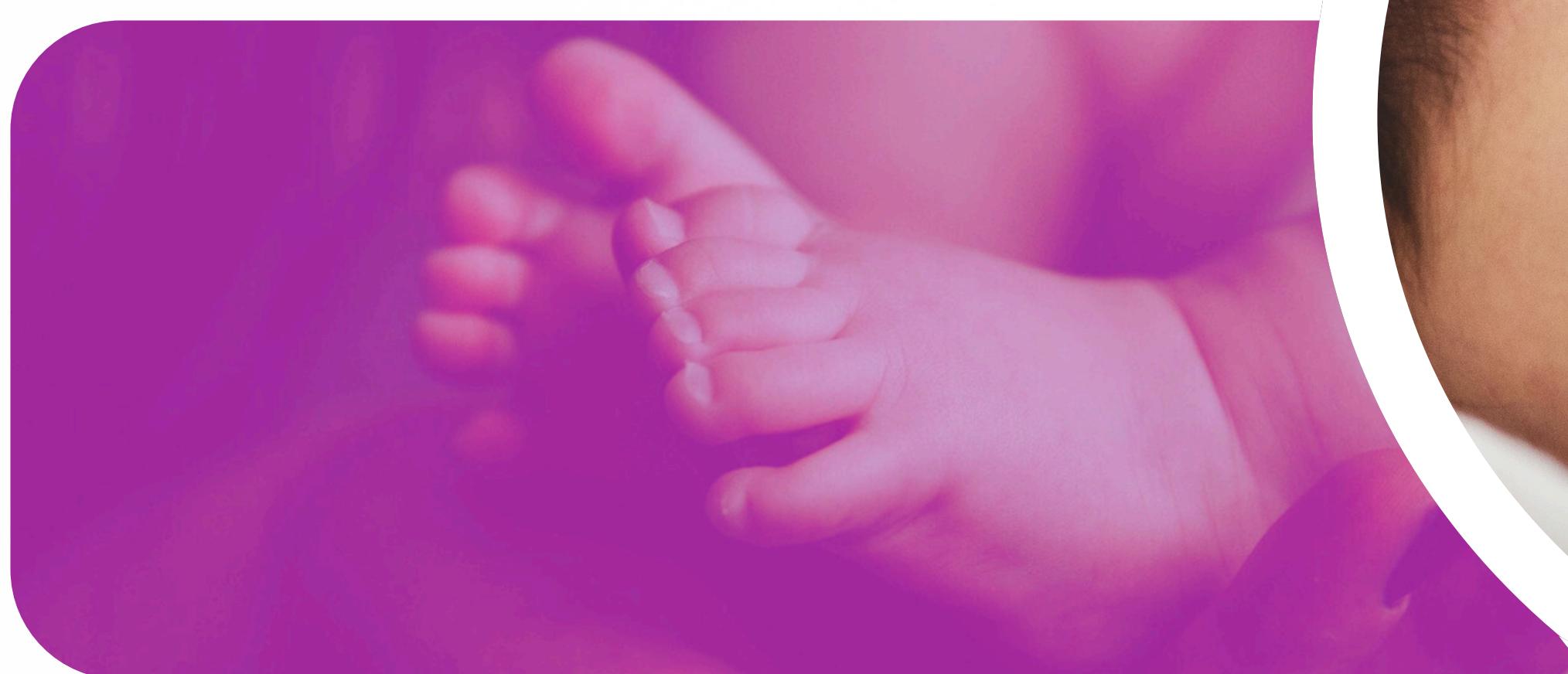


# LUCY COVER



## Podstawowe informacje:

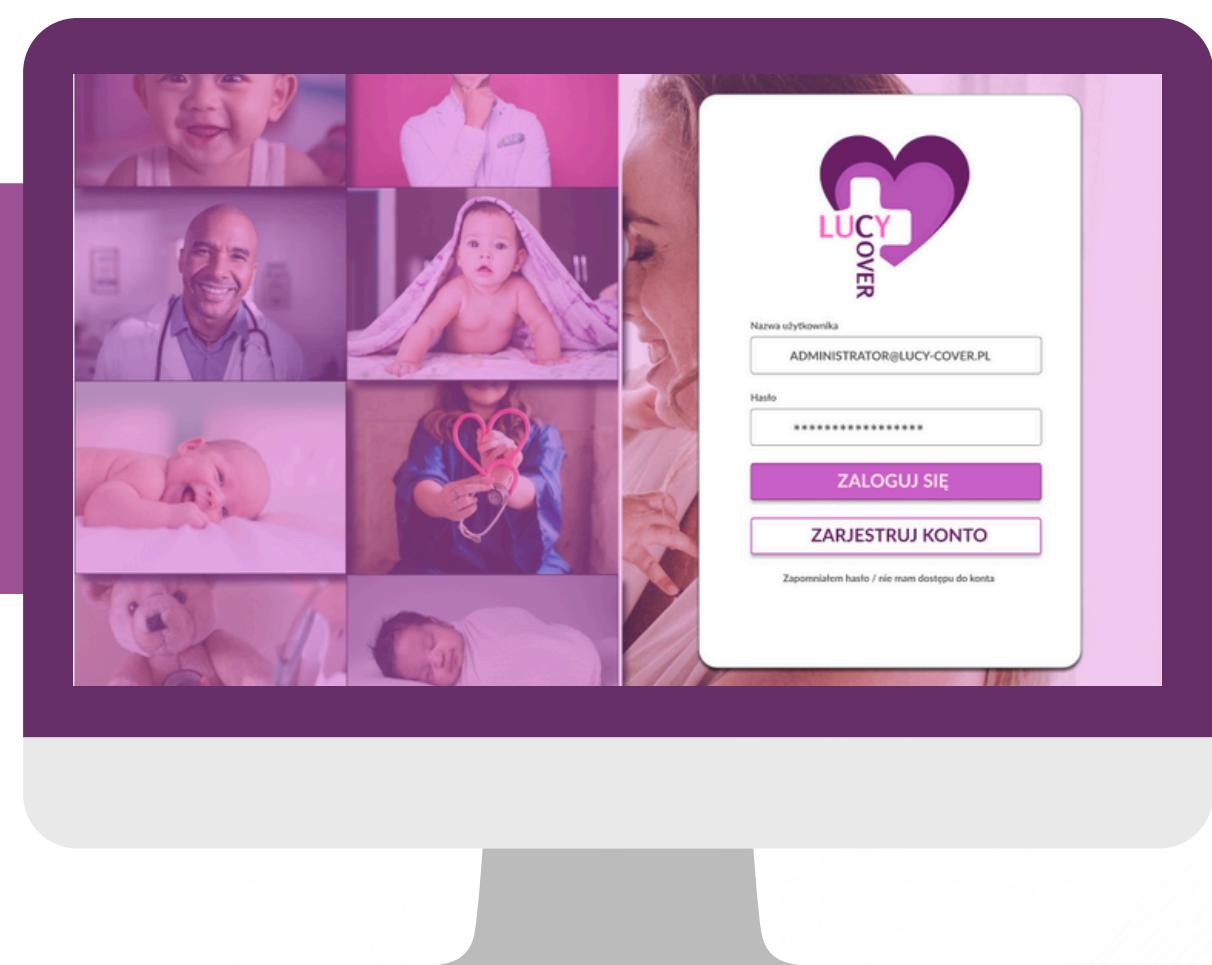
- 1** Projekt powstał na zlecenie lokalnych przychodni położniczych
- 2** Głównym celem aplikacji jest usprawnienie procesu zarządzania oraz bezpiecznego przechowywania danych
- 3** Narzędzie pracy wspierające pracowników przychodni na wszystkich etapach pracy

