1. Architektura wybrana do realizacji projektu
W przypadku aplikacji "Fakturomat" wybrana architektura jest oparta na modelu klient-serwer w oparciu o mikroserwisy oraz wzorzec MVVM (Model-View-ViewModel).

Dlaczego wybrano tę architekturę?
Mikroserwisy: Mikroserwisy pozwalają na podział systemu na mniejsze, niezależne moduły, co umożliwia skalowanie, łatwiejsze zarządzanie, a także zapewnia łatwiejsze wdrożenie i utrzymanie aplikacji w długim okresie. W tej architekturze aplikacja jest podzielona na usługi, z których każda odpowiada za inną część funkcjonalności (np. fakturowanie, zarządzanie firmami, autentykacja użytkowników), co umożliwia łatwiejsze wprowadzanie nowych funkcji i modyfikacji.
MVVM: Wzorzec Model-View-ViewModel jest szeroko stosowany w aplikacjach desktopowych i mobilnych, szczególnie przy wykorzystaniu technologii .NET, takiej jak WPF. Dzięki MVVM łatwiej zarządza się interakcją między warstwą prezentacji (widokiem), a logiką biznesową (modelem). Umożliwia to rozdzielenie logiki interfejsu użytkownika od samej logiki biznesowej, co zwiększa testowalność i skalowalność aplikacji.
Supabase jako Backend: W tej aplikacji do zarządzania danymi i autentykacją użytkowników wykorzystano Supabase, które jest open-source'owym odpowiednikiem Firebase, bazującym na PostgreSQL. To rozwiązanie pozwala na łatwą integrację z aplikacjami i umożliwia szybkie budowanie backendu bez konieczności zarządzania własną infrastrukturą serwerową.
2. Główne atrybuty jakościowe
Atrybuty jakościowe aplikacji "Fakturomat" to:

Skalowalność: Mikroserwisowa architektura pozwala na skalowanie poszczególnych komponentów w zależności od zapotrzebowania. Na przykład, jeśli w systemie pojawi się duża liczba faktur, można skalować odpowiednią usługę bez wpływu na inne.

Wydajność: Dzięki użyciu Supabase oraz prostych operacji bazodanowych, aplikacja może obsługiwać duże ilości danych. Procesy, takie jak generowanie faktur, są zoptymalizowane, aby zminimalizować czas odpowiedzi.

Bezpieczeństwo: W aplikacji zastosowano mechanizmy autentykacji użytkowników przez Supabase, który zarządza bezpieczeństwem przechowywania haseł i sesji. Komunikacja z backendem odbywa się przy użyciu HTTPS, co zapewnia szyfrowanie danych.

Użyteczność: Interfejs użytkownika jest prosty, czytelny i intuicyjny, co sprawia, że korzystanie z aplikacji jest łatwe nawet dla osób bez doświadczenia technicznego. Implementacja wzorca MVVM pomaga utrzymać czystość logiki i kodu, co wpływa na szybkość reakcji aplikacji.

Modularność: System jest zaprojektowany w sposób umożliwiający dodawanie nowych funkcji bez konieczności modyfikacji istniejących części aplikacji. Dzięki zastosowaniu mikroserwisów oraz wzorca MVVM, nową funkcjonalność można łatwo wprowadzić poprzez rozszerzanie istniejących modułów.

Testowalność: Zastosowanie wzorca MVVM oraz modularnej architektury pozwala na łatwe pisanie testów jednostkowych, które są kluczowe do zapewnienia jakości aplikacji.

3. Mechanizmy napędzające aplikację
Supabase: Supabase pełni rolę backendu aplikacji. Umożliwia autentykację użytkowników, przechowywanie danych w bazie PostgreSQL oraz obsługę plików (np. faktur w formacie PDF). Supabase zapewnia również mechanizmy bezpieczeństwa, takie jak role i uprawnienia użytkowników.

