POLITECHNIKA WROCŁAWSKA

WYDZIAŁ INFORMATYKI I ZARZĄDZANIA

KIERUNEK: Informatyka

SPECJALNOŚĆ:

## PRACA DYPLOMOWA

## INŻYNIERSKA

Tytuł pracy w języku polskim

Tytuł pracy w języku angielskim

AUTOR:

Paweł Ożóg

PROWADZĄCY PRACĘ:

# dr Ziemowit Nowak

OCENA PRACY:

### WROCŁAW …….

Spis treści

[PRACA DYPLOMOWA 1](#_Toc535873374)

[INŻYNIERSKA 1](#_Toc535873375)

[dr Ziemowit Nowak 1](#_Toc535873376)

[WROCŁAW ……. 1](#_Toc535873377)

[1. Wstęp 3](#_Toc535873378)

[2. Geneza podjęcia tematu 3](#_Toc535873379)

[3. Cel pracy 3](#_Toc535873380)

[4. Model aplikacji 3](#_Toc535873381)

[4.1. Specyfikacja wymagań 3](#_Toc535873382)

[4.1.1. Słownik pojęć 3](#_Toc535873383)

[4.1.2. Wymagania funkcjonalne 4](#_Toc535873384)

[4.1.3. Wymagania niefunkcjonalne 5](#_Toc535873385)

[4.1.4. Reguły biznesowe 5](#_Toc535873386)

[4.2. Model przypadków użycia 6](#_Toc535873387)

[5. Projekt aplikacji 6](#_Toc535873388)

[6. Narzędzia programistyczne 6](#_Toc535873389)

[6.1. IntelliJ 6](#_Toc535873390)

[6.2. Gradle 7](#_Toc535873391)

[6.2.1. Czym jest Gradle 7](#_Toc535873392)

[6.2.2. Inne narzędzia do budowania projektów 9](#_Toc535873393)

[6.3. Atom 10](#_Toc535873394)

[6.4. Git -poprawić 10](#_Toc535873395)

[6.4.1. Czym jest GIT 10](#_Toc535873396)

[6.4.2. HISTORIA GIT-a 11](#_Toc535873397)

[7. Technologie 12](#_Toc535873398)

[7.1. Relacyjna baza danych Oracle Express – poprawić składnię 12](#_Toc535873399)

[7.1.1. Bazy danych 12](#_Toc535873400)

[7.1.2. Systemy zarządzania bazami danych 13](#_Toc535873401)

[7.1.3. Oracle Database Express Edition 13](#_Toc535873402)

[7.1.3.1. Bazy danych i ich instancje 14](#_Toc535873403)

[7.2. Backend framework bazujący na Java SpringBoot 14](#_Toc535873404)

[7.2.1. Spring ma wady – bardziej po polsku ! 14](#_Toc535873405)

[7.3. Frontend framewok JavaScript Angular 4/5 15](#_Toc535873406)

[7.4. Spring Security 15](#_Toc535873407)

[8. Podsumowanie 15](#_Toc535873408)

[9. Spis rysunków 15](#_Toc535873409)

[10. Literatura 15](#_Toc535873410)

# 

1. **Wstęp**

# Geneza podjęcia tematu

# Cel pracy

# Model aplikacji

## Specyfikacja wymagań

### Słownik pojęć

* + - 1. Definicje
* Pacjent

definicja

* Lekarz

definicja

* Gabinet

definicja

* Wizyta
* System
  + - 1. Użytkownicy systemu

Osoby posiadające uprawnienia do korzystania z systemu za pośrednictwem konta.

1.5.1 Lekarz

Pełne uprawnienia

1.5.2. Pacjent

Okrojone uprawnienia

1.5.4 Gość

Potencjalny lekarz lub pacjent

1.6 Konto

Użytkownik systemu loguje się na nie w celu uzyskania uprawnień do korzystania z systemu oraz edycji swoich danych. Logowanie następuje po podaniu loginu i hasła. .

1.6.2 Konto Lekarz

Konto z uprawnieniami dla Lekarza.

1.6.3 Konto Pacjenta

Konto z uprawnieniami dla Pacjenta.

1.7 Uprawnienia

Określają zakres możliwości jakie przysługują danemu użytkownikowi.

1.22 Czytelny interfejs

Czytelna strona, łatwa w obsłudze, ułatwiająca poruszanie się w systemie klientowi.

