauracji – po uzupełnieniu nazwy

Po zatwierdzeniu na liście restauracji pojawia się utworzona przez nas restauracja (rys 4.7).



Rysunek 4.7: Widok listy restauracji po dodaniu nowej restauracji.

4.4.3 Dodanie wpisu do raportu z wybranego dnia

W celu dodania wpisu do raportu wybieramy interesującą nas kategorię: utarg, wydatki lub wpłaty (Rys 4.8).



Rysunek 4.8: Główny widok raportu, widoczne kategorie

Użytkownikowi wyświetla się lista wcześniej dodanych np. wydatków, aby dodać nowy wykorzystujemy przycisk „Dodaj” (Rys 4.9).



Rysunek 4.9: Lista wydatków

Następnie wymagane jest uzupełnienie danych oraz potwierdzenie dodania za pomocą przycisku „Potwierdź” (Rys 4.10 oraz Rys 4.11).

Rysunek 4.10: Formularz dodania nowego wydatku przed uzupełnieniem

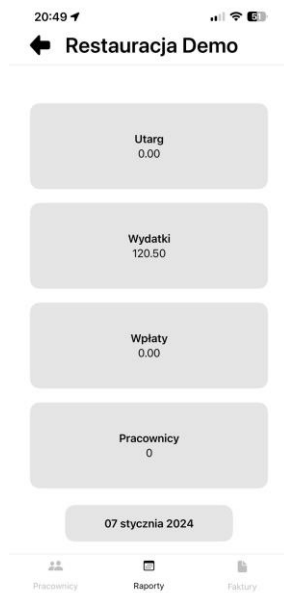
Rysunek 4.11: Formularz dodania nowego wydatku po uzupełnieniu

Dodany przez użytkownika wydatek wraz z wszystkimi wprowadzonymi danymi widoczny jest na liście. Widnieje również informacja, przez kogo zostały wprowadzone dane (Rys 4.12).



Rysunek 4.12: Lista wydatków – widoczny nowo dodany wydatek

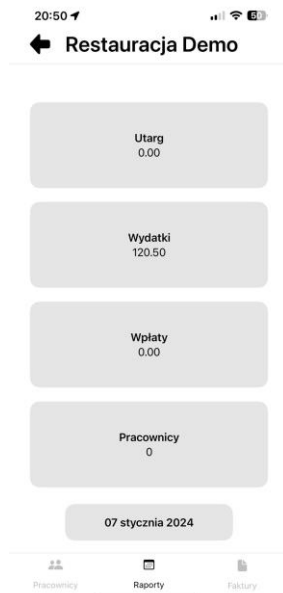
Aktualizuje się również główny widok raportu, przy kategorii „Wydatki” widoczna jest zaktualizowana suma wydatków (Rys 4.13)



Rysunek 4.13: Główny widok raportu po wprowadzeniu nowego wydatku

4.4.4 Rozpoczęcie oraz zakończenie dnia pracy

Użytkownik w celu rozpoczęcia swojej pracy musi przejść do kategorii „Pracownicy”, którą znajdzie na głównym widoku raportu (Rys 4.15).



Rysunek 4.15: Główny widok raportu

Użytkownikowi ukazuje się lista pracowników, którzy rozpoczęli już swoją pracę w danym dniu (Rys 4.16). W celu rozpoczęcia dnia użytkownik korzysta z przycisku „Rozpocznij pracę”.



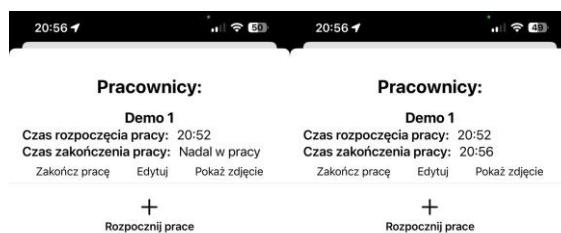
Rysunek 4.16: Widok listy pracowników

Uruchamia się kamera selfie telefonu, z którego korzysta użytkownik na górze przedstawiona jest informacja, o kto i o której rozpoczyna pracę (Rys 4.17). Po zrobieniu zdjęcia za pomocą przycisku na dole ekranu rozpoczyna się liczyć czas pracy.



Rysunek 4.17: Widok kamery telefonu

Użytkownik pojawia się w raporcie na liście pracowników wraz z informacją, o której rozpoczął pracę (Rys 4.18). Przedstawiona jest również informacja, że nadal jest w pracy. W celu zakończenia pracy użytkownik wykorzystuje przycisk „Zakończ pracę”, wtedy na raporcie wyświetla się czas zakończenia pracy (Rys 4.19) oraz podliczony zostaje łączny czas pracy.



