WOJSKOWA AKADEMIA TECHNICZNA

**im. Jarosława Dąbrowskiego**

**WYDZIAŁ CYBERNETYKI**



STUDIA I°

Temat: **Sprawozdanie z realizacji zadania z przedmiotu Projekt Zespołowy**

**INFORMATYKA**

…………………………………………………………………………

(kierunek studiów)

**INŻYNIERIA SYSTEMÓW**

…………………………………………………………………………

(specjalność)

|  |  |
| --- | --- |
| Wykonał: | Prowadzący: |
| **Radosław Relidzyński** | **dr inż. Jerzy STANIK** |

**Warszawa 2023**

Spis treści

[Wstęp 5](#_Toc137989503)

[Rozdział I. Opis dziedziny przedmiotu 6](#_Toc137989504)

[I.1. Opis organizacji 6](#_Toc137989505)

[I.2. Rozwiązania informatyczne zastosowane w organizacji 7](#_Toc137989506)

[I.3. Problemy organizacji związane z zastosowanymi rozwiązaniami informatycznymi 7](#_Toc137989507)

[I.4. Analiza SWOT 8](#_Toc137989508)

[I.5. Charakterystyka procesu, który będzie realizował serwis 8](#_Toc137989509)

[Rozdział II. Specyfikacja wymagań 9](#_Toc137989510)

[II.1. Wymagania funkcjonalne 9](#_Toc137989511)

[II.2. Wymagania pozafunkcjonalne 11](#_Toc137989512)

[II.3. Wymagania systemowe 12](#_Toc137989513)

[II.4. Diagram przypadków użycia 13](#_Toc137989514)

[II.5. Specyfikacja przypadków użycia 13](#_Toc137989515)

[Rozdział III. Metodyka realizacji i środowisko wytwórcze 18](#_Toc137989516)

[III.1. Metodyka 18](#_Toc137989517)

[III.2. Środowisko wytwórcze 18](#_Toc137989518)

[III.3. Podejście do tworzenia oprogramowania 22](#_Toc137989519)

[Rozdział IV. Projekt aplikacji 23](#_Toc137989520)

[IV.1. Diagram klas 23](#_Toc137989521)

[IV.2. Architektura 25](#_Toc137989522)

[IV.3. Projekt modułu klienta 26](#_Toc137989523)

[IV.4. Projekt modułu serwera 26](#_Toc137989524)

[IV.5. Projekt bazy danych 27](#_Toc137989525)

[IV.6. Projekt interfejsu 29](#_Toc137989526)

[Podsumowanie 32](#_Toc137989527)

[Bibliografia 33](#_Toc137989528)

[Spis rysunków 34](#_Toc137989529)

[Spis tabel 34](#_Toc137989530)

Wstęp

Temat projektu

Przedmiotem mojego projektu jest utworzenie społecznościowego serwisu typu helpdesk, czyli aplikacji internetowej pozwalającej jej użytkownikom na tworzenie oraz przeglądanie wpisów zawierających poradniki i instrukcje pokazujące jak rozwiązywać podstawowe problemy, na jakie może trafić użytkownik nie mający większych kompetencji w zakresie obsługi komputera.

Docelowo strona będzie zorientowana wokół instruktorów Związku Harcerstwa Polskiego, ale korzystać z niej będzie mógł każdy. Taki wybór grupy docelowej uwarunkowany jest tym, że sam jestem instruktorem tej organizacji i osobiście doświadczam sytuacji wskazujących na potrzebę powstania przestrzeni, w której ludzie nie posiadający obycia z komputerem czy Internetem mogli dowiedzieć się, w jaki sposób poradzić sobie w świecie cyfrowym.

Cel ćwiczenia

Celem ćwiczenia jest wykorzystanie dotychczas zebranej wiedzy z dziedziny inżynierii systemów informatycznych do stworzenia dokumentu opisującego realizację projektu na wspomniany przeze mnie temat.

Ćwiczenie nie obejmuje stworzenia całego systemu, a jedynie jego wizję wraz z planem realizacji.

1. Opis dziedziny przedmiotu
   1. Opis organizacji

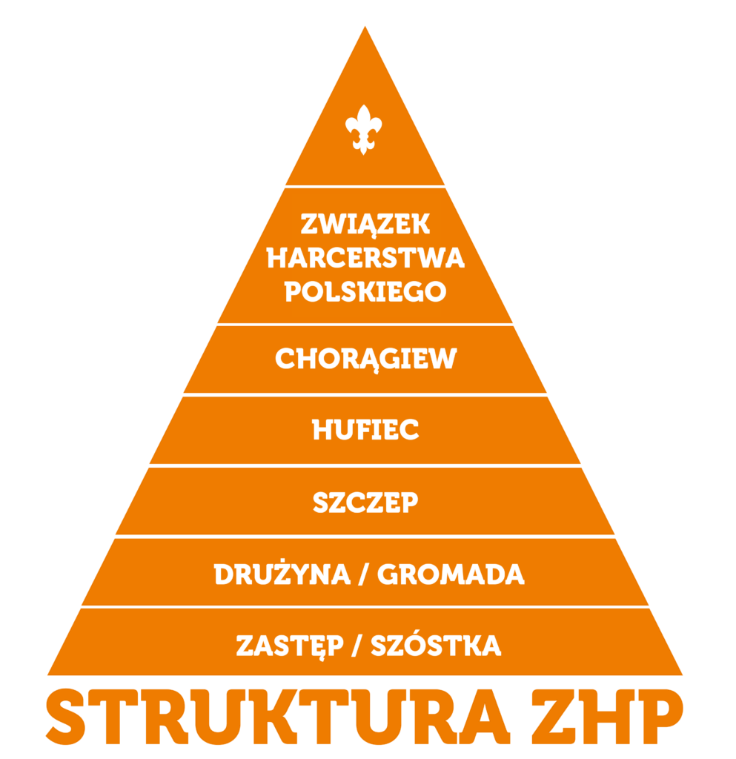
Obraz zawierający tekst

Opis wygenerowany automatycznie

1. Logo Związku Harcerstwa Polskiego

Związek Harcerstwa Polskiego (ZHP) jest to największa polska organizacja harcerska, powstała 9 listopada 1918 r., czyli na 2 dni przed odzyskaniem przez Polskę niepodległości. Jest to organizacja pożytku publicznego (OPP) typu non profit o charakterze społeczno-wychowawczym, zrzeszająca dzieci, młodzież i dorosłych wokół wartości takich jak przyjaźń, odpowiedzialność, patriotyzm czy ochrona przyrody.

Misją Związku Harcerstwa Polskiego jest wychowywanie młodzieży oraz wspieranie rozwoju i kształtowanie charakteru człowieka poprzez stawianie wyzwań. Wartości, którymi kieruje się organizacja wyrażone są przez prawo zucha, prawo harcerskie oraz zobowiązanie instruktorskie.



1. Struktura Związku Harcerstwa Polskiego

Związek Harcerstwa Polskiego posiada zhierarchizowaną strukturę opartą na powyższym schemacie (odpowiednio od dołu reprezentują one odpowiednik zespołu zadaniowego, grupę zespołów działającą najczęściej przy szkole, osiedle mieszkaniowe / miejscowość, dzielnicę / powiat, województwo lub Warszawę – chorągiew Stołeczna).

Z perspektywy państwa ciałami prawnymi są Chorągwie, to one odpowiadają za zadania administracyjne organizacji. Dodatkowo, obowiązki prawne posiadają również wszyscy instruktorzy, niezależnie od szczebla, na jakim działają. Podpisują oni umowę wolontariacką oraz klauzulę RODO.