### Wymagania funkcjonalne

1. Jako gość chce mieć możliwość rejestracji konta w celu wykonywania dostępu do aplikacji.
2. Jako gość chce mieć możliwość przeglądania lekarzy i gabinetów w celu otrzymania wiedzy na temat kompetencji lekarza i wyposażenia gabinetu.
3. Jako lekarz chcę mieć możliwość logowania na konto i możliwość zarządzania nim -zmiany danych osobowych w celu obsługi gabinetu.
4. Jako lekarz chcę mieć możliwość przeglądania wizyt z przypisanymi pacjentami
5. Jako klient chcę mieć możliwość przeglądania towarów, ich opisów i recenzji w celu sprawdzenia dostępnej ilości, ceny, specyfikacji towaru oraz opinii na jego temat.
6. Jako klient chcę mieć możliwość wyszukania towarów w celu znalezienia interesującej mnie oferty.
7. Jako klient chcę mieć możliwość dodania towarów do koszyka, przeglądanie składu koszyka oraz jego modyfikację w celu złożenia konkretnego zamówienia i uniknięcia omyłkowego zakupu.
8. Jako klient chcę mieć możliwość złożenia zamówienia w celu zakupu towaru.
9. Jako klient chcę mieć możliwość przeglądania swojej historii zamówień w celu weryfikacji zakupu.
10. Jako pracownik chcę mieć generowane raporty zamówień w celu przyspieszenia wysyłki oraz posiadania wszelkich informacji o osobie zamawiającej i towarach zamawianych.
11. Jako pracownik chcę mieć możliwość zarządzania towarami w celu ich dodawania, usuwania, edycji.
12. Jako administrator chcę mieć możliwość zarządzania systemem w celu zarządzania kontami użytkowników oraz w celu zarządzania magazynem.
13. Jako administrator chcę mieć możliwość zarządzania kontami użytkowników w celu usuwania, edytowania i tworzenia ich.
14. Jako administrator chcę mieć generowane raporty sprzedaży w celu predykcji sprzedaży ilości towarów w kolejnym miesiącu.
15. Jako pracownik chce mieć możliwość obsługi zamówienia

### Wymagania niefunkcjonalne

* + - 1. Jako użytkownik chcę mieć dostęp do strony sklepu za pomocą większości przeglądarek takich jak: Internet Explorer, Mozilla FireFox, Google Chrome, Safari, Opera. W celu korzystania z serwisu przy pomocy posiadanej przeglądarki, bez konieczności instalacji innej.
      2. Jako użytkownik w czasie korzystania ze strony wymagam bezpiecznego protokołu komunikacyjnego, w celu uniknięcia niebezpieczeństwa (np. wirusów, keyloggerów).
      3. Jako Lekarz chcę wydzieć wizyty zaraz po zalogowaniu bez konieczeności czekania na wynik wyszukiwania.
      4. Jako administrator chcę mieć możliwość tworzenia automatycznych kopii zapasowych.

### Reguły biznesowe

Nazwa aplikacja to WebGabinet. Jej zadanie jest ułatwienie obsługi jednoosobowej działalności gospodarczej w postacie gabinetu. Celem projektu jest dostarczenie użytkownikowi możliwości zarządzania wizytami lekarskimi, pacjentami oraz sprzętem medycznym w gabinecie stomatologicznym.

**Ograniczenia:**

* Hasło musi się składać z co najmniej 8 znaków i musi zawierać przynajmniej jedną cyfrę i co najmniej jedną wielką literę oraz znak specjalny
* Przy rejestracji wymagane jest podanie: emaila, hasła, imienia
* Towary zakupione w trakcie jednej sesji wygenerują dokładnie jedno zamówienie
* Adres email zarejestrowany w systemie odpowiada jednemu użytkownikowi
* Raport sprzedaży jest weryfikowany przez administratora
* Klient ma do wyboru trzy opcje wysyłki(odbiór osobisty, kurier, poczta)
* Klient ma do wyboru trzy opcje płatności(przelew, karta, gotówka)

**Wyzwalacze:**

* Użytkownik może dokonywać zakupów kiedy jest zarejestrowany
* Odbiór osobisty jest możliwy po okazaniu dokumentu tożsamości
* Wysyłka towaru następuje na adres podany w zamówieniu
* Jeśli klient doda towar do koszyka może złożyć zamówienie
* Klient posiada możliwość usunięcia towarów z koszyka
* Klient otrzyma paragon jeśli przy zamówieniu nie wybierze faktury

**Wnioski:**

* Pracownikom przysługuje obniżka na towary w wysokości 7%
* Przy zakupie większej ilości towarów niż 1 za wysyłkę płaci się tylko raz
* Kiedy towar ma niską popularność administrator może obniżyć cenę lub ogłosić promocję na dany artykuł

## Model przypadków użycia

# Projekt aplikacji

# Narzędzia programistyczne

## IntelliJ

IntelliJ IDEA jest dostępny jako bezpłatna edycja społecznościowa i pełnowartościowa wersja ostateczna. Z punktu widzenia licencjonowania, dobrą rzeczą jest to, że możesz użyć obu wersji do opracowania oprogramowania, które chcesz sprzedać. Warto wspomnieć, że nowe Studio Android, które służy do tworzenia aplikacji mobilnych na Androida, bazuje również na IntelliJ IDEA.