Rysunek 4.18: Widok listy pracowników po rozpoczęciu pracy przez pracownika

Rysunek 4.19: Widok listy pracowników po zakończeniu pracy przez pracownika

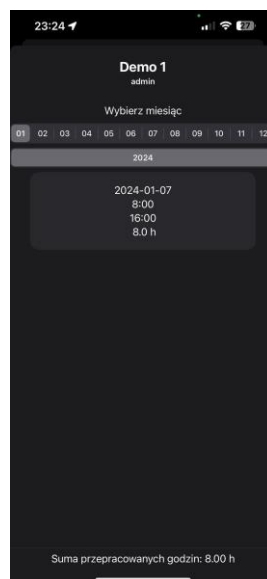
4.4.5 Sprawdzenie sumy przepracowanych godzin

Użytkownik w celu sprawdzenia ilości przepracowanych godzin musi wybrać na dolnym menu zakładkę „Pracownicy”, ukaże się wtedy lista pracowników (Rys 4.20).



Rysunek 4.20: Lista pracowników

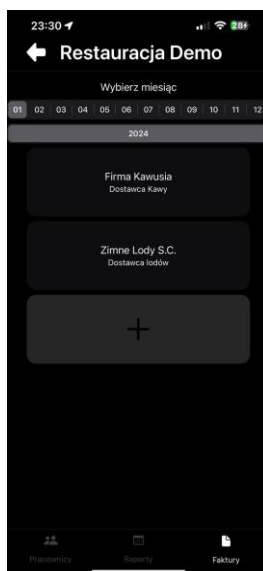
Po kliknięciu w wybranego pracownika wyświetla nam się lista dni, w których był w pracy z informacjami: godzina rozpoczęcia pracy, godzina zakończenia pracy, suma przepracowanych godzin. Na dole ekranu widoczne jest podsumowanie godzin w wybranym miesiącu. Na górze ekranu jest możliwość wyboru miesiąca, który użytkownik może sprawdzić. (Rys 4.21)



Rysunek 4.21: Widok dni pracy wybranego wcześniej pracownika

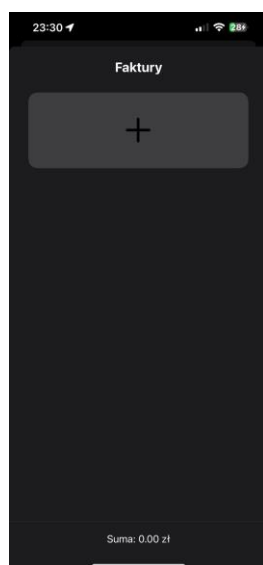
4.4.6 Dodanie nowej faktury do Menadżera Faktur

W celu dodania nowej faktury do Menadżera Faktur użytkownik na dolnym menu wybiera zakładkę „Faktury”, użytkownikowi wyświetla się lista dodanych wcześniej firm (Rys 4.22). Na górze użytkownik ma możliwość wybrania miesiąca, dla którego chce przeglądać dodane wcześniej faktury lub dodać nową. Następnie wybiera firmę, dla której chce dodać nową fakturę.



Rysunek 4.22: Lista firm dodanych do Menadżera Faktur

Na ekranie ukazuje się lista faktur dodanych dla danej firmy w wybranym miesiącu. W celu dodania nowej użytkownik wykorzystuje przycisk „plus” (Rys 4.23).



Rysunek 4.23: Lista faktur danej firmy w wybranym miesiącu

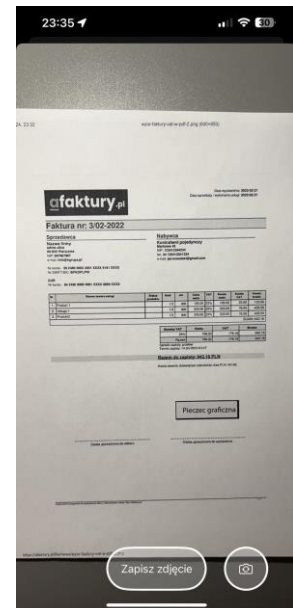
Użytkownik uzupełnia dane w formularzu dodawanej faktury (Rys 4.24 oraz Rys 4.25): numer faktury, data wystawienia faktury, termin płatności faktury, kwota, notatka (opcjonalna), zdjęcie faktury (opcjonalne – Rys. 4.26). Następnie potwierdza przyciskiem „Dodaj”

A screenshot of a mobile application interface for adding a new invoice. The form contains input fields for 'Numer faktury' (Invoice Number), 'Kwota' (Amount), and 'Notatka' (Note). Below these are date pickers for 'Data wystawienia faktury' (Invoice Date) and 'Data płatności' (Due Date), both set to '7 Jan 2024'. A blue link 'Zrób zdjęcie' (Take photo) is visible. At the bottom is a large green button labeled 'Dodaj' (Add).

