ResearchCruiseApp

Organizacja i infrastruktura projektu

2025-03-18

Stanisław Nieradko	s193044
Krzysztof Nasuta	s193328
Paweł Pstrągowski	s193473
Bartłomiej Krawisz	s193319
Filip Dawidowski	s193433

1. Opis projektu i produktu

Nazwa projektu

ResearchCruiseApp

Adresowany problem

Projekt ResearchCruiseApp dostosowany jest do zarządzania rejsami badawczymi statku badawczego "Oceanograf" należącego do Instytutu Oceanografii Uniwersytetu Gdańskiego. Aplikacja ma na celu usprawnienie procesów związanych z rezerwacją, obsługą i organizacją rejsów badawczych, a także umożliwienie efektywnej komunikacji między różnymi interesariuszami. Prace na aplikacją zostały rozpoczęte w zeszłym roku jako projekt grupowy oraz inżynierski poprzedniego zespołu. Są one kontynuowane przez obecny zespół w ramach własnego projektu grupowego oraz dwóch prac inżynierskich.

Obszar zastosowania i rynek

Aplikacja została utworzona dla konkretnego klienta, którym jest biuro armatora Uniwersytetu Gdańskiego. Obejmuje ona zarządzanie rejsami badawczymi, obsługę zgłoszeń oraz komunikację między różnymi interesariuszami. Aplikacja ma na celu usprawnienie procesów administracyjnych związanych z organizacją rejsów badawczych. Może ona zostać wykorzystana przez innych klientów z branży morskiej, takich jak instytucje naukowe, ale z uwagi na dostosowanie do konkretnego klienta, wymagałoby to pewnych zmian i nie jest obecnie planowane.

Zakres projektu

Do zakresu pracy nad projektem należą następujące zadania

- Przepisanie aplikacji frontendowej z użyciem frameworku React i języka TypeScript.
- Utrzymywanie i rozwijanie aplikacji backendowej napisanej w frameworku ASP.NET Core.
- Implementacja nowych funkcjonalności zgodnie z wymaganiami klienta.
- Naprawa zgłoszonych błędów.

Do ograniczeń związanych z projektem należy

- Jako silnik bazodanowy wykorzystywany jest Microsoft SQL Server.

Inne współpracujące systemy

Aplikacja obecnie nie współpracuje z innymi systemami, jednakże w przyszłości planowane jest dodanie integracji z Active Directory / SSO Uniwersytetu Gdańskiego w celu ułatwienia zarządzania użytkownikami oraz procesem logowania.

Główne etapy projektu

- Zbieranie wymagań projektowych i przekazanie projektu od poprzedniego zespołu
- Przepisanie aplikacji frontendowej
- Implementacja kluczowych funkcjonalności (m. in. formularz C) oraz naprawa zgłoszonych błędów
- Automatyzacja testowania i wdrażania

Terminy

Ukończenie pracy nad formularzem C: 7 kwietnia 2025

Ukończenie pracy nad projektem: grudzień 2025

2. Interesariusze i użytkownicy

Interesariusze

Możemy wyróżnić następujące grupy interesariuszy:

- **Administratorzy systemu** - osoby zarządzające całością aplikacji
- **Biuro armatora Uniwersytetu Gdańskiego** - pracownicy odpowiedzialni za organizację rejsami
- **Kierownicy rejsów** - osoby odpowiedzialne za zarządzanie konkretnymi rejsami
- **Przełożeni** - osoby odpowiedzialne za zatwierdzanie rejsów, nie posiadają jednak dostępu do reszty aplikacji
- **Goście zewnętrzni** - osoby, które mogą przeglądać dane aplikacji

Użytkownicy końcowi

Aplikacja obsługuje cztery typy użytkowników:

- **Administrator**: Ma pełny dostęp do wszystkich danych i funkcjonalności aplikacji. Odpowiada za zarządzanie użytkownikami, konfigurację systemu oraz nadzorowanie poprawności działania aplikacji.
- **Armator**: Ma pełny dostęp do danych związanych z organizacją rejsów. Może zarządzać zgłoszeniami i rejsami. Posiada ograniczone możliwości zarządzania użytkownikami.
- **Kierownik rejsu**: Posiada dostęp wyłącznie do danych związanych z własnymi rejsami oraz zgłoszeniami, w których pełni rolę zastępcy. Nie posiada możliwości zarządzania innymi użytkownikami ani modyfikowania ogólnych danych rejsowych.
- **Gość**: Posiada tylko możliwość przeglądania danych bez możliwości edycji.