Jakub Czarnecki

# Założenia aplikacji:

1

**Odseparowany dostęp dla wszystkich pracowników przychodni poprzez wprowadzenie kont użytkowników**



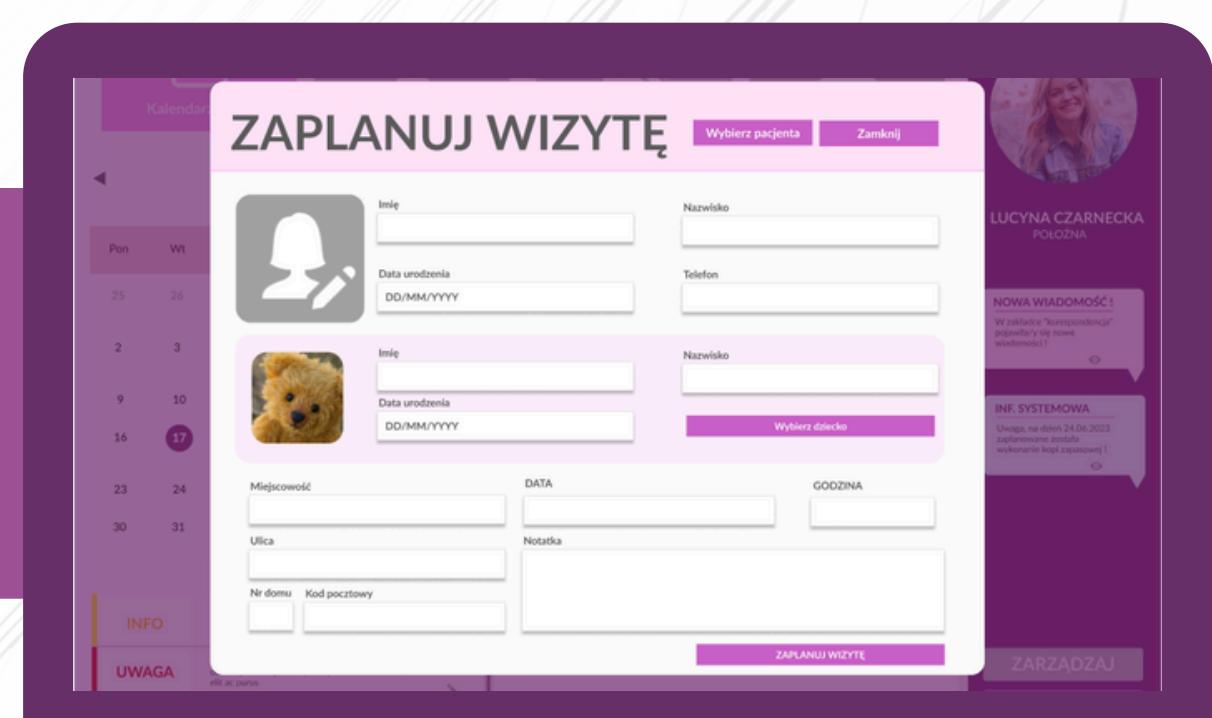
2

**AES**

**Aplikacja przechowuje wrażliwe dane pacjentów w sposób bezpieczny spełniając aktualne standardy cyberbezpieczeństwa**

3

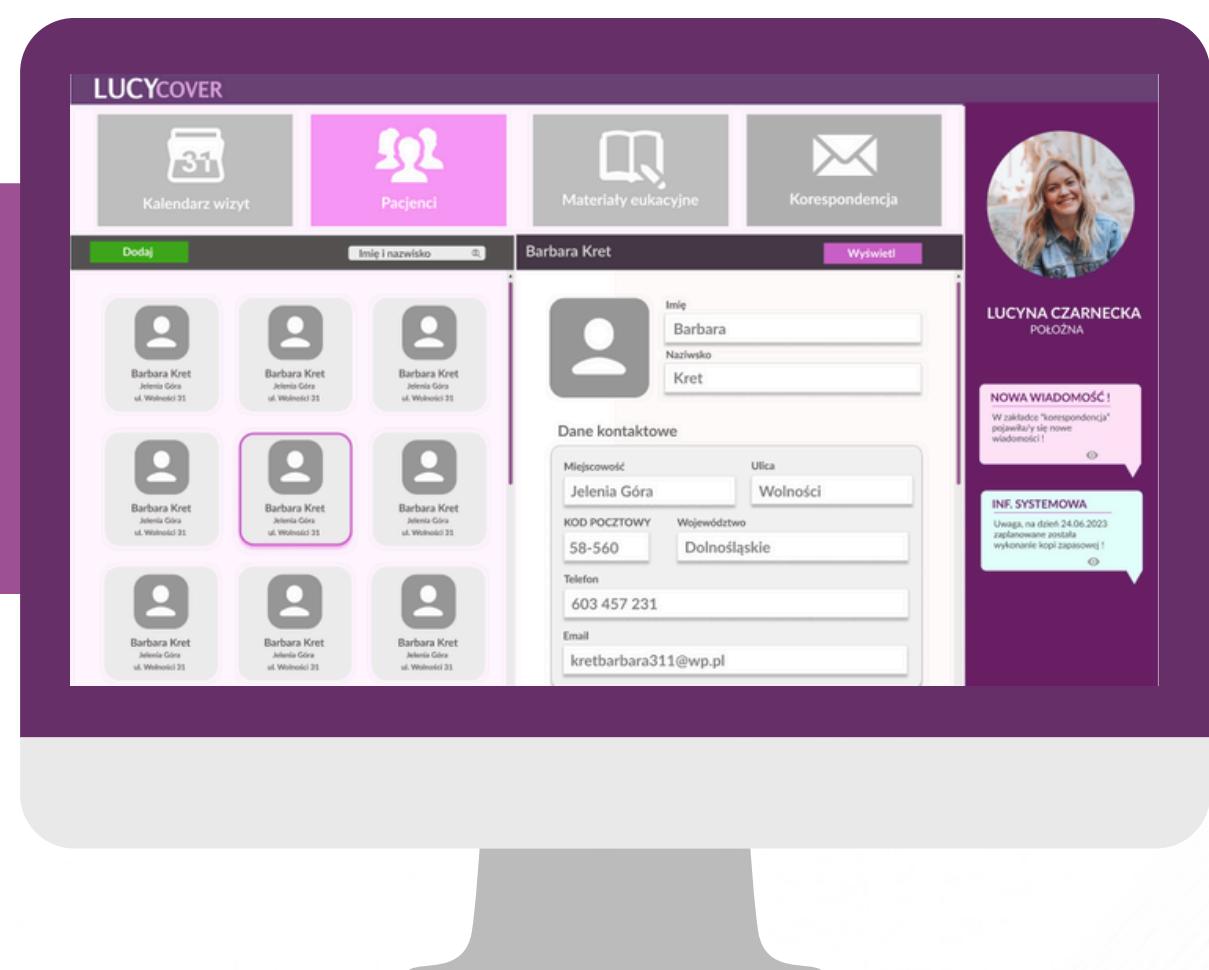
**Funkcja wygodnego planowania wizyt dla użytkowników systemu z opcją automatycznych powiadomień pacjenta za pośrednictwem E-Mail**