Struktura organizacji będzie niezbędna przy wdrażaniu całego projektu, ponieważ w ramach różnych elementów struktury będą różne zadania do wykonania, czy to w zakresie promowania serwisu, czy jego dalszego rozwijania.

* 1. Rozwiązania informatyczne zastosowane w organizacji

W celu stałego rozwoju organizacji, jej władze naczelne regularnie wprowadzają w życie projekty mające na celu zwiększenie innowacyjności organizacji. Do tego stosowane są między innymi takie narzędzia jak:

* Microsoft 365 – narzędzie do sprawnej komunikacji, zbierania dokumentacji oraz planowania działań. Pozwala to na sprawne grupowanie ludzi, zarządzenie dostępem do informacji oraz bezpieczne ich przechowywanie.
* Tipi – oficjalna ewidencja członków wraz z ich historią członkostwa (posiadane odznaczenia, przynależność do jednostek organizacji)
* Harcerski Serwis Szkoleniowy – platforma do nauki podstawowych kompetencji instruktorów, takich jak znajomość RODO czy organizacja obozu (wszyscy instruktorzy organizacji zobowiązani są do podpisania konkretnej umowy)
* Strony na Facebooku – przestrzeń do promocji organizacji oraz jej działań
* Konta bankowe – od niedawna dostępne nawet na poziomie drużyn, pozwalają na łatwiejszy dostęp do środków oraz sprawniejsze zarządzanie budżetem.
  1. Problemy organizacji związane z zastosowanymi rozwiązaniami informatycznymi

Główne problemy organizacji wynikające z wykorzystywanych rozwiązań informatycznych można zamknąć w poniższej liście:

* Użytkownicy systemu Microsoft 365 notorycznie napotykają na problemy z logowaniem, brakiem dostępu do niektórych funkcji lub niestabilnością działania platformy, co utrudnia efektywną pracę.
* Zbyt wiele instruktorów ZHP przechowuje dane wrażliwe na prywatnych komputerach i dyskach wirtualnych, co może prowadzić do naruszeń zasad RODO oraz naraża organizację na ryzyko utraty poufnych informacji
* Istnieją użytkownicy nieposiadający wprawy w podstawowej obsłudze komputera
* Użytkownicy nie posiadają wprawy w zakresie obsługi podstawowych narzędzi biurowych (Word, Excel, Outlook, Teams)
* Brak wprawy użytkowników narzędzi informatycznych prowadzi do utrudnień w codziennym działaniu organizacji, obniża efektywność oraz zaburza komunikację.
  1. Analiza SWOT

Poniżej przedstawiona jest tabela zawierająca analizę SWOT rozwiązań informatycznych w organizacji.

1. Analiza SWOT wpływu rozwiązań informatycznych na działania w ZHP

|  |  |
| --- | --- |
| S – mocne strony | W – słabe strony |
| Duże bezpieczeństwo danych  Łatwy dostęp na wielu urządzeniach  Możliwość współpracy użytkowników  Automatyzacja istotnych procesów w organizacji  Możliwość prowadzenia łatwej w przeglądaniu i precyzyjnej dokumentacji | Problemy z logowaniem do usług Microsoft 365  Narzędzia nie są dostosowane do osób nie wprawionych w obsługę narzędzi informatycznych |
| O – szanse | T - zagrożenia |
| Dalszy rozwój pozwala na znaczną eliminację zagrożeń wynikających z nieautoryzowanym dostępem do danych lub ich nieoczekiwaną utratą  Wraz z kolejnymi udogodnieniami instruktorzy mogą mniej skupiać się na zadaniach w zakresie logistki, a bardziej na wartości merytorycznej zajęć przygotowywanych dla harcerzy | Wielu użytkowników rezygnuje z tych rozwiązań na rzecz mniej bezpiecznych i bardziej ograniczonych, ale takich, z którymi są zaznajomieni  Użytkownicy, którzy nie mają wprawy w podstawowej obsłudze komputera nie radzą sobie z dostępnymi dla nich narzędziami  Użytkownicy nie wiedzą, skąd mogą czerpać wiedzę o tym, jak obsługiwać dostępne im narzędzia |

* 1. Charakterystyka procesu, który będzie realizował serwis

Zadaniem serwisu będzie stworzenie platformy do tworzenia i przeglądania wpisów z poradami prowadzącymi do rozwiązywania problemów użytkownika podstawowych narzędzi informatycznych na komputerze, z dużym ukierunkowaniem na te zastosowane w Związku Harcerstwa Polskiego.

Funkcjonalności serwisu będą reagować na wyniki analizy SWOT poprzez wpieranie mocnych stron i szans oraz odpowiadanie na potrzeby wynikające ze słabych stron i zagrożeń. Z tego powodu powinien on działać w następujących kontekstach:

* Zwiększać kompetencje użytkowników w zakresie korzystania z komputera i Internetu, niezależnie od poziomu zaawansowania
* Wychodzić naprzeciw problemom, z jakimi spotyka się użytkownik
* Być łatwy w obsłudze
* Posiadać porady dla użytkowników o różnym poziomie obeznania z komputerem
* Zachęcać użytkowników do wybierania profesjonalnych i bezpiecznych rozwiązań informatycznych
* Gromadzić wszelkie porady w zakresie obsługi popularnych narzędzi komputerowych

1. Specyfikacja wymagań
   1. Wymagania funkcjonalne

Wymagania funkcjonalne – specyfikacja określająca oczekiwane zachowanie systemu informatycznego w kontekście funkcji, które powinno spełniać. Wymagania funkcjonalne precyzują, co system ma robić oraz jakie zadania ma wykonywać w celu zrealizowania określonych celów biznesowych lub technicznych. Poniższa tabela przedstawia specyfikację wymagań funkcjonalnych dla projektu.

