Spis treści

1 Wstęp **1**

1.1 Idea pracy dyplomowej 1

1.2 Założenia projektu 1

1.3 Cel pracy 1

1.4 Zakres pracy 1

2 Wykorzystane technologie **3**

2.1 JavaScript 3

2.2 Node.js 4

2.3 Socket.IO 4

2.4 MongoDB 4

2.5 PhoneGap 5

3 Architektura systemu **7**

3.1 Serwer 7

3.2 Klient 7

3.3 Model bazy danych 7

3.4 Panel zarządzania 7

3.5 Komunikacja Klient-Serwer - API 7

4 Uruchomienie aplikacji **9**

4.1 Aplikacja serwerowa 9

4.2 Aplikacja mobilna na platformę Android 9

5 Panel zarządzania - konfiguracja **11**

5.1 Zarządzanie listą placów manewrowych 11

5.2 Zarządzanie listą instruktorów 11

6 Klient dla systemu Android - korzystanie **13**

6.1 Przykładowy przepływ 13

6.2 Pogląd na listę placów 13

6.3 Zajmowanie i zwalnianie placu 13

7 Podstawowe elementy **15**

7.1 Sekcje dokumentu 15
7.1.1 Podpodrozdział 15
7.2 Podstawowe elementy typograficzne 15
7.2.1 Twarda spacja 15
7.2.2 Formatowanie tekstu 15
7.3 Podział linii i paragrafy 16
7.4 Środowisko matematyczne 18
7.4.1 Twierdzenia i dowody 18

8 Rysunki, tabele i inne konstrukcje **21**
8.1 Rysunki 21
8.1.1 Wielostronicowe obrazki 21
8.1.2 Wymuszanie renderowania obrazków do pewnego miejsca 21
8.2 Listingi 22
8.3 Algorytmy 23
8.4 Tabele 24

9 Inne przydatne konstrukcje **27**
9.1 Symbole otoczone kółkiem 27
9.2 Tymczasowa zmiana rozmiaru strony 27
9.3 Wpisy bibliograficzne 27

Podziękowania

Pragnę podziękować mojemu Promotorowi za wsparcie merytoryczne i motywację, a także najbliższym mi osobom, za doping, wiarę w moje możliwości oraz wyrozumiałość, kiedy nie mogłem poświęcać Im tyle czasu, ile bym chciał.

Dziękuję również wszystkim zainteresowanym osobom, które wypytywały mnie o szczegóły techniczne i postępy tej pracy. Dzięki Wam mogłem lepiej zrozumieć mój projekt.

Rozdział 1

Wstęp

1.1 Idea pracy dyplomowej

Niniejsza praca opisuje projekt i wykonanie mobilnego systemu komunikacji przeznaczonego dla Ośrodków Szkolenia Kierowców. Jego architekturę, część wizualną interfejsów, sposób instalacji i uruchomienia, zalecane przypadki użycia oraz opracowane lub wykorzystane algorytmy.

1.2 Założenia projektu

Głównym założeniem projektu było stworzenie prostej aplikacji mobilnej, która będzie przyjazna dla użytkowników, a jednocześnie nadal będzie spełniała swoje zadanie - ułatwiała komunikacji pomiędzy instruktorami Ośrodków Szkolenia Kierowców w celu zmniejszenia kolizji zajętości placów manewrowych.

Kolejnym założeniem było wykorzystanie języka JavaScript w każdej płaszczyźnie aplikacji, zarówno klienta, jak i serwera oraz użycie nierelacyjnej bazy danych, która będzie przechowywała dane o strukturze zbliżonej do struktury obiektów tego języka.

1.3 Cel pracy

Celem pracy jest projekt i realizacja systemu mającego za zadanie ułatwić pracę instruktorów Prawa Jazdy poprzez zautomatyzowanie wymiany informacji dotyczącej zajętości placów manewrowych, z których mogą korzystać.

1.4 Zakres pracy

Projekt praktyczny, stworzony dla celów niniejszej pracy jest znaczną częścią trudu włożonego w jej powstanie i w jego skład wchodzi:

- aplikacja mobilna dla systemu Android
- aplikacja serwerowa będąca zapleczem dla mobilnego systemu.

Rozdział 2

Wykorzystane technologie

2.1 JavaScript

JavaScript jest prototypowym obiektowym językiem programowania. Jest to język Internetu, ze względu na to, że jego kod potrafi uruchomić prawie każda przeglądarka internetowa. Po ciężkim okresie, kiedy to został ogólnie przyjęty jako zabawka umożliwiająca tworzenie tzw. wodotrysków na stronach WWW, rozrósł znacząco i dzisiaj jest jednym z najpopularniejszych, a na pewno najsłynniejszym spośród internetowych języków programowania.

Dzisiejszą "Dobę Internetu"pełnej technologii AJAX, rozbudowanych klienckich aplikacji internetowych, aplikacji internetowych przypominających desktopowe m.in. czytników RSS, aplikacji biurowych, klientów pocztowych, interaktywnych map, programów do obróbki grafiki i wideo etc. do historii odeszły długotrwałe wywołania stron wykonywane przy każdym wejściu użytkownika w interakcję z przeglądarką. To wszystko możemy zawdzięczać głównie dzięki JavaScriptowi.

Początkowe wersje interpreterów JavaScript zostały osadzone w jednych z pierwszych wersji przeglądarek Netscape i Internet Explorer. Ta kliencka wersja JavaScriptu pozwala na umieszczanie wykonywalnej zawartości w stronach internetowych, co pozwala na budowanie interakcji z użytkownikiem, to coś więcej niż statyczne strony WWW. Skrypty języka JavaScript odpowiedzialne są za część zachowawczą stron, kiedy to HTML (ang. HyperText Markup Language) opisuje zawartość i strukturę DOM (ang. Document Object Model) oraz treść, a CSS (ang. Cascading Style Sheets) opisuje wygląd tych elementów.