# Założenia aplikacji:

4

Aplikacja jest centralną bazą wszystkich pacjentów przypisanych do konkretnych położnych

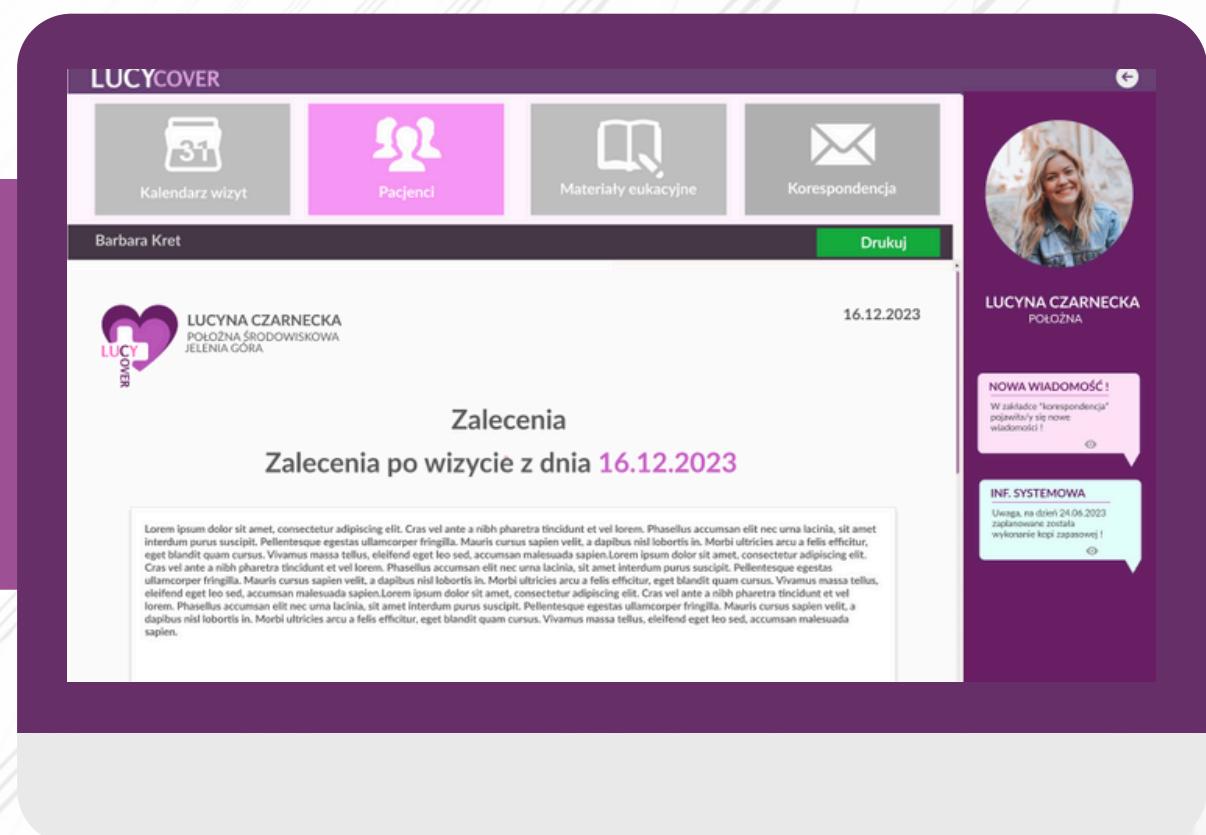


5

Przeniesienie papierowych wersji dokumentacji powizytowej na formę elektroniczną w celach usprawnienia procesów tworzenia i zwiększeniu bezpieczeństwa