Wzorzec MVVM: Mechanizm MVVM (Model-View-ViewModel) napędza interakcję między warstwą użytkownika (View) a logiką biznesową (Model), zapewniając jednocześnie czystość kodu i łatwość jego testowania.

Komponenty UI w WPF: Interfejs użytkownika oparty jest na technologii WPF, która umożliwia tworzenie zaawansowanych aplikacji desktopowych z bogatym interfejsem. Wykorzystanie WPF umożliwia dynamiczne aktualizowanie widoków w odpowiedzi na zmiany w modelu.

Zarządzanie sesjami i stanem: Aplikacja przechowuje informacje o sesji użytkownika, co umożliwia śledzenie stanu aplikacji i personalizację doświadczeń użytkowników.

4. Cykl biznesowy
Cykl biznesowy w aplikacji "Fakturomat" obejmuje kilka etapów:

Rejestracja i logowanie użytkownika: Użytkownik tworzy konto lub loguje się do systemu. Podczas logowania wykorzystywana jest autentykacja użytkownika przez Supabase.

Zarządzanie firmami: Użytkownik może dodać swoje firmy, edytować je lub usuwać. W każdej chwili może przypisać je do swojej sesji użytkownika.

Tworzenie faktur: Po zarejestrowaniu firm, użytkownik może tworzyć faktury. Dane są przechowywane w bazie danych i generowane w formacie PDF. Użytkownik może pobrać fakturę w dowolnym momencie.

Pobieranie faktur: Użytkownik ma dostęp do wszystkich swoich faktur, które zostały wcześniej utworzone. Można je pobrać lub edytować.

Integracja z firmami zewnętrznymi: System może być zintegrowany z firmami zewnętrznymi (np. firmami księgowymi), co umożliwia przesyłanie danych o fakturach lub generowanie raportów.

Zarządzanie użytkownikami (administratorzy): Administratorzy mają dostęp do zarządzania kontami użytkowników oraz konfiguracji aplikacji.

5. Dekompozycja modułowa systemu
Aplikacja "Fakturomat" składa się z kilku głównych modułów:

Moduł autentykacji: Zarządza logowaniem i rejestracją użytkowników, przechowywaniem haseł i sesji.

Moduł zarządzania firmami: Umożliwia użytkownikom dodawanie, edytowanie i usuwanie firm. Zawiera logikę biznesową do zarządzania danymi firm.

Moduł fakturowania: Odpowiada za generowanie faktur, ich zapis w bazie danych oraz tworzenie plików PDF.

Moduł przechowywania dokumentów: Odpowiada za przechowywanie i pobieranie plików PDF faktur. Wykorzystuje Supabase Storage.

Moduł administracyjny: Obejmuje zarządzanie użytkownikami i uprawnieniami. Administratorzy mogą zarządzać danymi użytkowników, firm i faktur.

Moduł raportowania: Obsługuje generowanie raportów, które administratorzy lub użytkownicy mogą pobierać.

6. Możliwość modyfikacji po wdrożeniu
Elastyczność i rozszerzalność: Aplikacja została zaprojektowana z myślą o łatwej modyfikacji i rozbudowie. Modularna struktura umożliwia dodawanie nowych funkcji i poprawek bez wpływu na resztę systemu.

Aktualizacje: Z uwagi na wykorzystanie Supabase, zmiany w backendzie mogą być wprowadzane bez konieczności przeprowadzania skomplikowanych migracji. Dzięki temu łatwiej jest wdrażać nowe funkcje i poprawki po wdrożeniu aplikacji.

Testowalność: Wzorzec MVVM i modularna struktura umożliwiają łatwe wprowadzanie testów jednostkowych i integracyjnych. Dzięki temu można szybko weryfikować poprawność nowych funkcji wprowadzanych do aplikacji.

Dostępność API: Dzięki udostępnionemu API backendu (np. z Supabase), możliwe jest integracje z innymi systemami, co zwiększa elastyczność i umożliwia rozbudowę aplikacji po jej wdrożeniu.