Szczegółowa tabela porównawcza znajduje się na stronie JetBrains: http://www.jetbrains.com/idea/features/editions\_comparison\_matrix.html. Krótko mówiąc, w wydaniu społecznościowym brakuje wielu funkcji, ale jeśli spojrzysz wystarczająco blisko, dostępnych jest kilka obejść. Na przykład, jeśli chcesz używać serwerów Tomcat lub Jetty w wersji Community Edition, możesz używać pluginów Maven do swobodnego uruchamiania i debugowania aplikacji internetowych. Możesz używać wersji Community Edition do tworzenia aplikacji wykorzystujących wiele frameworków, takich jak Play, Struts lub Spring. W końcu to wszystko Java. IDE nie pomoże ci w tym. Większość wskazówek dotyczących konfiguracji, ostrzeżeń, autouzupełniania i funkcji konfiguracji środowiska wykonawczego będzie niedostępna.

Natomiast wersja Ultimate Edition to w pełni funkcjonalny komercyjny IDE. Masz pełne wsparcie niemal wszystkich nowoczesnych frameworków i serwerów aplikacji. IDE pomoże Ci, zapewniając uzupełnianie kodu, wskazówki i diagramy. Obsługa języków w tej edycji jest również bardziej wszechstronna; dostaniesz na przykład analizę języków HTML i skryptów dostępnych w locie.

Oprócz dostarczonych funkcji, Ultimate Edition można podzielić na kategorie na podstawie licencji. W zależności od potrzeb można zakupić dowolną z następujących licencji:

Licencja komercyjna: IntelliJ IDEA może być używany przez dowolnego programistę w Twojej firmie, ale całkowita liczba jednoczesnych użytkowników nie może przekroczyć liczby zakupionych licencji.

Licencja osobista: IntelliJ IDEA może być używana tylko przez osobę, która ją kupiła. Możesz go używać na tylu komputerach, ile posiadasz, o ile jesteś jedynym użytkownikiem. Licencja osobista oczywiście może być również wykorzystana do opracowania komercyjnych produktów.

Dodatkowo istnieją opcje licencjonowania i rabaty oparte na grupie docelowej, dla startupów, studentów i nauczycieli, na edukację lub szkolenia, a na koniec na projekty open source.

## Gradle

### Czym jest Gradle

Tradycyjne tworzenie oprogramowania zazwyczaj wymaga pisania kodu, kompilowania kodu, uruchamiania testów i składania archiwum, które ostatecznie zostanie wdrożone lub rozproszone. Ponieważ projekty oprogramowania stały się bardziej złożone, dodatkowe kroki, takie jak wykonywanie statycznej analizy kodu, warunkowe włączanie zasobów i uruchamianie skanowania zabezpieczeń, stały się częścią głównego procesu budowania i wdrażania. Narzędzia do automatyzacji kompilacji pozwalają zautomatyzować te kroki, dzięki czemu kompilacje stają się powtarzalne i przewidywalne.

Opisując Gradle jednym zdaniem można stwierdzić że Gradle jest narzędziem służącym do budowania projektów1. Pozwala ono na zautomatyzowanie tego procesu. Używa się do tego tak zwanego języka domenowego - DSL (ang. Domain Specific Language), który ułatwia wykonywanie standardowych zadań związanych z budowaniem projektu.

Jeśli ktoś ma styczność wyłącznie z niezbyt dużymi projektami, nad którymi pracuje samodzielnie prawdopodobnie nie odczuwał potrzeby używania narzędzi tego typu. Jednak przy większych projektach narzędzie, które pozwala na zautomatyzowanie tego procesu jest bardzo pomocne.

