

Imię i nazwisko studenta: Zofia Zinkowska

Nr albumu: 180925

Poziom kształcenia: Studia pierwszego stopnia

Forma studiów: stacjonarne

Kierunek studiów: Fizyka Techniczna

Specjalność: Informatyka Stosowana

**PRACA DYPLOMOWA INŻYNIERSKA**

Tytuł pracy w języku polskim: Aplikacja „ewidencja roślin”

Tytuł pracy w języku angielskim: The „plant register” application

Opiekun pracy: dr. inż. Bartosz Reichel



**OŚWIADCZENIE: dotyczące pracy dyplomowej zatytułowanej:**

**Aplikacja „ewidencja roślin”**

Imię i nazwisko: Zofia Zinkowska

Data i miejsce urodzenia: 20.07.2000, Słupsk

Nr albumu: 180925

Wydział: Wydział Fizyki Technicznej i Matematyki Stosowanej

Kierunek: fizyka techniczna

Poziom kształcenia: pierwszy

Forma studiów: stacjonarnie

Typ pracy: praca dyplomowa inżynierska

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2019 r. poz. 1231, z późn. zm.) i konsekwencji dyscyplinarnych określonych w ustawie z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (t.j. Dz. U. z 2020 r. poz. 85, z późn. zm.), 1. a także odpowiedzialności cywilnoprawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.

Niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. pracy dyplomowej uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

29.03.2022, Zofia Zinkowska

Data i podpis lub uwierzytelnienie w portalu uczelnianym Moja PG

\*) Dokument został sporządzony w systemie teleinformatycznym, na podstawie §15 ust. 3b Rozporządzenia MNiSW z dnia 12 maja 2020 r. zmieniającego rozporządzenie w sprawie studiów (Dz.U. z 2020 r. poz. 853). Nie wymaga podpisu ani stempla

**STRESZCZENIE**

Praca dyplomowa przedstawia aplikację internetową, służącą do utrzymania / ewidencji roślin w ogrodzie botanicznym. W pierwszej oraz drugiej części pracy uwaga poświęcona jest wymaganiom jakim oprogramowanie musi sprostać. Trzecia część opisuje technologie oraz narzędzia programistyczne wykorzystane do implementacji. Kluczowym miejscem jest rozdział czwarty, który przedstawia projekt tworzonego oprogramowania. Końcowe rozdziały przedstawiają testy, podsumowanie, spis rysunków oraz bibliografię.

**Słowa kluczowe:**

Utrzymanie ewidencji roślin, ewidencja roślin, aplikacja internetowa, C#, TypeScript, ASP.NET Core, Vue.js.

**Dziedzina nauki i techniki z wymogami OECD:**

**ABSTRACT**

The thesis presents a web application for maintaining / registering plants in a botanical garden. In the first and second part of the work, the focus was on the requirements that the software should handle. The third part describes the technologies used for the implementation. The most significant part of the thesis is the fourth chapter, which presents the design of the software being developed. The final chapters present the tests, summary, list of figures, and a bibliography.

**Keywords:**

Plants maintennance registration, plants registration, web application, C#, TypeScript, ASP.NET Core, Vue.js.

Spis treści

[WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW 5](#_Toc120054323)

[1. Wstęp 5](#_Toc120054324)

[1.1. Aplikacja internetowa 5](#_Toc120054325)

[1.2. Cel projektowania aplikacji 5](#_Toc120054326)

[2. Specyfikacja wymagań 7](#_Toc120054327)

[2.1. Dziedzinowy słownik pojęć 7](#_Toc120054328)

[2.2. Interfejs graficzny aplikacji 7](#_Toc120054329)

[2.3. Wymagania funkcjonalne użytkownika 10](#_Toc120054330)

[2.4. Wymagania niefunkcjonalne użytkownika 10](#_Toc120054331)

[2.5. Wymagania niefunkcjonalne systemowe 10](#_Toc120054332)

[3. Przygotowanie do tworzenia aplikacji 11](#_Toc120054333)

[3.1. Wykorzystane narzędzia programistyczne 11](#_Toc120054334)

[3.1.1. Git 11](#_Toc120054335)

[3.1.2. Bootstrap 11](#_Toc120054336)

[3.1.3. Baza danych LiteDB 11](#_Toc120054337)

[3.1.4. ASP .NET Core 12](#_Toc120054338)