6

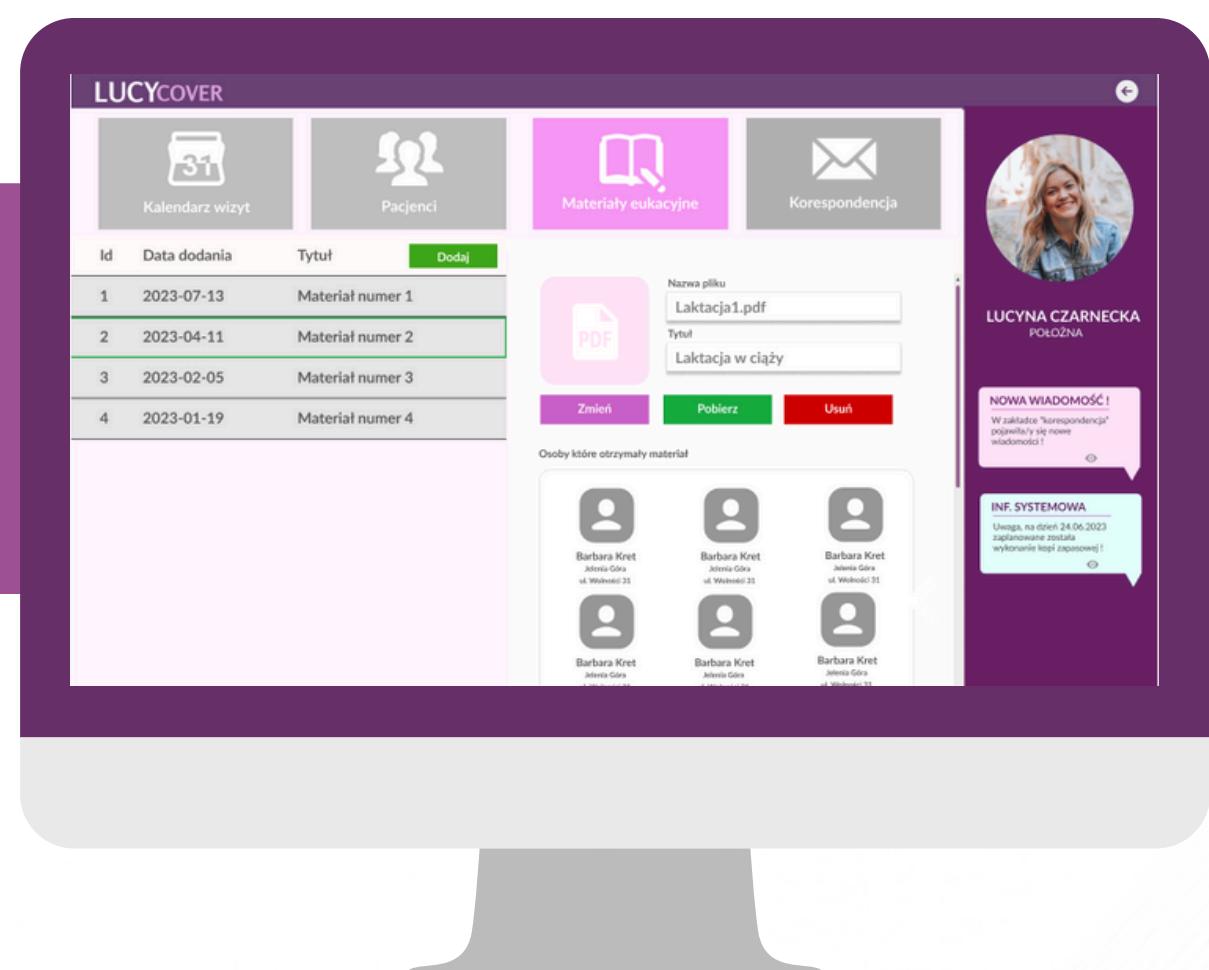
Wprowadzenie opcji wystawiania zaleceń dla pacjenta po wizycie z możliwością zapisania jej w historii oraz z opcją szybkiego wydrukowania



# Założenia aplikacji:

7

Aplikacja umożliwia przechowywanie plików z materiałami edukacyjnymi. Dodatkowo istnieje możliwość udostępniania ich za pomocą poczty elektronicznej



8

Implementacja rozbudowanej walidacji wprowadzanych danych do systemu w celu zapewnienia spójności i integralności informacji przechowywanych w bazie.