Gradle (http://gradle.org/) to najnowszy dodatek do rodziny narzędzi do automatyzacji projektów Java. Jest on dostępny na licencji Apache License 2.0, a jego pierwsza wersja (0.7) została wydana w 2009 roku, a następnie wersja 2.0 w 2014 roku. W chwili pisania tej książki wersja 2.7 jest aktualną wersją Gradle. Gradle zyskuje na popularności, ponieważ czerpie z doświadczeń innych istniejących narzędzi do budowania, takich jak Ant i Maven. Kilka wysokobudżetowych projektów, takich jak Android, Spring Framework i Hibernate, zmigrowało swoje systemy kompilacji, aby korzystać z Gradle.

#### Deklaratywne zarządzanie zależnościami

Większość projektów Java opiera się na innych projektach i frameworkach open source, aby działały poprawnie. Ręczne pobieranie tych zależności może być uciążliwe i śledzić ich wersje w miarę ich używania w projektach. Aby było to bardziej skomplikowane, zależności te mogą mieć swoje własne zależności (określane jako zależności przechodnie), które muszą zostać rozwiązane i pobrane.

Gradle zapewnia wygodny sposób deklarowania zależności projektu. Następnie automatycznie pobiera te zależności (wraz z przejściowymi zależnościami) i pozwala używać ich w swoich projektach. Upraszcza to zarządzanie zależnościami projektowymi. Ważne jest, aby pamiętać, że mówisz tylko Gradle, a nie w jaki sposób.

#### Deklaratywne kompilacje

Gradle używa języka opartego na domenie Groovy (http://groovy-lang.org/) (DSL-see https://docs.gradle.org/current/dsl/) do deklarowania kompilacji. DSL udostępnia zestaw elementów językowych, które można łatwo zmontować, aby tworzyć skrypty budujące, które są proste i wyraźnie wyrażają swoje zamiary.

#### Buduj według konwencji

Gradle zapewnia rozsądne wartości domyślne i konwencje dla projektów Java, Groovy, Web, Scala, Android i OSGi. Na przykład Gradle zaleca, aby cały kod źródłowy produkcji dla projektu Java znajdował się w folderze src \ main \ java. W ten sam sposób zawiera zalecenia dotyczące miejsca, w którym powinien znaleźć się kod testowy i zasoby. Ponadto zadania domyślne są konfigurowane automatycznie dla projektu Java, który skompilowałby kod, a następnie uruchom, przetestuj i wygeneruj artefakt JAR.

Przestrzeganie tych konwencji sprawi, że skrypty budujące będą bardzo zwięzłe. Nie musisz jednak przestrzegać tych konwencji. Ponieważ program DSL firmy Gradle oparty jest na Groovy, łatwo jest napisać kod Groovy, aby poprawić i odstąpić od tych konwencji.

#### Przyrostowe kompilacje

Złożone projekty często są uruchamiane w powolnych czasach kompilacji, ponieważ narzędzie do budowania próbuje "wyczyścić" i odbudować wszystko. Gradle rozwiązuje ten problem, zapewniając przyrostowe kompilacje, które pomijają wykonanie zadania, jeśli ani dane wejściowe, ani wyjściowe nie uległy zmianie. Na przykład zadanie JavaCompile pobiera zestaw plików źródłowych Java jako dane wejściowe i generuje zestaw plików klas. Gradle używa tych informacji, aby sprawdzić, czy pliki źródłowe się zmieniły i jeśli nie zostaną wykryte żadne zmiany, zadanie zostanie pominięte.

#### Gradle Wrapper

Gradle Wrapper to po prostu plik wsadowy (gradlew.bat) w środowisku Windows i skrypt powłoki dla środowisk Linux / Mac. Po uruchomieniu skrypt otacza i instaluje nową kopię środowiska wykonawczego Gradle na komputerze i wykonuje kompilację Gradle. Gradle Wrapper ułatwia uruchamianie nowych serwerów ciągłej integracji (CI), które mogą uruchamiać kompilacje bez dodatkowej konfiguracji. Opakowanie ułatwia również dystrybucję kodu i współpracę z innymi, ponieważ odbiorcy mogą z łatwością go skompilować.

#### Wtyczki

Gradle ułatwia rozszerzanie i dostosowywanie jego funkcjonalności za pomocą wtyczek. Wtyczki są komponentami, które można rozpowszechniać, które zawierają logikę wielokrotnego użytku i logikę zadania. Korzystając z wtyczek, można obsługiwać dodatkowe języki, tworzyć nowe zadania lub modyfikować istniejące funkcje zadań i rozszerzać język kompilacji, dodając nowe słowa kluczowe. Dzięki Gradle możesz z łatwością tworzyć własne wtyczki, umożliwiając integrację zadań i przepływów pracy specyficznych dla Twojej organizacji.