1. Spis wymagań funkcjonalnych

|  |  |  |  |
| --- | --- | --- | --- |
| L.p. | Wymaganie | Warunek | Opis wymagania |
| 1 | Rejestracja użytkowników | Obligatoryjne | System powinien umożliwiać rejestrację nowych użytkowników przy pomocy nazwy użytkownika, maila oraz hasła. |
| 2 | Logowanie użytkowników | Obligatoryjne | System powinien umożliwiać logowanie zarejestrowanych użytkowników za pomocą maila oraz hasła. |
| 3 | Dostęp dla niezalogowanych | Opcjonalne | System powinien umożliwiać dostęp do serwisu dla niezalogowanych użytkowników w sposób ograniczony, uniemożliwiający dodawanie wpisów. |
| 4 | Zarządzanie kontem | Obligatoryjne | System powinien udostępniać wszystkie funkcje CRUD w zakresie zarządzania danymi profilu użytkownika. |
| 5 | Zarządzanie autorskimi wpisami | Obligatoryjne | System powinien udostępniać wszystkie funkcje CRUD w zakresie autorskich wpisów użytkownika. |
| 6 | Publikowanie wpisów | Obligatoryjne | System powinien umożliwiać publikację autorskiego wpisu użytkownika. |
| 7 | Przechowywanie wpisów | Obligatoryjne | System powinien przechowywać wszystkie utworzone wpisy jako niezależne od siebie obiekty wraz z ich datą i godziną utworzenia oraz publikacji. |
| 8 | Przeglądanie wpisów | Obligatoryjne | System powinien udostępniać możliwość przeglądania, wyszukiwania i sortowania wpisów wszystkich użytkowników. |
| 9 | Zarządzanie uprawnieniami | Obligatoryjne | System powinien umożliwiać administratorowi zarządzanie poziomami uprawnień użytkowników. |
| 10 | Zarządzanie statusami | Obligatoryjne | System powinien przyporządkowywać do każdego wpisu status odpowiedni dla stanu, w jakim znajduje się wpis. |
| 11 | Zgłaszanie zapotrzebowania na wpis | Opcjonalne | System powinien umożliwiać zgłaszanie przez danego użytkownika zapotrzebowania na wpis z podaniem problematyki, jaką ma poruszać. Zgłoszenie ma przechowywać datę i godzinę jego przesłania. |
| 12 | Zgłaszanie błędu we wpisie | Obligatoryjne | System powinien umożliwiać użytkownikowi zgłaszanie uwag lub propozycji poprawek do wpisów. Zgłoszenie ma przechowywać datę i godzinę jego przesłania. |
| 13 | Historia wpisów zalogowanego użytkownika | Opcjonalne | System powinien przechowywać i udostępniać historię przeglądanych wpisów zalogowanego użytkownika. |
| 14 | Historia wpisów niezalogowanego użytkownika | Opcjonalne | System powinien udostępniać historię przeglądanych wpisów zalogowanego użytkownika, przechowywaną w ramach sesji użytkownika. |
| 15 | Kategorie wpisów | Opcjonalne | System powinien przechowywać informacje o kategoriach wpisów. |
| 16 | Formaty wpisów | Opcjonalne | System powinien umożliwiać tworzenie wpisów w różnym formacie. |
| 17 | Ulubione wpisy | Opcjonalne | System powinien umożliwiać dodawanie wpisów do listy ulubionych wpisów użytkownika. |
| 18 | Odnośniki do innych poradników | Opcjonalne | Tworzenie wpisów powinno obejmować wstawianie hiperłączy do poradników z innych stron internetowych. |
| 19 | Rola redaktora | Obligatoryjne | System powinien umożliwiać nadanie zalogowanemu użytkownikowi statusu redaktora. |
| 20 | Redaktor – ocena wpisów | Obligatoryjne | Redaktor powinien mieć możliwość oceniania niezatwierdzonego wpisu przydzielonego przez redaktora naczelnego, w celu jego zatwierdzenia lub odrzucenia. |
| 21 | Redaktor – obsługa zgłoszeń | Obligatoryjne | Redaktor powinien mieć możliwość obsługi zgłoszenia przydzielonego przez redaktora w celu podjęcia decyzji o dokonaniu sugerowanych poprawek lub ich odrzucenia. |
| 22 | Informacja zwrotna dla użytkownika | Opcjonalne | System powinien podawać informację zwrotną użytkownikowi o decyzji w sprawie zatwierdzenia jego wpisu lub rozpatrzenia zgłoszenia. |
| 23 | Redaktor naczelny | Obligatoryjne | System powinien umożliwiać nadanie redaktorowi roli redaktora naczelnego, który będzie odpowiadał za przydzielanie zgłoszeń do redaktorów. |

* 1. Wymagania pozafunkcjonalne

Wymaganie pozafunkcjonalne – specyfikacja określająca oczekiwania jakościowe wobec systemu. Nie opisuje konkretnych funkcjonalności, a jedynie warunki, które system powinien spełnić w zakresie wydajności, niezawodności, bezpieczeństwa, kompatybilności itp. Poniższa tabela przedstawia specyfikację wymagań pozafunkcjonalnych dla projektu.

1. Spis wymagań pozafunkcjonalnych

|  |  |  |
| --- | --- | --- |
| L.p. | Wymaganie | Opis wymagania |
| 1 | Bezpieczeństwo kont | System powinien zapewniać autoryzację i uwierzytelnianie użytkowników zachowujący bezpieczeństwo danych o użytkownikach. |
| 2 | Bezpieczeństwo informacji | System powinien zawierać zabezpieczenia przed atakami typu SQL injection, Dot Dot Slash itp. |
| 3 | Niezawodność działania | System powinien być dostępny dla użytkowników przez 99% czasu w ciągu roku. |
| 4 | Użyteczność interfejsu | Interfejs użytkownika powinien być prosty, intuicyjny i zgodny z zasadami projektowania User Experience. |
| 5 | Użyteczność dla różnych przeglądarek | System powinien obsługiwać różne przeglądarki internetowe. |
| 6 | Użyteczność dla różnych urządzeń | Interfejs użytkownika powinien być dostosowany do urządzenia, z którego korzysta użytkownik. |
| 7 | Niezawodność działania wielu użytkowników | System powinien być w stanie obsługiwać wielu użytkowników w jednym momencie bez występowania konfliktów w dostępie do funkcjonalności. |
| 8 | Zgodność z RODO | System powinien spełniać wymogi normy RODO (Rozporządzenia o Ochronie Danych Osobowych), |
| 9 | Zgodność ISO | System powinien być zgodny z normami bezpieczeństwa informacji, takimi jak ISO 27001. |
| 10 | Wydajność dostępu | System powinien umożliwiać szybki dostęp do danych, z czasem odpowiedzi poniżej 5 sekund. |
| 11 | Monitorowanie działania systemu | System powinien posiadać mechanizmy monitorowania i zarządzania zasobami poprzez logi oraz pogląd stanu systemu dla administratora |
| 12 | Odporność na awarie | System powinien być odporny na awarie oprogramowania i sieciowe poprzez przewidywanie i zapobieganie potencjalnych błędów użytkownika. |

* 1. Wymagania systemowe

Wymaganie systemowe - specyfikacje techniczne i parametry sprzętowe oraz oprogramowanie, które muszą być spełnione, aby dana aplikacja komputerowa, program, gra komputerowa, system operacyjny lub inny rodzaj oprogramowania działał prawidłowo. Poniższa tabela przedstawia specyfikację wymagań systemowych dla projektu.

1. Spis wymagań systemowych

|  |  |  |
| --- | --- | --- |
| L.p. | Wymaganie | Opis wymagania |
| 1 | System operacyjny | System powinien być kompatybilny z systemem operacyjnym Linux, Windows 10 lub nowszym, Android 10 lub nowszym. |
| 2 | Przeglądarka | System powinien być zgodny z najnowszymi wersjami przeglądarek internetowych, takich jak Google Chrome, Mozilla Firefox, Safari, Microsoft Edge itp. |
| 3 | Język programowania | System powinien być oparty na języku programowania Python wraz z frameworkiem Django. |
| 4 | Baza danych | System powinien korzystać z bazy danych umożliwiające przechowywanie wpisów o bardzo różnym formacie i wadze. |
| 5 | Rozdzielczość | System powinien być dostosowany do różnych rozdzielczości ekranów, takich jak ekran komputera, tablet, smartfon itp. |

* 1. Diagram przypadków użycia

Diagram przypadków użycia UML – graficzna reprezentacja interakcji pomiędzy aktorami (użytkownikami) a systemem, przedstawiająca różne przypadki użycia w celu zrozumienia funkcjonalności systemu. Poniższy rysunek przedstawia schemat diagramu przypadków użycia projektu.

Obraz zawierający tekst, zrzut ekranu, diagram

Opis wygenerowany automatycznie

1. Diagram przypadków użycia systemu
   1. Specyfikacja przypadków użycia

W tym rozdziale opisane zostały szczegóły dotyczące przypadków użycia. Poniższe tabele zawierają opisy przypadków użycia oraz opisy aktorów, a poniżej przedstawione są przykładowe scenariusze przypadków użycia.