9

Z poziomu aplikacji użytkownik ma możliwość prowadzenia konwersacji za pośrednictwem poczty elektronicznej z pacjentem



# Zastosowana architektura

## Podział na warstwy

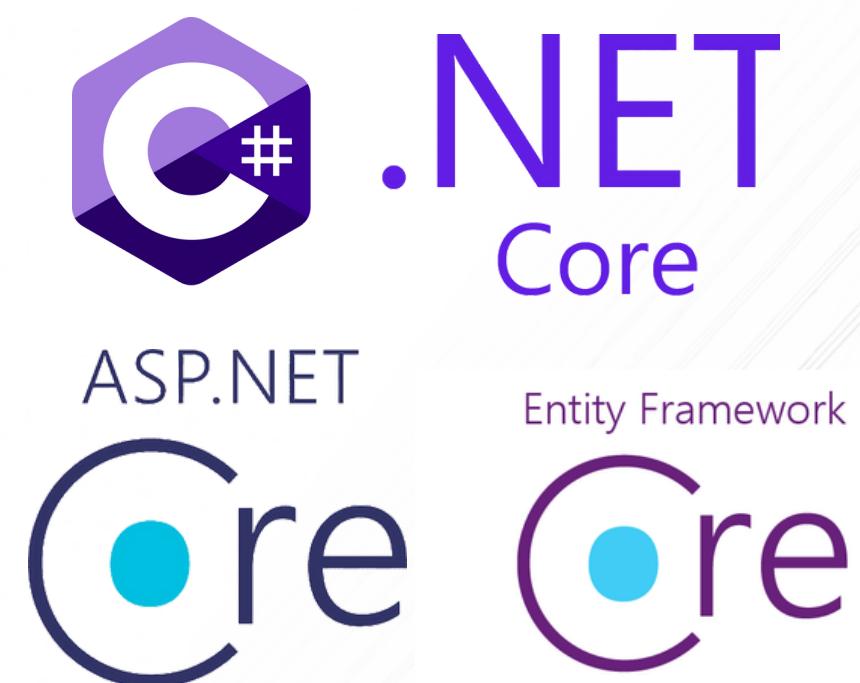
Aplikacja została podzielona na 3 odrębne niezależne warstwy komunikujące się między sobą za pośrednictwem interfejsów API. Dzięki temu podejściu aplikacja jest prosta w zarządzaniu oraz umożliwia rozbudowę w przyszłości.



### Aplikacja kliencka



### Aplikacja serwerowa



### Serwer bazodanowy



# Aplikacja Kliencka

Warstwa ta jest odpowiedzialna za odpowiednie wyświetlenie przetworzonych danych, zapewnienie efektów wizualnych, czy umożliwienie użytkownikowi intuicyjnej i bezproblemowej analizy zwracanych danych.

Odpowiedzialna jest również za pobieranie danych od użytkownika i prawidłowe dostarczenie ich do aplikacji przetwarzającej. W tym miejscu wykonywana jest również wstępna walidacja oraz ochrona przed potencjalnymi atakami, zanim zostanie nawiązane połączenie z serwerem.

Do utworzenia aplikacji klienckiej wykorzystano technologie:

- 1) React 18 - bazowa technologia
- 2) Node.js - do obsługi serwera lokalnego
- 3) Vite.js - do budowania aplikacji i udostępniania jej w wersji lokalnej
- 4) CSS3 - arkusze stylów
- 5) JavaScript - jako bazowy język.

Biblioteki wspomagające:

- React Router - do obsługi routingu na stronie
- React Router Dom - komponenty wspomagające React Router
- Tanstack Query - do obsługi i cachowania komunikacji z backendem
- React Redux - do obsługi stanu globalnego.
- MUI - gotowe komponenty wspomagające
- React Cookie - do obsługi ciasteczek
- Framer Motion - tworzenie animacji

The screenshot shows the Lucy Cover client application interface. At the top, there are two cards: one for 'Materiały eukuracyjne' (with a document icon) and one for 'Korespondencja' (with an envelope icon). Below these are sections for file management and user interaction. On the left, a sidebar shows a list of files: 'numer 3' and 'numer 4'. In the center, there's a file preview for 'Laktacja1.pdf' with options to 'Zmień', 'Pobierz', or 'Usuń'. To the right, a section titled 'Osoby które otrzymały materiał' lists multiple users, all named 'Barbara Kret' from 'Jelenia Góra ul. Wolności 31'. A large circular button in the bottom right corner has a plus sign and the text 'Dodaj osobę'.