#### Opensource

Gradle jest open source i nic nie kosztuje, aby pobrać i używać. Zawiera bogatą dokumentację online i wsparcie aktywnej społeczności. Dodatkowo Gradle Inc. oferuje doradztwo i wsparcie handlowe dla ekosystemu Gradle.

### Inne narzędzia do budowania projektów

#### Alternatywy Gradle

Oczywiście Gradle nie jest jedynym narzędziem, które pomaga przy budowaniu projektów. Wymienić tu trzeba byłoby kilka innych jak Ant, Maven, Ivy, Make czy Buildr. Nie jest to oczywiście pełna lista.

Dodatkowo „problem budowania” projektu występuje w każdym języku programowania, więc analogiczne narzędzia występują także dla innych języków.

#### Ant + Ivy

Apache Ant (http://ant.apache.org) to popularne narzędzie open source do tworzenia skryptów. Wydany w 2000 roku był pierwszym wśród "nowoczesnych" narzędzi kompilacji dla ekosystemu Java. Ramy są oparte na Javie i używają rozszerzalnego języka znaczników (XML) do swojej konfiguracji. Domyślnym plikiem konfiguracyjnym dla Ant jest plik build.xml.

Używanie Anta zazwyczaj wiąże się z definiowaniem zadań i celów. Jak sama nazwa wskazuje, zadanie Ant jest jednostką pracy, którą należy wykonać. Typowe zadania obejmują tworzenie katalogu, uruchamianie testu, kompilowanie kodu źródłowego, tworzenie pliku archiwum aplikacji WWW (WAR) i tak dalej.

Ant nie narzuca żadnych konwencji ani ograniczeń dotyczących Twojego projektu i jest niezwykle elastyczny. Ta elastyczność czasami skutkowała złożonymi, trudnymi do zrozumienia i utrzymania plikami build.xml.

Apache Ivy (http://ant.apache.org/ivy/) zapewnia zautomatyzowane zarządzanie zależnościami, dzięki czemu Ant jest bardziej radosny w użyciu. W przypadku Ivy deklarujemy zależności w pliku XML o nazwie ivy.xml, jak pokazano na listingu 1-2. Integracja Ivy z Ant polega na deklarowaniu nowych celów w pliku build.xml w celu pobrania i rozwiązania zależności.

#### Maven

Apache Maven jest obecnie najpopularniejszym narzędziem do automatyzacji kompilacji / zarządzania projektami w ekosystemie Java. Wydany w 2004 roku Maven próbował rozwiązać wiele problemów napotykanych przez użytkowników Ant. Wiąże się on z konwencją dotyczącą konfiguracji i wprowadza standardową strukturę katalogów do projektów. Wprowadzono także deklaratywne zarządzanie zależnościami i automatycznie pobrano potrzebne zależności z zewnętrznych repozytoriów.

Maven używa XML do dostarczania projektu, metadanych zależności i konfiguracji kompilacji. Ta informacja jest dostępna w pliku pom.xml zdefiniowanym w katalogu głównym projektu.

## Atom

## Git -poprawić

### Czym jest GIT

Git jest popularnym i szeroko stosowanym systemem zarządzania źródłami, który znacznie upraszcza cykl programowania. Umożliwia użytkownikom tworzenie, używanie i przełączanie między gałęziami w celu tworzenia treści tak łatwo, jak ludzie tworzą i przełączają się między plikami w ich codziennym przepływie pracy. Jest on realizowany za pomocą szybkiej, wydajnej architektury, która pozwala na łatwość eksperymentowania i udoskonalania lokalnych zmian w izolowanym środowisku przed udostępnieniem ich innym. Krótko mówiąc, pozwala on zwykłym użytkownikom skupić się na poprawianiu treści, zamiast martwić się zarządzaniem źródłami, zapewniając jednocześnie bardziej zaawansowanym użytkownikom możliwość nagrywania, edytowania i udostępniania zmian na dowolnym poziomie szczegółowości.

Krótko mówiąc, Git jest inny - naprawdę. Kiedy masz doświadczenie w korzystaniu z Git i rozumiesz je, dzięki temu poczujesz się wzmocniony i produktywny. Kiedy jesteś nowicjuszem w Git i próbujesz go zrozumieć, natkniesz się na model, który poprowadzi cię do myślenia inaczej o zarządzaniu zawartością w kontroli źródła.

Aby to zilustrować, istnieje stare powiedzenie, że "kiedy wszystko, co masz, to młotek, wszystko wygląda jak gwóźdź". Kiedy wszystko, co masz, to tradycyjny, scentralizowany system zarządzania źródłami, wszystko wygląda jak zmiana pliku, która jest droga Oddział.