Dodatkowe wymagania użytkowników

- Bezpieczny dostęp do aplikacji.
- Dane prywatne przechowywane w bazie danych powinny być zabezpieczone przed nieautoryzowanym dostępem oraz przechowywane zgodnie z obowiązującymi przepisami prawa.
- Aplikacja powinna gwarantować braku utraty danych nawet w przypadku awarii systemu.
- Interfejs aplikacji powinien być intuicyjny i łatwy w obsłudze, aby użytkownicy mogli szybko i sprawnie wykonywać swoje zadania.

3. Zespół

Przy projekcie pracują wspólnie dwa zespoły składające się łącznie z 5 osób. Każdy zespół będzie końcowo odpowiedzialny za określone obszary projektu, jednakże w początkowej fazie prac wszyscy członkowie pracują wspólnie nad przejęciem projektu od poprzednich zespołów, poprawą jakości kodu oraz naprawą zgłoszonych błędów.

Zespół 1: Rozwój portalu wspierającego obsługę rejsów dla biura Armatora Wydział Oceanografii i Geografii UG

- Stanisław Nieradko (*193044*): Lider zarówno podzespołu oraz całości projektu. Odpowiedzialny za całokształt prac włączając w to architekturę oraz implementację przepisanego frontendu, utrzymanie backendu oraz zarządzanie zespołem.
- Bartłomiej Krawisz (*193319*): Docelowo odpowiedzialny za rozplanowanie i implementację strategii testowania projektu. W początkowej fazie współpracuje wraz z resztą zespołu nad refaktoryzacją kodu front-endu.

- Paweł Pstrągowski (*193473*): Odpowiedzialny za implementację zaplanowanych funkcjonalności. W początkowej fazie współpracuje wraz z resztą zespołu nad refaktoryzacją kodu front-endu.

Zespół 2: Realizacja niefunkcjonalnych wymagań dla portalu wspierającego obsługę rejsów Biura Armatora Wydział Oceanografii i Geografii UG.

- Krzysztof Nasuta (*193328*): Lider podzespołu. Odpowiedzialny za rozwój DevOps, utrzymanie backendu oraz planowanie i refaktoryzację kodu frontend.
- Filip Dawidowski (*193433*): Odpowiedzialny za implementację zaplanowanych funkcjonalności. W początkowej fazie współpracuje wraz z resztą zespołu nad refaktoryzacją kodu front-endu.

Oba podzespoły utrzymują komunikację za pomocą platformy Discord na której odbywają się regularne spotkania oraz wymiana informacji. Wszelkie plany lub zmiany w kodzie odbywają się po wcześniejszej konsultacji z pozostałymi członkami zespołu w formie zdalnej.

Dane kontaktowe członków zespołu:

Imię i nazwisko	Adres e-mail	Nick na Discordzie
Stanisław Nieradko	s193044@student.pg.edu.pl	KanarekLife
Krzysztof Nasuta	s193328@student.pg.edu.pl	nasus.
Paweł Pstrągowski	s193473@student.pg.edu.pl	oenea
Bartłomiej Krawisz	s193319@student.pg.edu.pl	ketrab20
Filip Dawidowski	s193433@student.pg.edu.pl	fil6164

4. Komunikacja w zespole i z interesariuszami

Komunikacja w zespole odbywa się za pomocą komunikatora Discord, gdzie odbywają się regularne spotkania oraz wymiana informacji. Wszelkie plany lub zmiany w kodzie odbywają się po wcześniejszej konsultacji z pozostałymi członkami zespołu w formie zdalnej. Spotkania mają miejsce średnio trzy razy w tygodniu, w zależności od potrzeb projektu oraz postępu prac.