[3.1.5. Visual Studio 12](#_Toc120054339)

[3.1.6. Visual Studio Code 12](#_Toc120054340)

[3.1.7 Cypress 12](#_Toc120054341)

[3.1.8. Użyte biblioteki 13](#_Toc120054342)

[4. Projekt aplikacji - implementacja 14](#_Toc120054343)

[4.1. Diagram 16](#_Toc120054344)

[4.1.1 Diagramy maszyny stanowej 16](#_Toc120054345)

[4.2. Wyjątki 17](#_Toc120054346)

[4.3. Adresowanie 18](#_Toc120054347)

[5. Testy 19](#_Toc120054348)

[6. Podsumowanie 20](#_Toc120054349)

[7. Spis rysunków 22](#_Toc120054350)

# WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

# 1. Wstęp

Ze względu na rosnące zapotrzebowanie jakościowe, konieczny jest ciągły rozwój. Dlatego pojawiają się nowe rozwiązania w zakresie utrzymania / ewidencji roślin ułatwiające codzienną pracę.

## 1.1. Aplikacja internetowa

Aplikacja internetowa jest to program komputerowy pracujący na serwerze. Komunikuje się poprzez sieć komputerową z hostem użytkownika komputera, przy użyciu przeglądarki internetowej, będącej interaktywnym klientem aplikacji.

Aplikacje internetowe są jednym z najpopularniejszych typów programów komputerowych. Służą nie tylko do rozrywki, ale również jako systemy obsługi klientów. Aplikacje internetowe towarzyszą nam codziennie, od sterowania żarówką (smart home) do załatwiania spraw urzędowych (epuap).

Podstawową zaletą tworzenia oprogramowania tego typu jest fakt, że aplikacje internetowe są kompatybilne wieloplatformowo, gdyż nie zależą od systemu operacyjnego, a jedynym wymaganiem do sprawnego korzystania z programu jest posiadanie zainstalowanej przeglądarki internetowej z dostępem do internetu. Następną zaletą jest to, że użytkownik posiada dostęp do swoich danych nawet w momencie, kiedy następuje awaria urządzenia, ponieważ dane są gromadzone na serwerach dostawcy oprogramowania. Aplikacje webowe nie wymagają instalacji oraz aktualizacji (wdrażania i utrzymania) po stronie klienta, ponieważ zawsze serwowane są w najnowszej wersji serwera. Nie potrzebują weryfikacji / certyfikacji od poszczególnych producentów (np. Google Store, Apple Store).

Aplikacje internetowe oprócz zalet posiadają również wady. Jedną z nich jest całkowita zależność od serwera przez co użytkownik nie ma możliwości dokonywania operacji bez połączenia z siecią. Oprócz tego dostawca oprogramowania jest w stanie śledzić niemal każdy ruch użytkownika podczas jego korzystania z aplikacji. W dzisiejszych czasach mało, które aplikacje, nawet aplikacje nie internetowe, potrafią działać bez dostępu do internetu (np. aplikacja desktopowa nadal składowałaby dane na centralnym serwerze). W takiej architekturze utrata urządzenia byłaby jednoznaczna z utratą wszystkich danych.

## 1.2. Cel projektowania aplikacji

Na co dzień aplikacje internetowe ułatwiają nam zarządzanie pocztą, pozwalają robić zakupy i opłacać rachunki. Aplikacja ma na celu rozwiązanie problemu przestarzałego oprogramowania służącego do utrzymania / ewidencji roślin w ogrodzie botanicznym. Przedsięwzięcie dąży do wypełnienia brakujących potrzeb w systemie. Dzięki możliwości skanowania kodów etykiet oraz drukowaniu etykiet użytkownik będzie wiedział dokładnie jakie rośliny znajdują się w ogrodzie botanicznym. Nie tylko usprawni to pracę użytkownikowi, jak również użytkownik zyska na czasie, ponieważ automatyzacja procesów wcześniej wykonywanych w sposób manualny, w znaczny sposób wykorzystywała czas osób administrujących.

W osiągnięciu celu pomogą nowe technologie programistyczne tj. C# - ASP.NET Core, TypeScript, Vue.js.