Nie tak z Git. Git jest jednym z tych dobrych narzędzi, które pozwalają użytkownikom skupić się na opracowywaniu treści i upraszczaniu obiegów pracy. To nie jest kolejne narzędzie w skrzynce narzędziowej, to zestaw narzędzi. Zawiera wszystkie narzędzia potrzebne do zarządzania śledzeniem wszystkiego, od kilku plików dla pojedynczego użytkownika po projekty obejmujące setki użytkowników i ogromny zakres, na przykład jądro Linux. Obecnie wiele dużych firm korzysta z Git. Jest bezpłatny, jest potężny, skaluje się, a jego model działa, gdy jest używany zgodnie z projektem.

Git ma również pewne "odczucie", które jest atrakcyjne dla wielu osób. Git jest skonstruowany bardziej jak szereg pojedynczych narzędzi, które możesz uruchomić wbrew treściom, podobnie jak w przypadku użytkowników korzystających z systemów operacyjnych. Jednak nie stara się być systemem; zapewnia użytkownikom najwyższą kontrolę nad zawartością, nawet w razie konieczności aktualizacji historii. Git zarządza podstawowymi jednostkami, które odpowiadają strukturom katalogów, a nie poszczególnym plikom, więc zawartość, która rozciąga się na granice plików i katalogów, może być zarządzana razem. Git upraszcza rozgałęzianie, do momentu, w którym tworzenie, scalanie lub usuwanie gałęzi staje się prawie tak szybkie i łatwe jak tworzenie, scalanie lub usuwanie plików. Zapewnia również lokalne środowisko z pełną kontrolą zarządzania źródłami, które można aktualizować niezależnie od współużytkowanego środowiska publicznego.

Biorąc pod uwagę, że różni się od innych systemów zarządzania kodami źródłowymi (SCM), warto zrozumieć, w jaki sposób powstał Git. Poniższa sekcja zawiera niektóre z jej historii.

### HISTORIA GIT-a

Git ma swoje korzenie w środowisku programistycznym dla jądra Linux. We wczesnych latach 2000, zespół pracujący nad jądrem zaczął używać zastrzeżonego rozproszonego systemu kontroli źródła o nazwie BitKeeper (czasami skracanego jako BK). Początkowo zespół mógł korzystać z tego systemu za darmo. Z biegiem czasu różnice zdań rozwinęły się wokół wykorzystania BK do tego stopnia, że ​​właściciel tego systemu odwołał swobodne korzystanie z produktu. W tym czasie (w 2005 r.) Linus Torvalds, twórca systemu Linux, postanowił stworzyć nowy system, który utrzymywał rozproszony ideał, ale także zawierał kilka dodatkowych koncepcji, nad którymi pracował. Być może, co najważniejsze, chciał zapewnić szybką wydajność, jakiej wymagałby projekt z zakresu jądra Linux. W ten sposób powstała motywacja i pomysły na to, czym stała się Git.

Rozwój rozpoczął się na początku kwietnia 2005 r., A wstępne wydanie było gotowe do lipca. Pierwotnie istniał pomysł, aby Git był zestawem narzędzi, który mógłby mieć inne systemy zaimplementowane na nim. Jednak z czasem został przekształcony w pełnoprawny SCM.

Jeśli zastanawiasz się nad nazwą, istnieje wiele definicji słowa Git, ale wszystkie z nich implikują negatywną konotację na temat danej osoby. Git otrzymał swoją nazwę od swojego twórcy. Linus żartobliwie stwierdził, że nazwał wszystkie swoje projekty po sobie.

Narzędzie używane produkcyjnie

Od wczesnych początków Git stał się standardowym narzędziem branżowym. Oczywiście standard branżowy jest terminem względnym. Niemniej jednak, w oparciu o prawie wszystkie kryteria, Git pasuje. Jest stosowany na wszystkich poziomach przemysłu. Ogromne projekty, takie jak jądro Linuksa, są w nim zarządzane, a także nakazują jego użycie (patrz poniższa lista). Jest kluczowym składnikiem wielu ciągłych rurociągów integracji / ciągłego dostarczania. Zapotrzebowanie na wiedzę na ten temat stale wzrasta. Projekty komercyjne i open-source oraz aplikacje rozpoznają, że jeśli wymagają usług zarządzania źródłami, muszą zintegrować się z Git. Projekty i firmy korzystające z Git obejmują

* Google
* Facebook
* Microsoft
* Twitter
* LinkedIn
* Netflix
* O'Reilly
* PostgreSQL
* Android
* Linux
* Eclispe