Rysunek 4.24: Formularz dodania faktury przed uzupełnieniem

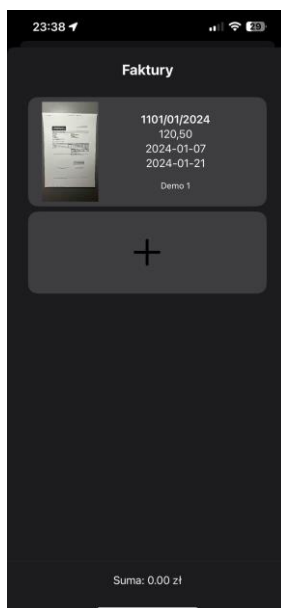
A screenshot of the same invoice form, now filled out. The 'Numer faktury' field contains '1101/01/2024', 'Kwota' contains '120,50', and 'Notatka' is empty. The dates remain '7 Jan 2024'. A small thumbnail image of the invoice is shown above the green 'Dodaj' button. A blue link 'Zrób zdjęcie' is still present.

Rysunek 4.25: Formularz dodania faktury po uzupełnieniu



Rysunek 4.26: Widok aparatu wykorzystywany w celu dodania zdjęcia faktury

Po dodaniu faktura wyświetla się na liście faktur (Rys 4.27) po kliknięciu w wybraną fakturę powiększa się zrobione wcześniej zdjęcie faktury (Rys 4.28).



Rysunek 4.27 – Lista faktur po dodaniu nowej faktury



Rysunek 4.28 – Powiększone zdjęcia dodanej wcześniej faktury

Rozdział 5

Specyfikacja wewnętrzna

Rozdział przedstawia analizę architektury systemu oraz opis struktur danych.

5.1 Przedstawienie Idei

Główną ideą podczas powstawania aplikacji był przejrzysty, intuicyjny interfejs, który będzie łatwy w obsłudze dla każdego potencjalnego użytkownika. Drugim ważnym aspektem była szybka i niezawodna baza danych.

5.2 Architektura systemu

5.2.1 Aplikacja mobilna

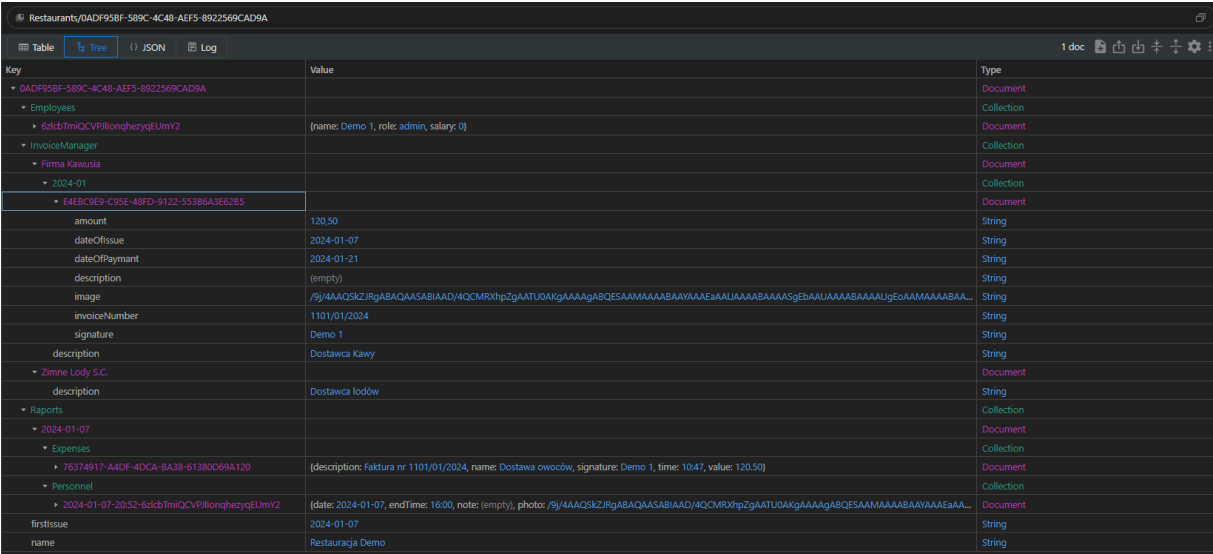
Głównym elementem przedstawionego projektu jest aplikacja mobilna napisana w języku Swift z wykorzystaniem SwiftUI. Język Swift jest językiem programistycznym dedykowanym do rozwiązań przeznaczonych na urządzenia firmy Apple Inc. W projekcie wykorzystany został Swift 5.9, co umożliwia korzystanie z najnowszych funkcji oraz bibliotek jednocześnie zapewniając bezpieczeństwo na najwyższym możliwym poziomie. Integracja Swift z frameworkiem SwiftUI pozwala tworzenie rozbudowanej aplikacji i płynną interakcję z użytkownikiem.