# 2. Specyfikacja wymagań

Rozdział przedstawia wymagania stawiane aplikacji internetowej służącej do utrzymania / ewidencji roślin. W rozdziale omawiane są kolejno wymagania biznesowe, funkcjonalne użytkownika oraz wymagania niefunkcjonalne użytkownika i systemowe.

## 2.1. Dziedzinowy słownik pojęć

Użytkownik systemu – osoba korzystająca z aplikacji do utrzymania / ewidencji roślin.

CSS - Cascading Style Sheets - Kaskadowe Arkusze Stylów.

## 2.2. Interfejs graficzny aplikacji

Interfejs graficzny został wykonany przy użyciu biblioteki Bootstrap-Vue 3. Używane są również style kaskadowe CSS.

#### Obraz zawierający tekst Opis wygenerowany automatycznie

Rysunek . Interfejs graficzny – strona główna

#### Graphical user interface Description automatically generated

Rysunek . Interfejs graficzny – ewidencja roślin

1. Lista roślin w ewidencji.
2. Przycisk otwierający stronę dodawania roślin do ewidencji.
3. Wyszukiwanie.
4. Przycisk otwierający stronę do edycji informacji o roślinie.
5. Przycisk usuwający daną pozycję z ewidencji.
6. Sortowanie nazw i dat dodania rosnąco / malejąco.

#### Graphical user interface, application Description automatically generated

Rysunek . Interfejs graficzny – formularz edycji pól z ewidencji roślin

1. Pole tekstowe z numerem identyfikacyjnym rośliny.
2. Pole z datą dodania rośliny do ewidencji.
3. Pole tekstowe do wprowadzenia nazwy rośliny.
4. Przycisk anulujący edycję.
5. Przycisk wysyłający dane z formularza do serwera.

#### Obraz zawierający tekst Opis wygenerowany automatycznie

Rysunek . Interfejs graficzny – ochrona roślin

#### Obraz zawierający tekst Opis wygenerowany automatycznie

Rysunek . Interfejs graficzny – dodawanie nowego wpisu do ewidencji roślin

Aby móc dodać roślinę do ewidencji roślin, należy wejść w ewidencję i wybrać dodaj.

1. Pole tekstowe do wyszukiwania pozycji w klasyfikacji roślin z podpowiadaniem od 3 pierwszych liter.
2. Przycisk anulujący dodawanie.
3. Przycisk zapisujący dane do ewidencji roślin.

## 2.3. Wymagania funkcjonalne użytkownika

Aplikacja powinna posiadać takie funkcjonalności jak:

* Dodawanie, edycja, usuwanie wpisów.
* Listowanie wpisów wraz z wyszukiwaniem i sortowaniem.
* Podpowiadanie podczas wyszukiwania.
* Możliwość dodawania i skanowania za pomocą kodów QR.
* Możliwość drukowania etykiet.

## 2.4. Wymagania niefunkcjonalne użytkownika

Aplikacja powinna odpowiadać aktualnym standardom sieciowym i wytycznym dla dostępności treści internetowych – ma prawidłowo wyświetlać się w różnych przeglądarkach:

* Google Chrome
* Microsoft Edge
* Mozilla FireFox

Obsługa dużego pliku z klasyfikacją roślin.

## 2.5. Wymagania niefunkcjonalne systemowe

* Aplikacja powinna korzystać z dostępnych darmowych rozwiązań systemów informatycznych.
* Aplikacja powinna być wieloplatformowa.
* Interfejs aplikacji – system powinien być łatwy w obsłudze poprzez przejrzysty wygląd interfejsu, zrozumiały dla niedoświadczonego użytkownika. Aplikacja powinna być zaopatrzona w regulamin usługi, zrozumiałe menu oraz przyciski na stronie.
* Niezawodność aplikacji – przeprowadzanie testów poszczególnych modułów aplikacji. Optymalne dobranie architektury sprzętowo-programowej na jakiej będzie działać aplikacja oraz serwer bazodanowy.

# 3. Przygotowanie do tworzenia aplikacji

Poniżej zostały przedstawione użyte technologie podczas implementacji aplikacji. Przy wyborze technologii autorce zależało na wykorzystaniu prostych i szybkich rozwiązań, darmowych oraz ogólnodostępnych. Dzięki odpowiedniemu doborze narzędzi rozwiązanie ewentualnych problemów było szybsze.

## 3.1. Wykorzystane narzędzia programistyczne

W tym rozdziale zostały przedstawione wykorzystane narzędzia programistyczne, które były wykorzystane podczas tworzenia projektu.