Podobnie jak w przypadku każdej wystarczająco udanej technologii open-source, cały ekosystem wyrósł wokół Gita. Ten punkt wart jest omówienia na chwilę. Podstawowe narzędzie, jakim jest Git, dało początek pozornie nieskończonej liczbie aplikacji, które jeszcze bardziej pomagają użytkownikom, którzy chcą z nim pracować - nazywane najczęściej słowami opartymi na git. Jeśli zaczniesz omawiać z kimś Git, możesz usłyszeć takie nazwy jak GitHub, Gitolite, Easy Git, Git Extensions, EGit i tak dalej. Dla niewtajemniczonych może to być trudne do zrozumienia, w jaki sposób każda z tych nazw odnosi się do oryginalnego narzędzia Git. Aby wyjaśnić niektóre z nieporozumień, dam ci przegląd tego, w jaki sposób różne oferty są podzielone na kategorie.

# Technologie

## Relacyjna baza danych Oracle Express – poprawić składnię

## Bazy danych

Komputery zazwyczaj przechowują i organizują duże ilości informacji w bazie danych. Baza danych, niezależnie od tego, czy komputer nią zarządza, jest niczym więcej niż uporządkowanym zbiorem powiązanych informacji. Baza danych to narzędzie, za pomocą którego można bezpiecznie przechowywać informacje i odpowiednio zorganizować je w celu szybkiego pobierania. Na przykład firma może używać bazy danych do przechowywania tabel rekordów klientów, odpowiednich zamówień sprzedaży, części produktów i list pracowników. Różni pracownicy mogą następnie korzystać z bazy danych, aby skutecznie wykonywać swoje zadania. Sprzedawcy mogą na przykład szybko wejść lub wyszukać zamówienia sprzedaży, kierownicy ds. Reklamy mogą analizować i prognozować sprzedaż produktów, a personel magazynowy może efektywnie zarządzać zapasami produktów.

Bazy danych występują w wielu odmianach. Odwrócone listy, hierarchiczne i sieciowe modele baz danych to starsze typy systemów baz danych przeznaczone przede wszystkim do określonych transakcji, które wprowadzają dane; nie są odpowiednie dla dynamicznych środowisk, w których interaktywna analiza danych ma kluczowe znaczenie.

Bardzo słabą stroną tych wcześniejszych systemów jest właśnie to, dlaczego relacyjne bazy danych dominują teraz w nowszych systemach zarządzania informacją. Relacyjne bazy danych są łatwe do zrozumienia, zaprojektowania i zbudowania. Relacyjne bazy danych przechowują i prezentują wszystkie informacje w tabelach, co jest łatwą do zrozumienia koncepcją. Ponadto relacyjne bazy danych ukrywają złożoność dostępu do danych od użytkownika, co sprawia, że ​​tworzenie aplikacji jest stosunkowo proste w porównaniu z innymi typami systemów baz danych.

## Systemy zarządzania bazami danych

System zarządzania bazami danych (DBMS) to oprogramowanie komputerowe, które zarządza dostępem do baz danych. Typowy wielozadaniowy DBMS wykonuje następujące zadania i nie tylko:

ImageA DBMS bezpiecznie zarządza współdzielonym dostępem do pojedynczej bazy danych wśród wielu jednoczesnych użytkowników. Na przykład blok DBMS blokuje dane, gdy użytkownicy dodają i aktualizują informacje, aby użytkownicy nie niszcząco zakłócali pracy innych użytkowników.

ImageA DBMS mądrze wykorzystuje zasoby komputera, dzięki czemu duża liczba użytkowników aplikacji może uzyskać dostęp do bazy danych i wykonywać pracę z krótkim czasem reakcji, co zapewnia maksymalną wydajność.

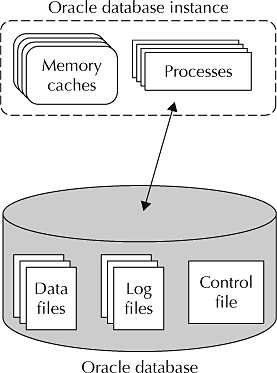
ImageA DBMS chroni dane bazy danych w taki sposób, że może odtworzyć utraconą pracę z powodu czegokolwiek, od prostej awarii zasilania, awarii dysku, a nawet kompletnej awarii witryny w niektórych przypadkach.