5.2.2 Baza danych

Aplikacja korzysta z Firebase Cloud Firestore Database, które jest niezawodnym i łatwo skalowalnym rozwiązaniem do przechowywania danych. Zaletą rozwiązania jest szybkość i prostota implementacji w naszym projekcie. Firebase Firestore jest oparte na modelu NoSQL, dane przechowywane są w formie kolekcji oraz dokumentów, co pozwala na efektywne zarządzanie nimi i możliwość łatwego dostosowania do ewentualnych zmian w strukturze bazy danych.

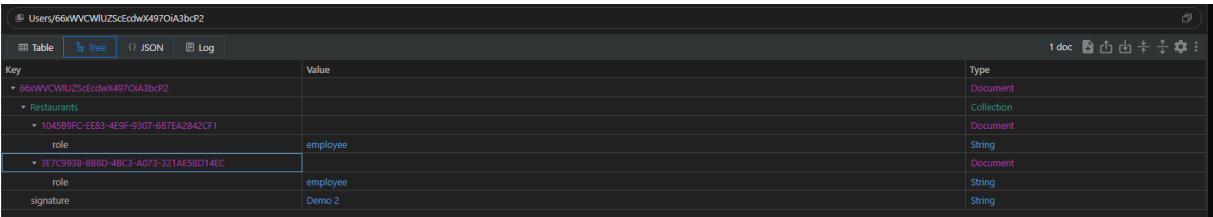
5.3 Opis struktury danych

Dane w przedstawianym projekcie przechowywane są w bazie danych Firebase Cloud Firestore Database. Struktura bazy danych oparta jest na NoSQL, dane przechowywane są w formie kolekcji oraz dokumentów, co przypomina format JSON. Baza danych posiada dwie główne gałęzie „Restaurants” (Rys 5.1) oraz „Users” (Rys 5.2).



Key	Value	Type
0ADF95BF-589C-4C48-AEF5-8922569CAD9A		Document
Employees		Collection
62cbTmiQCVpIlonqhezqEumY2	{name: Demo 1, role: admin, salary: 0}	Document
InvoiceManager		Collection
Firma Kawasta		Document
2024-01		Collection
E4EBC9E9-C95E-48FD-9122-55386A362B5		Document
amount	120.50	String
dateOfIssue	2024-01-07	String
dateOfPayment	2024-01-21	String
description	{empty}	String
image	/9j/4AAQSkZ/RgABAQAASABIAAD/4QCMR3hpZgAATU0AKgAAAAGABQESAAAAAABAAVAAAEaAAUAAAAAABAAASgEbAAUAAAAAABAAAUgEoAAMAAAAABAA...	String
invoiceNumber	1101/01/2024	String
signature	Demo 1	String
description	Dostawca Kawy	String
Zimne Lody S.C.		Document
description	Dostawca lodów	String
Reports		Collection
2024-01-07		Document
Expenses		Collection
76374917-AADF-4DCA-BA38-61380D69A120	{description: Faktura nr 1101/01/2024, name: Dostawa owocow, signature: Demo 1, time: 10:47, value: 120.50}	Document
Personnel		Collection
2024-01-07-2052-62cbTmiQCVpIlonqhezqEumY2	{date: 2024-01-07, endTime: 16:00, note: {empty}, photo: /9j/4AAQSkZ/RgABAQAASABIAAD/4QCMR3hpZgAATU0AKgAAAAGABQESAAAAAABAAVAAAEaAAUAAAAAABAAASgEbAAUAAAAAABAAAUgEoAAMAAAAABAA...	Document
firstIssue	2024-01-07	String
name	Restauracja Demo	String

Rysunek 5. 1: Kolekcja „Restaurants” na przykładzie restauracji „Restauracja Demo”