1. Opis przypadków użycia z diagramu przypadków użycia UML

|  |  |  |  |
| --- | --- | --- | --- |
| L.p. | Przypadek użycia | Opis przypadku użycia | Aktor |
| 1 | Przeglądanie wpisów | Użytkownik ma możliwość przeglądania dostępnych wpisów w systemie. | Użytkownik |
| 2 | Przeglądanie ostatnio oglądanych wpisów | Użytkownik może zobaczyć listę ostatnio oglądanych wpisów, aby łatwo do nich powrócić. | Użytkownik |
| 3 | Zgłaszanie błędu we wpisie | Użytkownik może zgłosić błąd w konkretnym wpisie, informując o tym redaktorów. | Użytkownik |
| 4 | Zgłaszanie zapotrzebowania na wpis | Użytkownik może zgłosić swoje zapotrzebowanie na nowy wpis, sugerując temat lub treść. | Użytkownik |
| 5 | Logowanie | Użytkownik może zalogować się do systemu, uzyskując dostęp do dodatkowych funkcji i informacji. | Zalogowany użytkownik |
| 6 | Rejestracja | Użytkownik może zarejestrować się w systemie, tworząc nowe konto użytkownika. | Zalogowany użytkownik |
| 7 | Wylogowanie | Zalogowany użytkownik może wylogować się z systemu, zakończając sesję. | Zalogowany użytkownik |
| 8 | Tworzenie wpisów | Zalogowany użytkownik może tworzyć nowe wpisy, wprowadzając treść i informacje. | Zalogowany użytkownik |
| 9 | Publikowanie wpisów | Zalogowany użytkownik może publikować swoje stworzone wpisy, czyniąc je widocznymi dla innych użytkowników. | Zalogowany użytkownik |
| 10 | Edytowanie autorskich wpisów | Zalogowany użytkownik może edytować swoje autorskie wpisy, wprowadzając zmiany w treści lub informacjach. | Zalogowany użytkownik |
| 11 | Dodawanie wpisów do ulubionych | Zalogowany użytkownik może dodać wybrany wpis do listy ulubionych, aby łatwo do niego powrócić w przyszłości. | Zalogowany użytkownik |
| 12 | Przeglądanie historii wpisów | Zalogowany użytkownik ma dostęp do historii swoich wcześniejszych wpisów, umożliwiając ich przeglądanie. | Zalogowany użytkownik |
| 13 | Przydzielanie zgłoszeń do redaktorów | Redaktor naczelny może przydzielić zgłoszenia dotyczące zmian w wpisach innym redaktorom w celu ich obsługi. | Redaktor naczelny |
| 14 | Obsłużenie zgłoszenia – zastosowanie zmian we wpisie | Redaktor ma możliwość wprowadzenia zalecanych zmian zgłoszonych w wpisie. | Redaktor |
| 15 | Obsłużenie zgłoszenia – odrzucenie zgłoszenia (z podaniem powodu) | Redaktor może odrzucić zgłoszenie zmian w wpisie, udzielając powodu takiej decyzji. | Redaktor |
| 16 | Ocena wpisu – zatwierdzenie wpisu | Redaktor może zatwierdzić wpis, uznając go za odpowiedni i gotowy do publikacji. | Redaktor |
| 17 | Ocena wpisu – odrzucenie wpisu (z podaniem powodu) | Redaktor może odrzucić wpis, podając uzasadnienie swojej decyzji. | Redaktor |
| 18 | Zapisywanie daty i godziny zgłoszenia | Zegar systemowy zapisuje datę i godzinę zgłoszenia konkretnego wpisu lub żądania. | Zegar systemowy |
| 19 | Zapisywanie daty i godziny publikacji wpisu | Zegar systemowy rejestruje datę i godzinę publikacji wpisu, co umożliwia śledzenie chronologii treści. | Zegar systemowy |
| 20 | Zapisywanie daty i godziny utworzenia wpisu | Zegar systemowy zapisuje datę i godzinę utworzenia wpisu, dostarczając informacji o jego powstaniu. | Zegar systemowy |

1. Opis aktorów z diagramu przypadków użycia UML

|  |  |  |
| --- | --- | --- |
| L.p. | Aktor | Opis aktora |
| 1 | Użytkownik | Osoba korzystająca z systemu, która ma możliwość przeglądania wpisów, zgłaszania błędów, zgłaszania zapotrzebowania na nowe wpisy oraz przeglądania historii wpisów. Użytkownik może również przeglądać ostatnio oglądane wpisy i dodawać wpisy do ulubionych. Nie ma dostępu do zaawansowanych funkcji, takich jak tworzenie, edytowanie lub publikowanie wpisów. |
| 2 | Zalogowany użytkownik | Osoba, która posiada aktywne konto w systemie i zalogowała się. Zalogowany użytkownik ma dostęp do pełnej funkcjonalności systemu, w tym tworzenia, edytowania i publikowania wpisów. Może również dodawać wpisy do ulubionych, przeglądać historię wpisów oraz korzystać z innych zaawansowanych funkcji, takich jak edytowanie autorskich wpisów i przeglądanie ostatnio oglądanych wpisów. |
| 3 | Redaktor naczelny | Osoba odpowiedzialna za zarządzanie redakcją i procesem redakcyjnym w systemie. Redaktor naczelny ma uprawnienia do przydzielania zgłoszeń do redaktorów, nadzorowania obsługi zgłoszeń oraz podejmowania decyzji dotyczących zatwierdzenia lub odrzucenia wpisów. Jest również odpowiedzialny za koordynację pracy redaktorów i zapewnienie wysokiej jakości treści w systemie. |
| 4 | Redaktor | Osoba odpowiedzialna za redagowanie i ocenę wpisów zgłaszanych przez użytkowników. Redaktor ma możliwość obsługi zgłoszeń, wprowadzania zalecanych zmian we wpisach oraz podejmowania decyzji dotyczących zatwierdzenia lub odrzucenia wpisów. Pracuje pod nadzorem redaktora naczelnego i współpracuje z innymi redaktorami w celu utrzymania wysokiego standardu treści w systemie. |
| 5 | Zegar systemowy | Komponent systemu odpowiedzialny za śledzenie daty i godziny. Zegar systemowy jest używany do rejestrowania czasu zgłoszenia wpisów, daty publikacji oraz daty i godziny utworzenia wpisów. Zapewnia dokładność danych czasowych w systemie i umożliwia monitorowanie chronologii działań. |

Opis scenariuszy przypadków użycia:

* + - 1. Przeglądanie i dodawanie wpisów ulubionych
         * Aktorzy: Użytkownik, Zalogowany użytkownik
         * Przypadki użycia: Przeglądanie wpisów, Przeglądanie ostatnio oglądanych wpisów, Logowanie, Dodawanie wpisów do ulubionych
         * Scenariusz:

Użytkownik otwiera stronę główną systemu.

Użytkownik przegląda dostępne wpisy, czytając ich tytuły i opisy.

Użytkownik znajduje interesujący wpis i klika na przycisk "Czytaj więcej".

Użytkownik jest przekierowywany do strony z pełną treścią wpisu.

Użytkownik loguje się do systemu, aby uzyskać dodatkowe funkcje.

Po zalogowaniu się, użytkownik wraca do strony wpisu i klika na przycisk "Dodaj do ulubionych".

System dodaje wpis do listy ulubionych użytkownika.

Użytkownik może teraz łatwo odnaleźć ten wpis w przyszłości, korzystając z listy ulubionych.