Możesz kupić jeden z kilku dostępnych na rynku DBMS, aby budować i zarządzać bazami danych. Obecnie wiodącym na rynku systemem DBMS jest Oracle Database Oracle, znana również jako Oracle. Najnowsza wersja Oracle to Oracle Database 10g. Dwa cele tej książki

mają na celu nauczyć cię, jak działa Oracle i zapewniają praktyczne doświadczenie z wykorzystaniem najczęściej używanych funkcji oprogramowania.

## Oracle Database Express Edition

Oracle Database 10 g Express Edition Bezpłatna-wersja Oracle obsługująca najpopularniejsze funkcje bazy danych Oracle. Nazywana i od tej pory określana w tej książce jako Oracle XE, można użyć Express Edition do projektowania, tworzenia, wdrażania i obsługi aplikacji opartych na Oracle, które działają na systemach jednoprocesorowych z bazami danych o wielkości 4 gigabajtów (GB) lub mniejszych. Oracle XE jest bardzo łatwy w instalacji i automatycznie zawiera zaawansowane narzędzia do tworzenia aplikacji, które można wykorzystać do szybkiego korzystania z Oracle. Płyta CD tej książki zawiera bezpłatną kopię Oracle XE, ale zawsze możesz pobrać najnowszą wersję ze strony internetowej Oracle.



### Bazy danych i ich instancje

Baza danych Oracle to zbiór powiązanych plików systemu operacyjnego, które firma Oracle wykorzystuje do przechowywania zestawu powiązanych informacji i zarządzania nimi. Strukturalnie baza danych Oracle ma trzy podstawowe typy plików: pliki danych, pliki dzienników i pliki kontrolne. Kolejne rozdziały tej książki objaśnią więcej na temat celu i zarządzania każdym typem pliku bazy danych.

## Backend framework bazujący na Java SpringBoot

## Spring ma wady – bardziej po polsku !

Spring Framework istnieje od ponad dekady i znalazł miejsce jako de facto standardowy framework do tworzenia aplikacji Java. Przy tak długiej i upartej historii niektórzy mogliby pomyśleć, że projekt Spring osiadł, opierając się na laurach i nie robi nic nowego ani ekscytującego. Niektórzy mogbliby nawet powiedzieć, że Spring jest spuścizną i że nadszedł czas, aby szukać innowacji w innych miejscach. Jest całkowicie błędne myślenie.

W ekosystemie Spring wydarzyło się wiele ekscytujących nowych rzeczy, w tym praca w dziedzinie przetwarzania w chmurze, dużych zbiorów danych, braku danych na podstawie schematów, programowania reaktywnego i opracowywania aplikacji po stronie klienta.

Być może najbardziej ekscytującą, najbardziej wzruszającą i zmieniającą gry nową rzeczą, jaka pojawi się na Spring w ubiegłym roku, jest Spring Boot. Spring Boot oferuje nowy paradygmat do rozwijania aplikacji Spring z minimalnym tarciem. Dzięki Spring Boot będziesz w stanie rozwijać aplikacje Spring z większą zwinnością i być w stanie skupić się na zaspokojeniu potrzeb twojej aplikacji przy minimalnym (lub nawet bez) myśleniu o konfiguracji Spring. Tak naprawdę jedną z głównych rzeczy, którą robi Spring Boot, jest wyprowadzenie Springa z drogi, aby można było załatwiać sprawy.

## Frontend framewok JavaScript Angular 4/5

## Spring Security

# Podsumowanie

# Spis rysunków

# Literatura

**LITERATURA PODSTAWOWA:**

[1] Jason Price, Oracle Database 12c i SQL. Programowanie, Helion, Gliwice 2015

[2] Craig Walls, Spring w akcji. Wydanie IV, Gliwice 2015

[3] Cay S. Horstmann, Java. Techniki zaawansowane. Wydanie X, Helion, Gliwice 2017

[4] Gion Kunz, Angular 2. Tworzenie interaktywnych aplikacji internetowych, Helion, Gliwice 2017

**LITERATURA UZUPEŁNIAJĄCA:**

[1] Radosław Sokół, Testowanie aplikacji Java za pomocą JUnit, Helion, Gliwice 2018

[2] Tim Berglund, Gradle Beyond the Basics 2013 ebook

[3] Dokumentacja produktów i rozwiązań systemów webowych

[4] John Vlissides, Ralph Johnson, Richard Helm, Erich Gamma,Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional 1994

[5]Agile for Everybody. Creating Fast, Flexible, and Customer-First Organizations, Matt LeMay, 2018 O'Reilly Media ebook

[6] Sudha Belida, Balaji Varanasi: Introducing Gradle Apress 2015

[7] Jarosław Krochmalski: IntelliJ IDEA Essentials Packt Publishing 2014