### 3.1.1. Git

Git jest rozproszonym systemem kontroli wersji. Został stworzony przez Linusa Torvaldsa w 2005 roku.

System z założenia miał podtrzymywać wielkie rozproszone projekty programistyczne, takie jak np. Linux.

Najważniejsze cechy:

* Programowanie rozproszone – system dostarcza programiście pracującemu nad projektem lokalną kopię rozwoju aplikacji.
* Zgodność z protokołami i systemami – repozytoria mogą być publikowane za pomocą HTTP, FTP, etc.
* Programowanie nieliniowe – wsparcie dla nieliniowego wytwarzania oprogramowania

### 3.1.2. Bootstrap

Bootstrap – biblioteka użyta w projekcie. Zawiera zestaw przydatnych narzędzi ułatwiających tworzenie interfejsu graficznego stron oraz aplikacji internetowych. Oferuje domyślny zestaw tematyczny obejmujący kolorystykę i stylistykę wyglądu strony, jednocześnie pozwalając na łatwą modyfikację tego zestawu tematycznego, bez ingerencji w kod poszczególnych elementów. [1] [2] [3]

W swojej aplikacji użyłam wersji Bootstrap 5.2.0.

### 3.1.3. Baza danych LiteDB

LiteDB to szybka, prosta, typu open source z zerową konfiguracją, wbudowana baza danych NoSQL dla .NET w pełni napisana w języku C#. LiteDB organizuje dokumenty w magazynach dokumentów znanych jako kolekcje. Co oznacza, że każda kolekcja jest identyfikowana przez unikatową nazwę i zawiera jeden lub więcej dokumentów, które mają ten sam schemat. Ponieważ LiteDB jest bazą danych bez serwera, nie musimy instalować jej w swoim systemie. Wystarczy dodać odwołanie do pliku LiteDB.dll w swoim projekcie. Alternatywnie można zainstalować LiteDB za pomocą Menadżera pakietów NuGet w programie Visual Studio. [4]

W swojej aplikacji użyłam wersji LiteDB 5.0.12.

### 3.1.4. ASP .NET Core

ASP .NET Core (w swojej aplikacji użyłam wersji ASP .NET Core 6.0) jest to platforma o otwartym kodzie (Open Source) do tworzenia aplikacji internetowych, stworzona przez firmę Microsoft. Umożliwia tworzenie zarówno aplikacji typowo backendowych (Web API), jak i w pełni funkcjonalnych aplikacji internetowych (backend + frontend). W pracy użyłam tylko funkcjonalności Web API, ze względu na architekturę Single Page Application (SPA), gdzie część frontendowa jest napisana we frameworku Vue.js [5] [6]

### 3.1.5. Visual Studio

Visual Studio jest to bogate w funkcje zintegrowane środowisko programistyczne (IDE), stworzone przez firmę Microsoft. Środowisko to jest dedykowanym rozwiązaniem używanym przy tworzeniu aplikacji ASP .NET Core, w którym została stworzona część backendowa aplikacji. Dodatkowo natywnie współpracuje z systemem kontrolowania wersji Git.

W swojej aplikacji użyłam wersji Microsoft Visual Studio Community 2022 (64-bit) 17.3.4.

### 3.1.6. Visual Studio Code

Visual Studio Code to prosty edytor kodu źródłowego, stworzony przez firmę Microsoft. Program jest bogaty w rozszerzenia dla różnych języków programowania i platform programistycznych, z czego większość jest darmowa i posiada otwarty kod (Open Source). Między innymi posiada wtyczki dla użytych w aplikacji technologii i języków takich jak Vue.js, Node.js, TypeScript, Bootstrap. Natywnie integruje się z systemem wersjonowania Git, co znacząco ułatwia tworzenie oprogramowania przy użyciu tego narzędzia.

W swojej aplikacji użyłam wersji Visual Studio Code 1.72.2.

### Cypress

Cypress jest to narzędzie do tworzenia i uruchamiania testów automatycznych typu end-to-end oparty na platformie Node.js. Jest to zestaw mniejszych narzędzi, które pomogą uruchomić test, scenariusz testowy, w przeglądarce (narzędzie wspiera wiele przeglądarek, np. Edge, Chrome, FireFox ) oraz za pomocą specjalnego API, które Cypress wystawia. Cały kod testów piszemy w języku JavaScript lub TypeScript. Z narzędzia możemy korzystać praktycznie od razu po instalacji, nie wymaga praktycznie żadnej konfiguracji. [7]

W swojej aplikacji użyłam wersji Cypress 10.11.0.