# Aplikacja serwerowa

Warstwa ta przede wszystkim odpowiada za przetwarzanie dostarczanych oraz udostępnianych danych, kontroluje przepływ informacji pomiędzy bazą danych, a aplikacją kliencką. Jest odpowiedzialna za autoryzację oraz autentykację użytkowników podczas prób dostępów do interfejsów API. Web serwis waliduje wszystkie otrzymywane informacje zapewniając ich spójność oraz minimalizując wprowadzanie błędnych informacji do systemu.

Aby zapewnić niezawodność działania, solidną jakość kodu, bezpieczeństwo oraz skalowalność web serwisy oparte o technologie ASP.NET Core tworzone są zgodnie z jasno określona strukturą pozwalającą zachować pewny poziom modułowości oraz abstrakcji poszczególnych funkcjonalności aplikacji. Warstwa logiki biznesowej aplikacji LucyCover podzielona została na:

- Kontrolery,
- Zależności,
- Serwisy,
- Wyjątki,
- Middleware.

Do utworzenia aplikacji serwerowej wykorzystano technologie:

- 1) .NET 7 - jako platforma programistyczna
- 2) C# - jako język programowania
- 3) ASP .NET CORE - wykorzystana technologia do stworzenia projektu
- 4) Entity Framework - ORM do komunikacji z serwerem bazodanowym
- 5) XUni - technologia do tworzenia testów aplikacji
- 6) MimeKit - do obsługi korespondencji pocztowej
- 8) Protokoły IMAP oraz SMTP - do obsługi poczty elektronicznej
- 9) JWT BEARER - Podstawowy mechanizm autentykacji
- 10) FluentValidator - Zaawansowane walidowanie przychodzących
- 11) Bogus - generowanie losowych danych
- 11) AutoMapper - Do mapowania obiektów między DTO a encjami z bazy danych

# Serwer bazodanowy

Dane medyczne klasyfikują się do ścisłe strzeżonych i ich utrata bądź wyciek niesie ze sobą szereg poważnych konsekwencji. W związku z tym podczas analizy biznesowej wymogów aplikacji ustanowiono szereg wytycznych i norm jakie powinny zostać zachowane podczas projektowania i zarządzania aplikacją:

- 1) Dostęp do danych traktowany jest jako oddzielna abstrakcja uzyskana poprzez wykorzystanie wzorca projektowego "Repozytoria"
- 2) Zarządzanie komunikacją z bazą danych opiera się o wydajny system ORM EntityFramework
- 3) Wszelkie zapytania do bazy danych są tworzone przy wykorzystaniu zapytań LINQ aby uniknąć zagrożeń takich jak SQL Injection
- 4) Rozbudowa bazy danych opiera się o podejście "Code First"
- 5) Wszelkie dane przed przekazaniem do systemu ORM zostają zaszyfrowane z wykorzystaniem algorytmu AES-16
- 6) Użytkownik końcowy otrzymuje rezultaty ograniczone do wymaganego minimum poprzez mapowanie na przystosowane modele DTO

Do utworzenia warstwy dostępu do danych wykorzystano technologie:

- 1) MSSQL
- 2) Entity Framework
- 3) Biblioteka obsługująca algorytm AES dla Entity FrameWork
- 4) Wzorzec projektowy REPOSITORY