* + - 1. Zgłaszanie błędu w wpisie
         * Aktorzy: Użytkownik, Redaktor naczelny
         * Przypadki użycia: Przeglądanie wpisów, Zgłaszanie błędu we wpisie, Logowanie, Obsłużenie zgłoszenia – zastosowanie zmian we wpisie (przez redaktora), Zapisywanie daty i godziny zgłoszenia (przez zegar systemowy)
         * Scenariusz:

Użytkownik czyta wpis w systemie i zauważa błąd w treści lub informacjach.

Użytkownik klika na przycisk "Zgłoś błąd" obok wpisu.

System otwiera formularz zgłaszania błędu, w którym użytkownik może opisać problem.

Użytkownik wpisuje szczegóły dotyczące błędu i przesyła zgłoszenie.

Redaktor naczelny otrzymuje powiadomienie o nowym zgłoszeniu.

Redaktor naczelny loguje się do systemu i otwiera zgłoszenie.

Redaktor naczelny analizuje zgłoszenie i przekazuje je odpowiedniemu redaktorowi.

Redaktor wprowadza zmiany w wpisie, naprawiając zgłoszony błąd.

Po zakończeniu edycji, redaktor naczelny zatwierdza wprowadzone zmiany.

* + - 1. Tworzenie i publikowanie nowego wpisu
         * Aktorzy: Zalogowany użytkownik, Redaktor naczelny, Redaktor
         * Przypadki użycia: Logowanie, Tworzenie wpisów, Publikowanie wpisów, Ocena wpisu – zatwierdzenie wpisu (przez redaktora naczelnego), Zapisywanie daty i godziny publikacji wpisu (przez zegar systemowy)
         * Scenariusz:

Zalogowany użytkownik otwiera panel użytkownika w systemie.

Użytkownik klika na przycisk "Nowy wpis" lub podobną opcję.

System otwiera edytor wpisów, w którym użytkownik może wprowadzać treść, dodawać obrazy itp.

Użytkownik pisze treść wpisu, dodaje tytuł, kategorie i inne informacje.

Po zakończeniu tworzenia wpisu, użytkownik klika na przycisk "Opublikuj".

Wprowadzony wpis zostaje zapisany w systemie i oznaczony jako "W trakcie recenzji".

Redaktor naczelny otrzymuje powiadomienie o nowym wpisie do zatwierdzenia.

Redaktor naczelny loguje się do systemu i otwiera wpis do recenzji.

Redaktor naczelny zapoznaje się z nowym wpisem i przekazuje go odpowiedniemu redaktorowi.

Jeśli wpis spełnia kryteria i jest odpowiedni, redaktor zatwierdza go.

System publikuje wpis na stronie głównej i staje się widoczny dla innych użytkowników.

1. Metodyka realizacji i środowisko wytwórcze
   1. Metodyka

Metodyka – szerokie podejście do procesu wytwarzania oprogramowania, określające różne aspekty zarządzania projektem takie jak planowanie oraz wykonywanie prac, określanie zespołu projektowego, sposób określenia, kiedy system jest gotowy.

**Wybrana metodyka: XPrince**

Opis metodyki:

* XPrince (eXtreme PRogramming IN Controlled Environments) jest jedną ze zwinnych metodyk wytwarzania oprogramowania, która za cel obiera sobie wyważenie między zwinnością, a dyscypliną, biorąc najlepsze aspekty każdej z nich.
* XPrince bazuje na 3 innych metodykach: XP, PRINCE2, RUP.

Uzasadnienie decyzji:

1. XPrince zapewnia elastyczność i skalowalność, przez co pozwala na adaptację projektu do stale zmieniającej się rzeczywistości, która wymusza czasami aktualizację wymagań.
2. XPince stosuje podejście iteracyjne, dzięki czemu pozwala na dostarczanie gotowych elementów systemu bez konieczności jego pełnego wykonania.
3. XPrince uwzględnia zaangażowanie użytkowników, kładzie duży nacisk na jak najlepsze zrozumienie ich potrzeb w celu stworzenia oprogramowania jak najbardziej dostosowanego pod ich oczekiwania.

Etapy tworzenie oprogramowania w metodyce XPrince to:

1. Rozpoczęcie projektu
2. Inicjacja projektu
3. Elaboracja
4. Wydanie
5. Zamknięcie projektu
   1. Środowisko wytwórcze

Poniższe tabele przedstawiają kolejne etapy realizacji projektu w ramach metodyki XPrince. Każda tabela to jeden etap.

1. Etap 1 Rozpoczęcie projektu, metodyka XPrince

|  |  |
| --- | --- |
| **Zadania** | * **Wybór zespołu zarządzającego projektem** * **Wybór zespołu wykonawczego projektu** * **Przygotowanie wizji i koncepcji systemu** * **Wstępne opracowanie biznesowe** * **Specyfikacja wymagań** * **Zaplanowanie inicjacji systemu** |
| Metody | * Rozmowa - kontakt z klientem * Analiza * Analiza biznesowa |
| Techniki | * Burza mózgów * Diagramy UML * Diagramy BPMN * BPML |
| Narzędzia | * draw.io – koncepcja systemu (tworzenie diagramów) * Zoom – kontakt z klientem * Microsoft Teams - organizacja zespołów * Aurea Software – analiza biznesowa * Microsoft Word – dokumentacja specyfikacji |
| Artefakty | * Zebrany zespół zarządzający projektem oraz wykonujący projekt (struktura organizacyjna) * Udokumentowana specyfikacja wymagań |

1. Etap 2 Inicjacja, metodyka XPrince

|  |  |
| --- | --- |
| **Zadania** | * **Przygotowanie planu projektu** * **Zaproponowanie początkowej architektury** * **Opracowanie aspektu biznesowego** * **Przygotowanie środowiska organizacyjnego** * **Wyznaczenie kryteriów oceny jakości systemu** * **Zaplanowanie kolejnych wydań systemu** * **Zaplanowanie etapu elaboracji** |
| Metody | * Kontakt z klientem * UC Points * Delficka * Rozmowa * Analiza biznesowa |
| Techniki | * Diagramy BPMN * Diagramy przypadków użycia UML * Diagram Ishikawy * Quality Gate * BPML |
| Narzędzia | * draw.io / Visual Paradigm – tworzenie diagramów * Microsoft Teams – komunikacja * Microsoft Word – dokumentacja specyfikacji * Aurea Software – analiza biznesowa * Sonarqube – wyznaczanie kryteriów oceny jakości (Quality Gate |
| Artefakty | * Gotowy plan projektu i wstępna propozycja architektury * Opracowane środowisko organizacyjne (lista narzędzi) * Specyfikacja kryteriów oceny jakości systemu |

1. Etap 3 Elaboracja, metodyka XPrince

|  |  |
| --- | --- |
| **Zadania** | * **Specyfikacja mechanizmów architektonicznych** * **Ocena ryzyka** * **Opracowanie szkieletu projektu** * **Dopracowanie planu (menadżer) i wymagań (architekt)** |
| Metody | * UC Points * Delficka * Analiza jakościowa * MDA * Rozmowa - kontakt z klientem |
| Techniki | * Diagramy UML * Produkcja kodu * Gra Planistyczna * Burza mózgów * Tworzenie metamodelu orz modelu implementacyjnego |
| Narzędzia | * draw.io / Visual Paradigm – tworzenie diagramów * Kryptos72 – ocena ryzyka * Visual Studio / JetBrains – IDE * Kryptos72 – ocena ryzyka * Zoom – kontakt z klientem * Microsoft Word – dokumentacja mechanizmów architektonicznych |
| Artefakty | * Szkielet projektu z definicją klas i metod * Dopracowany plan projektu * Gotowy model architektury systemu * Rejestr ryzyka |