Key	Value	Type
66xWVCWUJZScEcdwX4970iA3bcP2		Document
Restaurants		Collection
1045B9FC-EE83-4E9F-9307-6B7EA2842CF1		Document
role	employee	String
3E7C9938-8B8D-48C3-A073-321AE5BD14EC		Document
role	employee	String
signature	Demo 2	String

Rysunek 5.2: Kolekcja „Users” na przykładzie użytkownika „Demo 1”

Rozdział 6

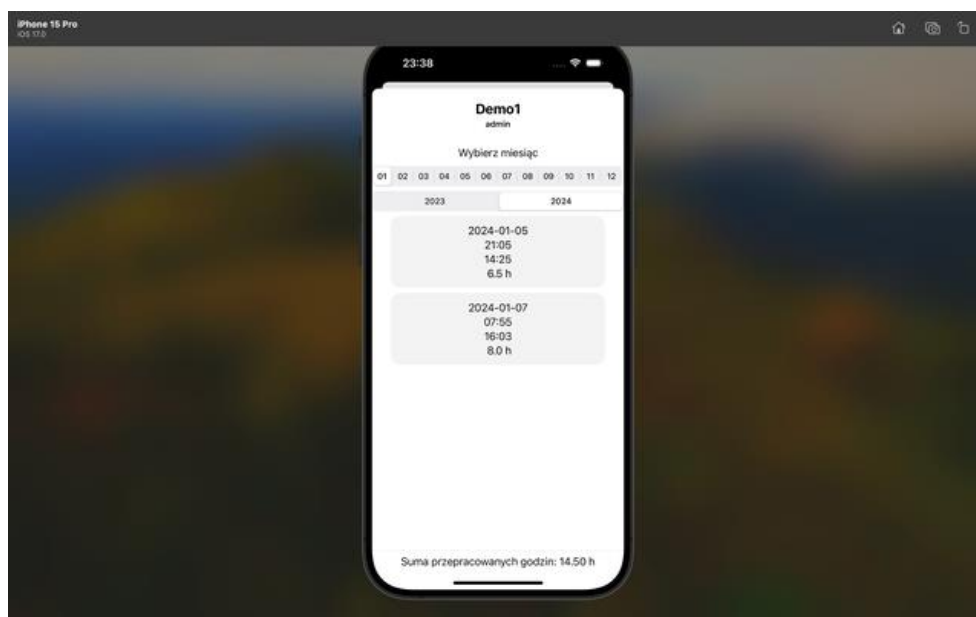
Weryfikacja i walidacja

Rozdział przedstawia sposób testowania aplikacji wraz z informacją o sprzęcie na jakim była testowana.

6.1 Testowanie

Testy aplikacji zostały przeprowadzone dwuetapowo, pierwszym etapem było testowanie każdej funkcjonalności bezpośrednio po jej wdrożeniu do aplikacji za pomocą symulatora dostępnego w środowisku XCode (Rysunek 6.1), do tego celu wykorzystywane były symulatory kilku telefonów z różnymi wersjami oprogramowania systemowego:

- iPhone 13 Pro Max, iOS 17.1.1
- iPhone 8, iOS 16.7.2
- iPhone 15 Pro, iOS 17.0.1



Rysunek 6.1: Aplikacja uruchomiona w symulatorze – iPhone 15 Pro, iOS 17.0.1

Kolejnym krokiem po ukończeniu pierwszego etapu testów było wdrożenie aplikacji do pierwszej restauracji. Aplikacja została udostępniona z wykorzystaniem Apple TestFlight do restauracji posiadającej 12 pracowników w wieku od 18 do 39 lat. Podczas trwania drugiego etapu testów aplikacja została zainstalowana na 6 różnych urządzeniach z różnymi wersjami oprogramowania systemowego (Rys 6.2):

- iPhone 13 Pro Max, iOS 17.1.1
- iPhone 13, iOS 17.2.1
- iPhone 11 Pro, iOS 17.1.2
- iPhone 14, iOS 17.2.1
- iPhone 14 Pro Max, iOS 17.2.1
- iPhone 8 Plus, iOS 16.7.2

Testers (6)