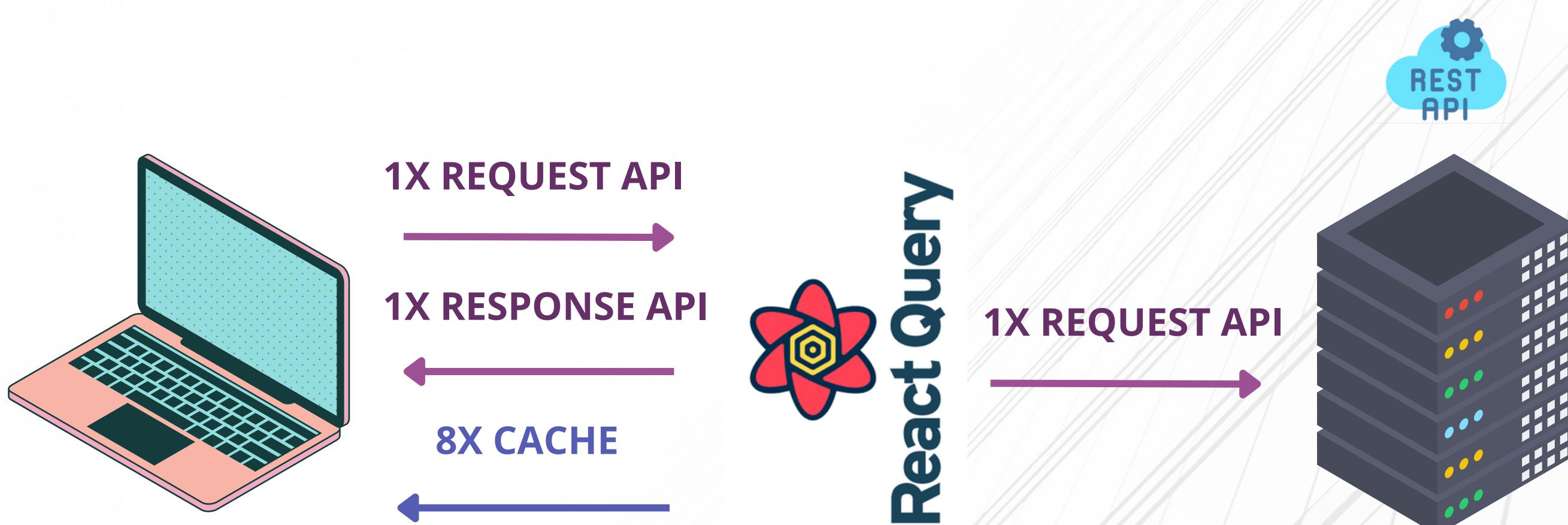
# Komunikacja REST API

Komunikacja między warstwami projektu oparta została o interfejsy API. Aby zachować jak najlepsze standardy komunikacyjne, interfejsy zostały zaprojektowane w stylu architektonicznym REST API. Udało się uzyskać drugi poziom dojrzałości (według Modelu Dojrzałości REST Richarda Fieldinga)

## Podział na czasowniki HTTP      Logiczne URI

```
[HttpGet("{patientId}")]
public ActionResult<RecommendationList.DTO> GetAll([FromRoute] Guid patientId)
{
    RecommendationList.DTO recommendationList = _service.GetAll(patientId);
    return Ok(recommendationList);
}
```

Do obsługi komunikacji wykorzystano sprawdzone istniejące rozwiązanie - Tanstack Query. Pozwala na szybkie i wydajne konstruowanie zapytań do serwera przed załadowaniem strony lub w trakcie jej działania. Dodatkowo TanstackQuery pozwala skonfigurować mechanizm cache, który znacznie zmniejsza ilość wysyłanych zapytań do serwera.



# Plany rozwojowe

Mimo ilości wprowadzonych funkcjonalności oraz poświęconego czasu na samodzielne rozwijanie projektu, w dalszym ciągu istnieje szereg nowych pomysłów i usprawnień, które powinny zostać wdrożone w najbliższym czasie jako kolejna iteracja projektu.

**W tym momencie projekt został przekazany do testowania użytkownikom w celu oceny aplikacji, oraz wykrycia nieprawidłowości.**

**Pomysły na kolejne usprawnienia:**

- Wiele mechanizmów, można uprościć wykorzystując wzorce projektowe
- Dalszy rozwój aplikacji klienckiej byłby bezpieczniejszy, kiedy przepisana zostałaby na TypeScript.
- Wdrożenie systemu loggowania w oparciu o ILogger.
- Implementacje panelu administratora wraz z rolą administratora.
- Wykorzystanie Microsoft identity do podziału użytkowników na role oraz wprowadzenie dodatkowych usprawnień autentykacji.
- Wdrożenie aplikacji na chmurę w celu zapewnienia dostępu zdalnego do aplikacji.
- Opakowanie aplikacji w kontener Docker aby, skrócić czas instalacji zależności.
- Implementacja funkcji okresowych kopii zapasowych systemu

# Zakończenie

Projekt wraz z dokumentacją oraz instrukcją uruchomienia znajduje się na moim profilu github. Serdecznie zapraszam do próby samodzielnego uruchomienia i testowania aplikacji.

Dodatkowo dla osób ceniących sobie swój czas, przygotowałem materiał w formie filmu na Youtube w, którym dokładnie prezentuje wszystkie funkcjonalności aplikacji. Dzięki temu zaoszczędzicie Państwo swój czas na procesie instalacji aplikacji i wymaganych zależności.

## LINKI