1. Etap 4 Wydanie, metodyka XPrince

|  |  |
| --- | --- |
| **Zadania** | * **Produkcja kodu i przypadków testowych** * **Przygotowanie testów akceptacyjnych** * **Sprawdzanie zgodności z wymaganiami** * **Wdrożenie** |
| Metody | * Peer review * Kontrola wersji * Dziel i zwyciężaj |
| Techniki | * Gra Planistyczna * Produkcja kodu * Równoległe programowanie i testowanie (TDD) * Kontakt z klientem (szczególnie podczas wdrażania) * Dzielenie zadań na podzadania |
| Narzędzia | * Visual Studio / JetBrains – IDE do wytwarzania oprogramowania * Zoom – kontakt z klientem * Git, TFS – system kontroli wersji * Jenkins – automatyzacja testów * Sonarqube – ocena jakości * Jira – zarządzanie statusem kodu (w trakcie pracy, gotowy do Peer review) * FitNesse – tworzenie testów akceptacyjnych |
| Artefakty | * Kolejne przetestowane i wdrożone wersje systemu |

1. Etap 5 Zamknięcie projektu, metodyka XPrince

|  |  |
| --- | --- |
| **Zadania** | * **Sprawdzenie kompletności oprogramowania** * **Przygotowanie dokumentacji technicznej** * **Ocena systemu w trakcie jego pracy** |
| Metody | * Graficzna * Analiza jakościowa * Rozmowa - kontakt z klientem * Ewaluacja |
| Techniki | * Diagramy BPMN * Diagramy aktywności * Diagramy przypadków użycia * Burza mózgów – ocena efektów pracy * Opracowanie i analiza dokumentacji |
| Narzędzia | * draw.io – diagramy do dokumentacji * Microsoft Word – tworzenie dokumentacji * Zoom – spotkania podsumowujące kompletność i ocenę systemu * Sonarqube – ocena jakości * Kryptos72 – ocena ryzyka |
| Artefakty | * Gotowy system * Opracowana dokumentacja techniczna * Ocena jakościowa systemu |

* 1. Podejście do tworzenia oprogramowania

W ramach realizacji projektu zastosowane zostanie podejście zwinne oraz iteracyjne.

Podejście zwinne w tworzeniu oprogramowania zakłada adaptacyjność i elastyczność wobec zmieniających się wymagań i warunków projektowych.

Manifest zwinności:

1. Ludzie i interakcje ponad procesy i narzędzia
2. Działające oprogramowanie ponad szczegółową dokumentację
3. Współpraca z klientem ponad negocjacje umów
4. Reagowanie na zmiany ponad realizację założonego planu

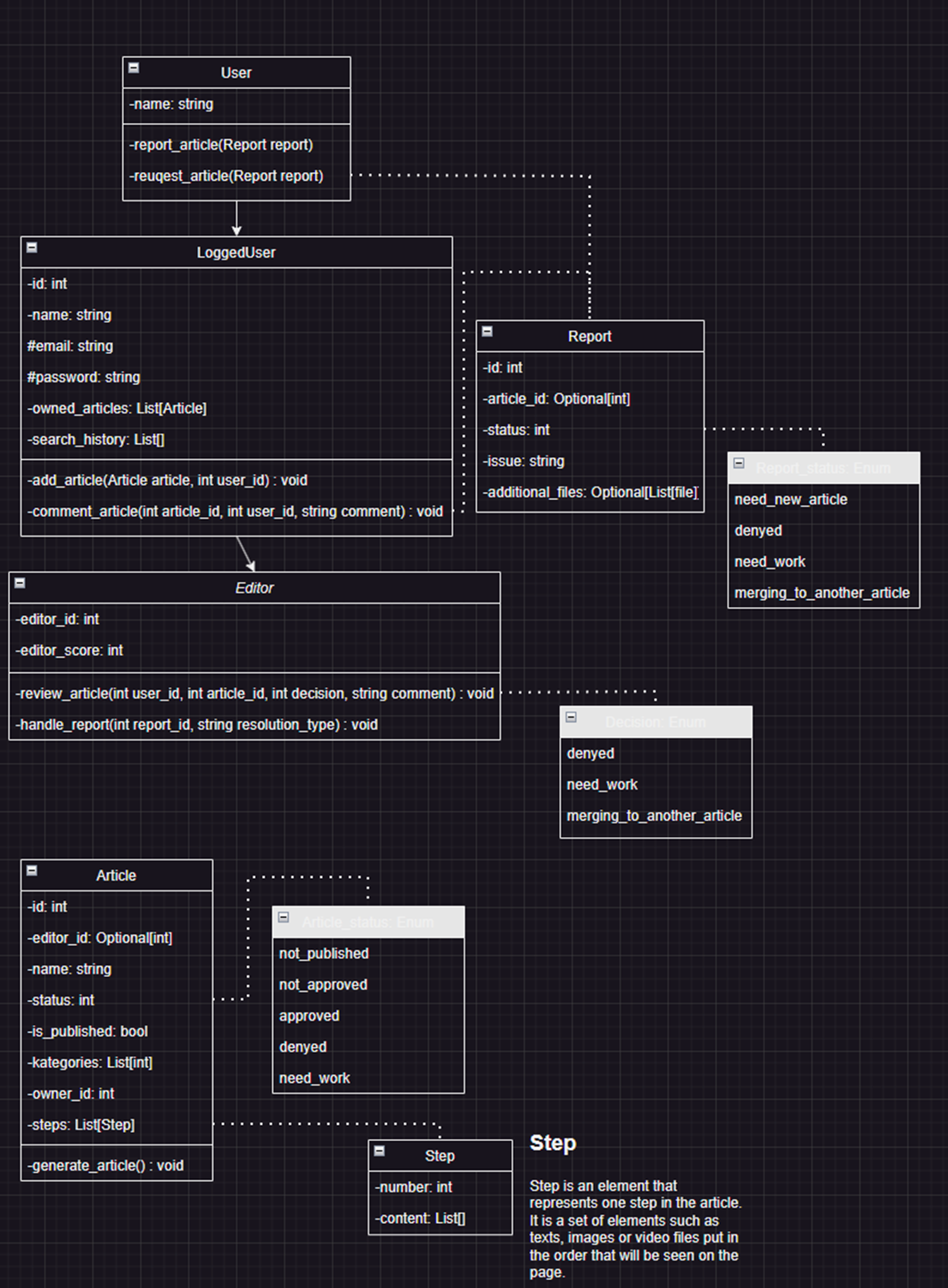
Metodyka XPrince realizuje wszystkie 4 elementy manifestu, kładąc szczególny nacisk na współpracę z klientem.

W podejściu iteracyjnym, tworzenie oprogramowania jest podzielone na serię iteracji, z których każda ma określony cel i dostarcza określone rezultaty. Każda iteracja w metodyce XPrince składa się z analizy wymagań, projektowania, implementacji, testowania i wdrażania. W miarę postępu projektu, kolejne iteracje mogą dostarczać rozwiniętych wersji systemu, wzbogacając go o nowe funkcje lub poprawiając istniejące. Dzięki temu podejściu, system może być sukcesywnie rozwijany i dostosowywany na podstawie bieżących potrzeb użytkowników, którzy mają możliwość korzystania z wypracowanych podczas iteracji elementów systemu.