### 3.1.8. Użyte biblioteki

* Axios to prosty klient HTTP oparty na obietnicy przeglądarki i node.js Axios zapewnia prostą w użyciu bibliotekę w małym pakiecie z bardzo rozszerzalnym interfejsem.
* Vue3-simple-typeahead prosty i lekki komponent Vue3 do wpisywania / autouzupełniania, który wyświetla sugerowaną listę elementów podczas wpisywania przez użytkownika.
* Moment.js biblioteka ułatwiająca pracę z datami i godzinami w JavaScript / TypeScript, między innymi formatowanie czy obsługę stref czasowych. [8]
* Qrcode.vue biblioteka umożliwiająca generowanie kodów QR. [9]
* Vue3-print-nb biblioteka umożliwiająca drukowanie strony HTML.

# 4. Projekt aplikacji - implementacja

Interfejs graficzny aplikacji jest wykonany przy użyciu biblioteki Bootstrap.

#### Text Description automatically generated

Rysunek . Fragment pliku Startup.cs

Na Rysunku 6. zostało przedstawione dodanie wszystkich serwisów do kontenera DI (ang. dependency injection). Ten fragment kodu definiuje cykl życia poszczególnych serwisów: singleton (jedna instancja na całą aplikację), scoped (jedna instancja na jedno zapytanie http).

#### Text Description automatically generated

Rysunek 7. Fragment pliku Services/TaxonomyProvider.cs

Na Rysunku 7. przedstawiono fragment kodu, w którym wczytujemy plik classification.txt zawierający klasyfikację roślin zgromadzoną w kompendium World Flora Online [10]. Plik jest tak obszerny, że w celu optymalizacji plik jest wczytywany tylko raz na początku uruchomienia aplikacji (cykl życia singleton).

#### Text Description automatically generated

Rysunek . Plik Services/RegisterRepository.cs

Na Rysunku 8. przedstawiono implementację serwisu RegisterRepository.cs. Serwis ten służy do ukrycia szczegółów implementacyjnych związanych z bazą LiteDB poprzez wyabstrahowanie metod CRUD. Jest to typowy przykład zastosowania wzorca projektowego „repozytorium”.

## 4.1. Diagram

### 4.1.1 Diagramy maszyny stanowej

#### 

Rysunek . Dodanie rośliny do ewidencji - Diagram maszyny stanowej



Rysunek . Kontrolka do wyszukiwania klasyfikacji - Diagram maszyny stanowej

## 4.2. Wyjątki

Nazwa aplikacji to Ewidencja Roślin. Oprogramowanie ma za zadanie ułatwić utrzymanie / ewidencję roślin. W rozdziale omawiane są wymagania biznesowe, wymagania funkcjonalne i niefunkcjonalne – użytkownika i systemowe.

## 4.3. Adresowanie

Nazwa aplikacji to Ewidencja Roślin. Oprogramowanie ma za zadanie ułatwić utrzymanie / ewidencję roślin. W rozdziale omawiane są wymagania biznesowe, wymagania funkcjonalne i niefunkcjonalne – użytkownika i systemowe.

# 5. Testy

Aby aplikacja mogła trafić w ręce użytkowników, powinna być dobrze przetestowana. Osobami odpowiedzialnymi za napisanie przypadków testowych oraz ich przeprowadzenie są testerzy oprogramowania. Przypadek testowy powinien posiadać zdefiniowany cel, określone warunki wstępne oraz kroki, na kroki testu składa się opis oczekiwanego rezultatu i poszczególnych kroków.

Testy możemy podzielić na kilka rodzajów. Podstawowe testy to testy jednostkowe oraz integracyjne, możemy również wyróżnić testy systemowe oraz akceptacyjne. Testy powinny sprawdzać jak najwięcej możliwych przypadków. Jeżeli wszystkie testy przechodzą pozytywnie, możemy mieć pewność, że dana funkcjonalność działa poprawnie. Jeśli jednak w aplikacji pojawi się błąd, z łatwością możemy znaleźć źródło problemu.