Add Filter

<input type="checkbox"/>	TESTER	STATUS ^	SESSIONS	CRASHES	FEEDBACK	DEVICES
<input type="checkbox"/>	Public link Anonymous	Installed 1.0 (7) Dec 3, 2023				iPhone 13 Pro Max iOS 17.1.1
<input type="checkbox"/>	Public link Anonymous	Installed 1.0 (10) Jan 5, 2024		5		iPhone 13 iOS 17.2.1
<input type="checkbox"/>	Public link Anonymous	Installed 1.0 (10) Jan 5, 2024		1		iPhone 11 Pro iOS 17.1.2
<input type="checkbox"/>	Public link Anonymous	Installed 1.0 (10) Jan 5, 2024				iPhone 14 iOS 17.2.1
<input type="checkbox"/>	Public link Anonymous	Installed 1.0 (10) Jan 5, 2024				iPhone 14 Pro Max iOS 17.2.1
<input type="checkbox"/>	Public link Anonymous	Installed 1.0 (10) Jan 6, 2024				iPhone 8 Plus iOS 16.7.2

Rysunek 6.2: Urządzenia wykorzystane do drugiego etapu testów

Łącznie zostało udostępnione 11 wersji aplikacji do testów dla użytkowników (Rys 6.3).

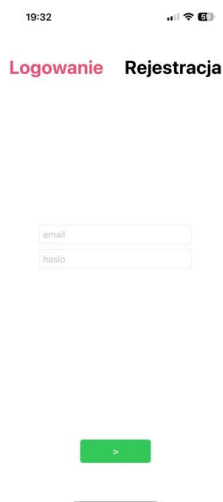
Version 1.0

BUILD ^	STATUS	GROUPS	INVITES	INSTALLS	SESSIONS	CRASHES	FEEDBACK
11	Ready to Submit Expires in 88 days		-	-	-	-	-
10	Testing Expires in 87 days		6	3	63	-	-
9	Testing Expires in 70 days		6	5	123	-	-
8	Testing Expires in 58 days		6	6	-	5	-
7	Testing Expires in 54 days		6	6	66	-	-
6	Testing Expires in 39 days		6	4	-	1	-
5	Approved Expires in 37 days		-	2	-	-	-
4	Approved Expires in 33 days		-	3	-	-	-
3	Testing Expires in 27 days		6	2	-	-	-
2	Testing Expires in 18 days		6	2	-	-	-
1	Expired		6	3	-	-	-

Rysunek 6.3: Wersje udostępnione podczas drugiego etapu testów

6.2 Walidacja

Podczas logowania oraz rejestracji następuje walidacja wprowadzonych danych, która odbywa się po stronie Google Firebase Authentication [1]. Użytkownik wypełnia formularz logowania lub rejestracji (Rys 6.4), po wciśnięciu przycisku „Dalej” system komunikuje się z Firebase Authentication (Rys 6.5), które następnie zwraca nam dane zalogowanego użytkownika bądź w formie kodu błędu informacje o napotkanym problemie podczas próby logowania.



Rysunek 6.4: Formularz logowania w aplikacji

```
@MainActor
func login() async -> Bool{

    do{
        let authDataResult = try await Auth.auth().signIn(withEmail: self.email, password: self.password)
        data = authDataResult.user
        self.isLogged = true

        if await self.fetchData(){
            return true
        }
        else{
            return false
        }
    }
    catch{
        print("There was an issue when trying to sign in: \(error)")
        self.errorMessage = error.localizedDescription
        return false
    }
}
```

Rysunek 6.5: Kod funkcji odpowiadającej logowanie się użytkownika z pomocą Firebase Authentication

Najważniejsze kody błędów:

Code	Meaning
<code>FIRAuthErrorCodeOperationNotAllowed</code>	Indicates that email and password accounts are not enabled. Enable them in the Auth section of the Firebase console .
<code>FIRAuthErrorCodeInvalidEmail</code>	Indicates the email address is malformed.
<code>FIRAuthErrorCodeUserDisabled</code>	Indicates the user's account is disabled.
<code>FIRAuthErrorCodeWrongPassword</code>	Indicates the user attempted sign in with a wrong password.
<code>FIRAuthErrorCodeNetworkError</code>	Indicates a network error occurred during the operation.
<code>FIRAuthErrorCodeUserNotFound</code>	Indicates the user account was not found. This could happen if the user account has been deleted.
<code>FIRAuthErrorCodeUserTokenExpired</code>	Indicates the current user's token has expired, for example, the user may have changed account password on another device. You must prompt the user to sign in again on this device.
<code>FIRAuthErrorCodeTooManyRequests</code>	Indicates that the request has been blocked after an abnormal number of requests have been made from the caller device to the Firebase Authentication servers. Retry again after some time.
<code>FIRAuthErrorCodeInvalidAPIKey</code>	Indicates the application has been configured with an invalid API key.
<code>FIRAuthErrorCodeAppNotAuthorized</code>	Indicates the App is not authorized to use Firebase Authentication with the provided API Key. go to the Google API Console and check under the credentials tab that the API key you are using has your application's bundle ID whitelisted.
<code>FIRAuthErrorCodeKeychainError</code>	Indicates an error occurred when accessing the keychain. The <code>NSLocalizedFailureReasonErrorKey</code> and <code>NSUnderlyingErrorKey</code> fields in the <code>NSError.userInfo</code> dictionary will contain more information about the error encountered.
<code>FIRAuthErrorCodeInternalError</code>	Indicates an internal error occurred. Please report the error with the entire <code>NSError</code> object.