Podejście zwinne i iteracyjne przyczyniają się do skrócenia cyklu wytwarzania oprogramowania, zwiększenia zaangażowania użytkowników i umożliwienia szybkiej reakcji na zmieniające się wymagania. Pozwala to na bardziej efektywne dostarczanie wartościowych rozwiązań, minimalizując ryzyko projektowe i zwiększając szanse na sukces projektu. Korzyści te są kluczowe dla tego projektu, ponieważ współpraca z docelowym użytkownikiem wraz z elastycznością projektu dadzą przestrzeń na stworzenie wydajnego i adekwatnego systemu.

1. Projekt aplikacji
   1. Diagram klas

Poniższy rysunek zawiera diagram klas UML, zawierający informacje o tym jakie obiekty będą zawierały się w systemie oraz jakie są dodatkowe zaimplementowane mechanizmy służące optymalizacji kodu i przepływu danych.



1. Diagram klas
   1. Architektura

W ramach projektu zastosowana zostanie architektura klient-serwer. Jest to powszechnie stosowany model architektoniczny, w którym system jest podzielony na dwie główne części: klienta i serwera. Klient jest odpowiedzialny za interakcję użytkownika z systemem, natomiast serwer zarządza przetwarzaniem danych i udostępnia usługi dla klienta.

Uzasadnienie decyzji:

Skalowalność: Architektura klient-serwer umożliwia łatwe skalowanie systemu wraz z rozwojem. Serwer może być dostosowany do obsługi większej liczby klientów, a także zwiększenia obciążenia i rozszerzenia funkcjonalności.

Wydajność: Dzięki architekturze klient-serwer możliwe jest optymalne wykorzystanie zasobów, takich jak pamięć, procesor czy sieć. Serwer może być zoptymalizowany pod kątem wydajności, co przekłada się na szybsze przetwarzanie żądań klienta.

Bezpieczeństwo: Architektura klient-serwer zapewnia odpowiednie mechanizmy bezpieczeństwa, takie jak uwierzytelnianie, autoryzacja czy szyfrowanie danych. Dostęp do zasobów serwera może być kontrolowany, co zwiększa bezpieczeństwo systemu.

Dostępność: Architektura klient-serwer może być zaprojektowana w taki sposób, aby zapewnić wysoką dostępność systemu oraz dostosowania do wielu platform. Możliwość redundancji i rozproszenia serwerów umożliwia minimalizację czasu przestoju i zapewnienie nieprzerwanej pracy.

Zgodność z wymaganiami biznesowymi: Architektura klient-serwer może być dostosowana do celów i strategii biznesowej. Umożliwia łatwiejsze dostosowanie systemu do wymagań biznesowych.

Technologie i narzędzia: Architektura klient-serwer jest kompatybilna z wieloma technologiami i narzędziami, co umożliwia wybór odpowiednich rozwiązań do implementacji systemu.

Schemat architektury przedstawiony jest na poniższym rysunku:

Obraz zawierający tekst, zrzut ekranu, Czcionka, diagram

Opis wygenerowany automatycznie

1. Schemat architektury klient-serwer systemu
   1. Projekt modułu klienta

Moduł klienta w ramach architektury klient-serwer jest odpowiedzialny za interakcję użytkownika z systemem. W przypadku opisywanego serwisu, moduł klienta będzie dostarczał interfejs użytkownika, który umożliwi tworzenie i przeglądanie wpisów z poradami dotyczącymi podstawowych narzędzi informatycznych. Moduł ten będzie charakteryzować się następującymi cechami:

Intuicyjny interfejs: Interfejs użytkownika będzie zaprojektowany w taki sposób, aby był łatwy w obsłudze dla użytkowników o różnym poziomie zaawansowania. Będzie uwzględniał potrzeby i oczekiwania użytkowników, umożliwiając im szybki dostęp do funkcji serwisu.

Personalizacja: Moduł klienta będzie oferował możliwość personalizacji doświadczenia użytkownika. Użytkownicy będą mogli dostosować preferencje, np. wybierając preferowane narzędzia, otrzymując rekomendacje dostosowane do ich umiejętności, a także śledząc interesujące ich wątki czy kategorie.

Wsparcie dla różnych poziomów zaawansowania: Moduł klienta będzie zapewniał porady i wskazówki dostosowane do poziomu obeznania użytkownika z komputerem. Będzie umożliwiał wybór porad dla początkujących, zaawansowanych czy ekspertów, aby zapewnić optymalne wsparcie dla każdego użytkownika.

Wizualizacja treści: Interfejs klienta będzie umożliwiał czytelne i atrakcyjne przedstawienie porad, zastosowanie odpowiednich grafik, diagramów czy filmów instruktażowych, aby ułatwić użytkownikom zrozumienie i wdrożenie omawianych zagadnień.

* 1. Projekt modułu serwera

Moduł serwera w architekturze klient-serwer jest kluczowym elementem systemu. W przypadku opisanego serwisu, moduł serwera będzie odpowiedzialny za zarządzanie przetwarzaniem danych oraz udostępnianie usług dla klienta. Poniżej przedstawiam charakterystykę modułu serwera:

Przetwarzanie danych: Moduł serwera będzie gromadził, przetwarzał i przechowywał wpisy z poradami użytkowników. Będzie zarządzał bazą danych, umożliwiając efektywne wyszukiwanie, dodawanie, modyfikację i usuwanie wpisów.

Analiza SWOT: Moduł serwera będzie reagował na wyniki analizy SWOT, dostarczając funkcjonalności, które wspierają mocne strony i szanse oraz odpowiadają na potrzeby wynikające ze słabych stron i zagrożeń. Na podstawie tych wyników, serwer będzie dostarczał użytkownikom spersonalizowane porady i rekomendacje.

Skalowalność i wydajność: Moduł serwera powinien być zaprojektowany w taki sposób, aby umożliwiał łatwe skalowanie systemu wraz z rozwojem. Będzie zoptymalizowany pod kątem wydajności, aby obsłużyć duże obciążenie i zapewnić szybką odpowiedź na żądania użytkowników.

Bezpieczeństwo i poufność danych: Moduł serwera będzie implementował odpowiednie mechanizmy bezpieczeństwa, takie jak uwierzytelnianie, autoryzacja i szyfrowanie danych. Będzie chronił poufność informacji użytkowników oraz zapewniał integralność i dostępność systemu.

Przechowywanie i zarządzanie poradami: Moduł serwera będzie odpowiedzialny za przechowywanie porad w odpowiedniej strukturze danych. Będzie umożliwiał kategoryzację, tagowanie i indeksowanie wpisów, aby użytkownicy mogli łatwo przeszukiwać i odnaleźć interesujące ich treści.

* 1. Projekt bazy danych

W ramach projektu zastosowana będzie nierelacyjna baza danych. Jej najważniejszymi obiektami będą użytkownik oraz wpis, przechowujące wszystkie potrzebne informacje o tych elementach. Poniżej rozpisane są schematy dla obiektów użytkowników oraz wpisów.

Kolekcja „users” (do użytkowników)

* \_id: unikalny identyfikator użytkownika (klucz główny)
* name: nazwa użytkownika
* email: adres e-mail użytkownika
* password: hasło użytkownika
* created\_at: data utworzenia konta użytkownika
* role: rola użytkownika w systemie (np. administrator, użytkownik, redaktor)
* search\_history: lista ostatnich przeglądanych wpisów

Kolekcja „articles” (do wpisów):

* \_id: unikalny identyfikator wpisu (klucz główny)
* title: tytuł wpisu
* content: zawartość wpisu (elastyczne pole, które może zawierać różne formaty danych, np. tekst, obrazy, multimedia)
* author\_id: identyfikator autora wpisu (klucz obcy do kolekcji "users")
* created\_at: data utworzenia wpisu
* tags: lista tagów przypisanych do wpisu

Rozwinięcie atrybutu content:

Jest to najbardziej kluczowy element bazy danych, ponieważ przechowywać on będzie wszystkie informacje tym co wpis zawiera oraz w jakiej kolejności. Będzie to lista różnych elementów (tekstu, obrazów, …), które będą w takiej kolejności, jak zapisane, generowane na stronie.