W projekcie przetestowano aplikację do ewidencjonowania roślin botanicznych w narzędziu Cypress, użyte funkcje oraz testy znajdują się w plikach folderu cypress.

Do aplikacji internetowej stworzyłam testy automatyczne typu end-to-end, dzięki którym sprawdziłam poprawność działania aplikacji.

Testy end-to-end są bardziej kosztowne, trudniejsze w utrzymaniu ponieważ muszą wykonać kilka kroków, aby osiągnąć ten stan aplikacji, który testuje. Dzięki temu, że test będzie działał w przeglądarce będziemy mogli sprawdzić wszystkie warstwy aplikacji, zaczynając od bazy danych przez backend, aż do frontendu.

# 6. Podsumowanie

Celem pracy inżynierskiej, było nie tylko stworzenie wymagań, zaprojektowanie i zaimplementowanie aplikacji, służącej do ewidencjonowania roślin botanicznych, ale również chęć podjęcia pracy w technologii ASP .NET Core.

Przed podjęciem pracy nad aplikacją przeanalizowano dostępne rozwiązania oraz zwrócono szczególną uwagę na wady i zalety funkcjonalności oraz interfejsów.

Dzięki implementacji aplikacji internetowej mogłam poznać wiele nowych technologii. Doświadczenie, które zdobyłam będę mogła wykorzystać w kolejnych projektach w trakcie mojej dalszej edukacji oraz kariery zawodowej.

W przyszłości można by było stworzyć odpowiednik aplikacji internetowej w postaci aplikacji mobilnej na system IOS oraz Android. Wprowadzenie konta użytkownika z dodatkowymi możliwościami.

**Bibliografia**

[1] Getting Started | BootstrapVue, <https://bootstrap-vue.org/docs#documentation-sections> (data dostępu 20.11.2022r.).

[2] Get started with Bootstrap · Bootstrap v5.2, <https://getbootstrap.com/docs/5.2/getting-started/introduction/> (data dostępu 20.11.2022r.).

[3] Bootstrap (framework) – Wikipedia, wolna encyklopedia, <https://pl.wikipedia.org/wiki/Bootstrap_(framework)> (data dostępu 20.11.2022r.).

[4] LiteDB :: A .NET embedded NoSQL database, <https://www.litedb.org/> (data dostępu 20.11.2022r.).

[5] What is ASP.NET Core? | .NET, <https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core> (data dostępu 20.11.2022r.).

[6] Create web APIs with ASP.NET Core | Microsoft Learn, <https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0> (data dostępu 20.11.2022r.).

[7] JavaScript End to End Testing Framework | cypress.io testing tools, <https://www.cypress.io/> (data dostępu 22.11.2022r.).

[8] Moment.js | Home, <https://momentjs.com/> (data dostępu 20.11.2022r.).

[9] Scopewu/qrcode.vue: A Vue.js component to generate qrcode, <https://github.com/scopewu/qrcode.vue> (data dostępu 22.11.2022r.).

[10] Download data, <http://www.worldfloraonline.org/downloadData;jsessionid=DEBD4E00FC95C2339448A54AFD1046B5> (data dostępu 20.11.2022r.).

[10]

# 7. Spis rysunków

[Rysunek 1. Interfejs graficzny – strona główna 7](#_Toc119792691)

[Rysunek 2. Interfejs graficzny – ewidencja roślin 8](#_Toc119792692)

[Rysunek 3. Interfejs graficzny – formularz edycji pól z ewidencji roślin 8](#_Toc119792693)

[Rysunek 4. Interfejs graficzny – ochrona roślin 9](#_Toc119792694)

[Rysunek 5. Interfejs graficzny – dodawanie nowego wpisu do ewidencji roślin 9](#_Toc119792695)

[Rysunek 6. Fragment pliku Startup.cs 13](#_Toc119792696)

[Rysunek 7. Fragment pliku Services/TaxonomyProvider.cs 13](#_Toc119792697)

[Rysunek 8. Plik Services/RegisterRepository.cs 14](#_Toc119792698)

[Rysunek 9. Dodanie rośliny do ewidencji - Diagram maszyny stanowej 15](#_Toc119792699)

[Rysunek 10. Kontrolka do wyszukiwania klasyfikacji - Diagram maszyny stanowej 16](#_Toc119792700)