6.3 Wykryte i usunięte błędy

Najważniejszym błędem, wykrytym podczas całego procesu powstawania aplikacji był błąd funkcji odpowiedzialnej ze obliczanie czasu pracy. System przewiduje możliwość rozpoczęcia oraz zakończenia pracy co 15 minut (Rys 6.7) np. 12:00, 12:15, 12:30, 12:45, 13:00.... Użytkownik powinien w aplikacji rozpocząć swój dzień pracy najpóźniej równo o godzinie rozpoczęcia pracy, w przeciwnym razie system będzie liczył przepracowany czas od

następnego dostępnego okna rozpoczęcia pracy. Pracownik może zaznaczyć zakończenie pracy nie wcześniej niż o godzinie najbliższego okna rozliczenia czasu pracy.

```
func calcWorkTime() -> Int{
    if self.endTime != ""{
        var startHour = Int(self.startTime.dropLast(3))!
        var startMin = String(self.startTime.dropFirst(3))

        switch Int(startMin)! {
            case 0:
                startMin = "00"
            case 1..<16:
                startMin = "15"
            case 16..<31:
                startMin = "30"
            case 31..<46:
                startMin = "45"
            case 46..<60:
                startHour += 1
                if startHour == 24 {
                    startHour = 0
                }
                startMin = "00"
            default:
                print("Error")
        }

        let newStartTime = String(format: "%02d", startHour) + ":" + startMin
        var endHour = String(endTime.dropLast(3))
        var endMin = String(endTime.dropFirst(3))

        switch Int(endMin)! {
            case 0..<14:
                endMin = "00"
            case 14..<29:
                endMin = "15"
            case 29..<44:
                endMin = "30"
            case 44..<60:
                endMin = "45"
            default:
                print("Error")
        }

        let newEndTime = endHour + ":" + endMin
    }
}
```

Rysunek 6.7: Fragment funkcji odpowiedzialnej za liczenie czasu pracy

Podczas powstawania funkcji powstał błąd, podczas rozpoczęcia pracy o godzinie np. 7:55 funkcja błędnie zaokrąglala godzinę do 7:00 zamiast 8:00, co przełożyło się na błędne

wyliczenia czasu pracy użytkowników. Problem udało się rozwiązać dokonując korekty w kodzie funkcji (Rys 6.8).

```
case 46..<60:  
  startHour += 1  
  if startHour == 24 {  
    startHour = 0  
  }  
  startMin = "00"
```

Rys 6.8: Fragment kodu, w którym występował błąd

Rozdział 7

Podsumowanie i wnioski

Celem pracy było zaprojektowanie oraz implementacja aplikacji mobilnej wspierającej zarządzanie lokalem gastronomicznym. Motywacją było uproszczenie zarządzania restauracją jednocześnie przekładając się na oszczędność czasu oraz zwiększenie wydajności pracy firmy.

Cel udało się zrealizować z założonymi w wymaganiach funkcjonalnościami. Gotowa aplikacja sprawdza się w zarządzaniu czasem pracy pracowników. Poza obliczaniem czasu pracy na podstawie czasu rozpoczęcia oraz zakończenia pracy, przekłada się na zmniejszenie ilości spóźnień pracowników, dzięki kontroli czasu rozpoczęcia pracy. Kompletowanie dziennych raportów w aplikacji pozwala właścicielowi lub menadżerowi firmy na szybką kontrolę dziennych wydatków oraz utargów. Zawarty w aplikacji menadżer faktur poza obliczaniem miesięcznych należności dla konkretnej firmy pozwala na zdalny wgląd w faktury.