Komunikacja z opiekunem projektu odbywa się za pomocą platformy Teams w formie spotkań online mających miejsce średnio raz co dwa lub trzy tygodnie. Podczas tych spotkań omawiane są postępy prac, plany na przyszłość oraz wszelkie problemy napotkane podczas realizacji projektu. Wszelkie ważne decyzje dotyczące projektu oraz komunikacji z klientem są konsultowane z opiekunem projektu.

Komunikacja z klientem odbywa się w nieregularnych interwałach za pomocą spotkań na platformie Teams oraz przy pomocy systemu *Issues* na platformie GitHub. Z uwagi na specyfikę projektu, komunikacja nie musi być na bieżąco, także większość informacji zbierana jest w trakcie spotkań lub w formie zgłoszeń na platformie GitHub, gdzie klient może zgłaszać błędy, propozycje zmian oraz nowe funkcjonalności. Na podstawie tej komunikacji zespół projektowy wraz z opiekunem projektu podejmuje decyzje dotyczące dalszego rozwoju aplikacji.

5. Współdzielenie dokumentów i kodu

Dokumentacja projektu jest tworzona przy pomocy narzędzia typst. Kod tworzący dokument przechowywany jest wraz z resztą kodu projektu w repozytorium Git hostowanym na opisanej dalej platformie GitHub.

Repozytorium projektu przechowywane jest na platformie GitHub pod adresem `https://github.com/KanarekLife/ResearchCruiseApp`. Repozytorium bazowane jest na uprzednim repozytorium, które zostało przekazane przez poprzedni zespół projektowy. Wszelkie zmiany w kodzie oraz dokumentacji są dokonywane za pomocą systemu kontroli wersji git oraz podlegają recenzji kodu przez pozostałych członków zespołu. W aktualnej fazie projektu najważniejszą gałęzią jest `task/frontend-refactor` w której odbywa się refaktoryzacja kodu front-endu. Gałąź `main` jest gałęzią produkcyjną, która zawiera najnowszą wersję aplikacji niezawierającą jednak najnowszych zmian i poprawek.

Porządek w repozytorium utrzymywany jest przez liderów obu podzespołów, którzy odpowiedzialni są za zarządzanie gałęziami oraz recenzję kodu. Wszelkie zmiany w kodzie muszą być zatwierdzone przez obu liderów zespołów przed scaleniem z gałęzią główną. Wszelkie zmiany w dokumentacji podlegają temu samemu procesowi co zmiany w kodzie. Szablon dokumentacji projektu jest dostępny w pliku `docs/template.typ`. Aktualne wersje dokumentów przechowywane są w folderze `docs` w repozytorium projektu obok ich plików źródłowych `.typ`, dzięki czemu są wersjonowane.

6. Narzędzia

- GitHub - platforma do hostowania repozytorium kodu
- Git - system kontroli wersji
- Discord - platforma do komunikacji w zespole
- Microsoft Teams - platforma do komunikacji z opiekunem projektu oraz klientem
- Typst - narzędzie do tworzenia dokumentacji projektu
- Visual Studio Code - środowisko programistyczne do pracy nad kodem front-endu
- JetBrains Rider - środowisko programistyczne do pracy nad kodem backendu
- `pnpm`, `node` - narzędzia do budowania i uruchamiania aplikacji frontendowej
- `React`, biblioteki TanStack, `react.motion` i inne - framework i biblioteki do tworzenia aplikacji frontendowej
- `.NET Core`, `dotnet` - framework i narzędzie do tworzenia aplikacji backendowej
- Microsoft SQL Server - baza danych wykorzystywana w aplikacji
- Docker Compose - narzędzie do zarządzania kontenerami Docker
- Docker - narzędzie do uruchamiania aplikacji w kontenerach

Testowanie aktualnie odbywa się ręcznie, jednakże planowane jest wdrożenie testów jednostkowych oraz integracyjnych w celu zapewnienia jakości kodu oraz uniknięcia błędów w przyszłości.