Poniższy rysunek przedstawia schematy kolekcji wraz z zaznaczeniem zależności między nimi (ze względu na rodzaj bazy danych nie będzie to relacja, natomiast artykuły będą pobierały id użytkownika będącego autorem).

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

1. Schemat bazy danych

Przykładowe zapytania do wykonywania podstawowych czynności w bazie danych:

* Dodawanie nowego użytkownika: db.users.insertOne({name: "John Doe", email: "john@example.com", password: "hashed\_password", created\_at: ISODate("2023-06-13"), role: "user”})
* Dodawanie nowego wpisu: db.posts.insertOne({title: "Przykładowy wpis", content: "Lorem ipsum dolor sit amet...", author\_id: ObjectId("user\_id"), created\_at: ISODate("2023-06-13"), tags: ["tag1", "tag2"]})
* Pobieranie wszystkich wpisów danego użytkownika: db.posts.find({author\_id: ObjectId("user\_id")})
* Pobieranie wpisów zawierających określony tag: db.posts.find({tags: "tag1”})
  1. Projekt interfejsu

Ze względu na grupę docelową projektu interfejs graficzny musi być prosty w obsłudze oraz intuicyjny. Projekt uwzględniać musi zarówno aspekt estetyki i ładnego wyglądu aplikacji, jak i wszelkie potrzeby co do funkcjonalności. Poniższe rysunki przedstawiają ramowy układ stron dla projektu interfejsu dla różnych okien dialogowych.

Obraz zawierający tekst, zrzut ekranu, wyświetlacz

Opis wygenerowany automatycznie

1. Szablon dla wszystkich stron interfejsu

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

1. Wygląd strony głównej interfejsu

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

1. Wygląd strony wpisu interfejsu

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

1. Wygląd kreatora wpisów interfejsu

Podsumowanie

Celem ćwiczenia było wykorzystanie dotychczas zebranej wiedzy z dziedziny inżynierii systemów informatycznych do stworzenia dokumentu opisującego realizację projektu na wybrany temat. Utworzenie tego sprawozdania realizuje ten cel. Tworząc diagramy, specyfikację systemu, dokumentację jego właściwości można było w sposób dokładny i holistyczny zastosować zebraną wiedzę z zajęć i zaprezentować umiejętności. Ćwiczenie nie obejmowało stworzenia całego systemu, a jedynie jego opisania i stworzenia planu jego wytwarzania. Poprzez odwołanie się do metod, technik, narzędzi oraz innych istotnych dla inżynierii systemu zagadnień udało się wybrać i opisać zastosowanie wybranej przeze mnie metodyki wytwarzania oprogramowania jaką jest metodyka XPrince.

Dokumentacja zawiera wszystkie niezbędne informacje dotyczące tego jaki powinien być system oraz co powinien zawierać, aby spełniał on swoje przeznaczenie. Poprzez specyfikację wymagań można zweryfikować, kiedy system spełnia swoje minimum, a kiedy jest w pełni gotowy. Opisane jest jaką należy zastosować architekturę, jak przechowywać dane oraz jakie uprawnienia powinni posiadać konkretni użytkownicy systemu. Dzięki temu w momencie realizacji prac nad tworzeniem oprogramowania systemu można uniknąć niejasności w zakresie kształtu i tego, jak należy realizować elementy systemu.

Podsumowując, realizacja ćwiczenia, a za razem tworzenie sprawozdania było wartościową okazją do przetestowania oraz zaprezentowania umiejętności z zakresu inżynierii systemów. W wyniku prac powstała rozbudowana dokumentacja umożliwiająca utworzenie jakościowego systemu.

Bibliografia

Organizacji ZHP:

1. <https://pl.wikipedia.org/wiki/Zwi%C4%85zek_Harcerstwa_Polskiego>
2. <https://zhp.pl/>

O analizie biznesowej:

1. <https://www.jcommerce.pl/jpro/artykuly/7-grzechow-glownych-w-analizie-biznesowej>

O specyfikacji wymagań:

1. <https://pl.wikipedia.org/wiki/Wymaganie_(in%C5%BCynieria)>
2. <https://visuresolutions.com/pl/blog/functional-requirements/>
3. <https://wazniak.mimuw.edu.pl/images/6/62/Zio-9-wyk.pdf>

O diagramach (use case, UML, BPMN)

1. <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>
2. <http://www.maciejczak.pl/download/pproc-c2.pdf>
3. <https://www.visual-paradigm.com/guide/bpmn/how-to-use-data-objects-in-bpmn/>

O metodyce XPrince

1. <https://pl.wikipedia.org/wiki/XPrince>
2. <https://mfiles.pl/pl/index.php/Metodyka_XPrince>
3. <https://teamquest.pl/baza-wiedzy/project-management/xprince/137.html>

O architekturze klient-serwer

1. <https://pl.wikipedia.org/wiki/Klient-serwer>
2. <https://vavatech.pl/technologie/architektura/Klient-Server>
3. <https://esezam.okno.pw.edu.pl/mod/book/view.php?id=8&chapterid=56>

Spis rysunków

[Rys. 1. Logo Związku Harcerstwa Polskiego 6](#_Toc137989433)

[Rys. 2. Struktura Związku Harcerstwa Polskiego 6](#_Toc137989434)

[Rys. 3. Diagram przypadków użycia systemu 13](#_Toc137989435)

[Rys. 4. Diagram klas 25](#_Toc137989436)

[Rys. 5. Schemat architektury klient-server ssytemu 26](#_Toc137989437)

[Rys. 6. Schemat bazy danych 28](#_Toc137989438)

[Rys. 7. Szablon dla wszystkich stron interfejsu 29](#_Toc137989439)

[Rys. 8. Wygląd strony głównej interfejsu 30](#_Toc137989440)

[Rys. 9. Wygląd strony wpisu interfejsu 30](#_Toc137989441)

[Rys. 10. Wygląd kreatora wpisów interfejsu 31](#_Toc137989442)

Spis tabel

[Tab. 1. Analiza SWOT wpływu rozwiązań informatycznych na działania w ZHP 8](#_Toc137989443)

[Tab. 2. Spis wymagań funkcjonalnych 9](#_Toc137989444)

[Tab. 3. Spis wymagań pozafunkcjonalnych 11](#_Toc137989445)

[Tab. 4. Spis wymagań systemowych 12](#_Toc137989446)

[Tab. 5. Opis przypadków użycia z diagramu przypadków użycia UML 13](#_Toc137989447)

[Tab. 6. Opis aktorów z diagramu przypadków użycia UML 15](#_Toc137989448)

[Tab. 7. Etap 1 Rozpoczęcie projektu, metodyka XPrince 18](#_Toc137989449)

[Tab. 8. Etap 2 Inicjacja, metodyka XPrince 19](#_Toc137989450)

[Tab. 9. Etap 3 Elaboracja, metodyka XPrince 20](#_Toc137989451)

[Tab. 10. Etap 4 Wydanie, metodyka XPrince 21](#_Toc137989452)

[Tab. 11. Etap 5 Zamknięcie projektu, metodyka XPrince 21](#_Toc137989453)