7.1 Możliwe kierunki rozwoju aplikacji

Aplikacja pozostawia duże pole do dalszego rozwoju, poniżej wymienione zostały najważniejsze z potencjalnych przyszłych ścieżek rozwoju:

- Stworzenie odpowiednika programu w postaci aplikacji mobilnej przeznaczonej na system Android
- Dodanie nowych funkcjonalności:
 - Generowanie miesięcznych podsumowań, wyliczanie miesięcznego zysku
 - Generowanie miesięcznych raportów w formacie do druku
 - Menadżer urlopów pracowników
 - Menadżer umów zawartych z pracownikami
 - Moduł do zarządzania stanem magazynowym
 - Możliwość zarządzania rezerwacjami stolików
 - Zarządzanie grafikami pracy

- System powiadomień o ważnych wydarzeniach – spóźnienie pracownika, zmiana w grafiku pracy itp.
- Analiza trendów sprzedaży

Bibliografia

- [1] Google Firebase Authentication – Dokumentacja z 2023-12-05 <https://firebase.google.com/docs/auth/ios/errors> (dostęp: 10.12.2023r)
- [2] Google Firebase Cloud Firestore – Dokumentacja z 2023-12-05 <https://firebase.google.com/docs/firestore> (dostęp: 08.12.2023r)
- [3] Apple Swift – Dokumentacja <https://developer.apple.com/documentation/swift> (dostęp: 20.10.2023r)
- [4] Apple SwiftUI – Dokumentacja <https://developer.apple.com/documentation/swiftui/> (dostęp: 21.10.2023r)
- [5] Apple Human Interface Guidelines <https://developer.apple.com/design/human-interface-guidelines> (dostęp 25.10.2023r)

Dodatki

Spis skrótów i symboli

HoReCa od słów Hotel, Restaurant, Cafe, czyli branża obejmująca hotele, restauracje i kawiarnie, działająca w sektorze usług gastronomicznych i noclegowych.

iOS iPhone Operating System

UI Interfejs Użytkownika

Źródła

Jeżeli w pracy konieczne jest umieszczenie długich fragmentów kodu źródłowego, należy je przenieść do tego miejsca.

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie, do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- dane testowe
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

2.1	Grafika przedstawiająca rodzinę produktów GASTRO	10
2.2	Grafika przedstawia menu główne programu GASTRO POS	10
2.3	Grafika przedstawia wygląd programu GASTRO SZEŁ	11
2.4	Grafika przedstawia wygląd strony internetowej mojeGASTRO	11
3.1	Diagram przypadków użycia	16
4.1	Ekran logowania widoczny po uruchomieniu aplikacji	20
4.2	Ekran logowania po uzupełnieniu danych logowania	20
4.3	Ekran listy restauracji widoczny po pomyślnym zalogowaniu	20
4.4	Widok listy restauracji	21
4.5	Okienko dodawania restauracji – wymagane uzupełnienie nazwy	21
4.6	Okienko dodawania restauracji – po uzupełnieniu nazwy	21
4.7	Widok listy restauracji po dodaniu nowej restauracji	22
4.8	Główny widok raportu, widoczne kategorie	22
4.9	Lista wydatków	23
4.10	Formularz dodania nowego wydatku przed uzupełnieniem	23
4.11	Formularz dodania nowego wydatku po uzupełnieniu	24
4.12	Lista wydatków – widoczny nowo dodany wydatek	24
4.13	Główny widok raportu po wprowadzeniu nowego wydatku	24
4.15	Główny widok raportu	25
4.16	Widok listy pracowników	25
4.17	Widok kamery telefonu	26
4.18	Widok listy pracowników po rozpoczęciu pracy przez pracownika	26
4.19	Widok listy pracowników po zakończeniu pracy przez pracownika	26

4.20	Lista pracowników	27
4.21	Widok dni pracy wybranego wcześniej pracownika	27
4.22	Lista firm dodanych do Menadżera Faktur	28
4.23	Lista faktur danej firmy w wybranym miesiącu	28
4.24	Formularz dodania faktury przed uzupełnieniem	29
4.25	Formularz dodania faktury po uzupełnieniu	29
4.26	Widok aparatu wykorzystywany w celu dodania zdjęcia faktury	29
4.27	Lista faktur po dodaniu nowej faktury	29
4.28	Powiększone zdjęcia dodanej wcześniej faktury	29
5.1	Kolekcja „Restaurants” na przykładzie restauracji „Restauracja Demo”	31
5.2	Kolekcja „Users” na przykładzie użytkownika „Demo 1”	31
6.1	Aplikacja uruchomiona w symulatorze – iPhone 15 Pro, iOS 17.0.1	32
6.2	Urządzenia wykorzystane do drugiego etapu testów	33
6.3	Wersje udostępnione podczas drugiego etapu testów	33
6.4	Formularz logowania w aplikacji	34
6.5	Kod funkcji odpowiadającej logowanie się użytkownika	34
6.7	Fragment funkcji odpowiedzialnej za liczenie czasu pracy	36
6.8	Fragment kodu, w którym występował błąd	37