JavaScript jest otwartym standardem, ale standaryzacją tego języka zajmuje się ECMA International (ang. ang. European Computer Manufacturers Association), która na jego podstawie wydała standard języka skryptowego o nazwie ECMAScript, którego już niedługo szósta wersja wprowadzić ma nowe mechanizmy obiektowości, takie jak np. pełnowymiarowe dziedziczenie za pomocą klas (które dzisiaj jest realizowane przez prototypowanie obiektów), modularyzację i hermetyzację (które dzisiaj są realizowane przez zewnętrzne biblioteki wprowadzające tego typu abstrakcje), a także wiele innych usprawnień, które zapewne przyciągną jeszcze większe rzesze programistów na całym świecie.

Obecnie przeglądarki WWW to nie jedyne interpretery w obrębie których możliwe jest uruchomienie kodu JavaScript. Można tworzyć kod do zastosowań po stronie serwera WWW (m.in .NET lub Node.js), skrypty i aplikacje wiersza poleceń CLI (ang. Command Line Interface), aplikacje desktopowe (Node-webkit, Alchemium), aplikacje dla urządzeń przenośnych (m.in Sencha Touch, PhoneGap), aplikacje dla Smart TV (m.in. Mautilus), sterowanie mikrokomputerami takimi jak Raspberry PI (np. pijs.io) i Arduino (np. Johnny-Five, noduino), systemy operacyjne (Firefox OS, Chrome OS) rozszerzenia aplikacji takich jak Adobe Photoshop, Google Chrome, Mozilla Firefox.

2.2 Node.js

Node.js jest stosunkowo nową platformą deweloperską stworzoną przez Ryana Dahla, pozwalającą programistom JavaScript na tworzenie kodu serwerowego o wysokiej wydajności.

Dokładniej rzecz ujmując jest to silnik Google V8 z projektu Chromium (Google Chrome) opakowany w taki sposób, aby można było go wykorzystywać jako niezależny interpreter z możliwościami podobnymi do np. Ruby, Pythona, PHP itd.

Asynchroniczna, bazująca na zdarzeniach charakterystyka języka JavaScript sprawia, że Node.js jest bardzo wydajny, a ze względu na jego uniwersalność i wysoki poziom abstrakcji m.in. dynamiczne typowanie zmiennych, przyjazną składnię wbudowanych instrukcji i złożonych obiektów JSON (ang. JavaScript Object Notation), pisanie w nim to czysta przyjemność.

Podobnie jak większość dystrybucji Linuksa lub języki programowania takie jak Ruby, Python czy PHP, Node.js posiada swój manager pakietów NPM (ang. Node Package Manager) agregujący ogromną ilość pakietów rozszerzających jego możliwości.

Kolejnym atutem jest możliwość rozwijania kodu serwerowego i klienckiego w jednym języku programowania, przykładowo algorytmy walidacji, metody wejścia/wyjścia i operacje na plikach binarnych, co powoduje coraz większe zainteresowanie wśród programistów innych technologii typu server-side takich jak Python, PHP, Ruby, Java, Perl, C#.

2.3 Socket.IO

HTML5 WebSocket to technologia zapewniająca kanał komunikacji pomiędzy przeglądarką internetową, a serwerem internetowym w obu kierunkach za pomocą jednego gniazda TCP.

Wszystkie nowoczesne przeglądarki wspierają już tą technologię, a Socket.IO jest biblioteką, która pozwala na stosunkowo szybkie tworzenie aplikacji czasu rzeczywistego w przeglądarkach i aplikacjach mobilnych za pomocą ujednoliconego interfejsu, zacierając przy tym różnice pomiędzy różnymi mechanizmami transportu i interpretacjami twórców oprogramowania.

Węzeł komunikacji w przypadku WebSockets jest w przeciwieństwie do AJAX otwarty na stałe pomiędzy klientem, a serwerem. Jednak dane pomiędzy socketami są przesyłane poprzez HTTP jako obiekty XHR (XMLHttpRequest) podobnie jak w przypadku AJAX'a.

2.4 MongoDB

MongoDB jest nierelacyjną bazą danych (NoSQL), która przechowuje dane w formacie klucz-wartość o postaci dokumentów JSON, takim samym jak obiekty JavaScript, co sprawia, że nie ma potrzeby konwersji danych podczas odczytu i zapisu danych przez Node.js, dzięki temu wymiana danych jest szybsza niż w przypadku nierelacyjnych danych a komunikacja nie blokuje systemów wejścia/wyjścia.

Dokumenty takie nie wymagają określonej struktury i dane w nich nie są sztywno powiązane relacjami, dzięki czemu są wysoce skalowalne i pozwalają na szybszy rozwój aplikacji dzięki skoncentrowaniu na celu, a nie środkach do jego osiągnięcia.

Mongoose

Istnieje wiele sterowników do MongoDB dla różnych języków programowania. W niniejszej pracy został użyty pakiet Mongoose.js, narzędzie typu ODM (Object Document Mapping), które dba o schemat modelu danych, jego walidację i zgodność typów dla konkretnych pól, a także abstrahuje relacyjne powiązania między kolekcjami bazy danych, jeśli zachodzi taka potrzeba.

2.5 PhoneGap

PhoneGap[1] jest frameworkiem do budowania aplikacji mobilnych o otwartym kodzie źródłowym stworzonym przez firmę Nitobi, z czasem odsprzedanym do Adobe Systems[2], która to nadal czynnie rozwija produkt.

Pozwala na tworzenie aplikacji webowych w technologiach HTML, CSS i Javascript, które po kompilacji za pośrednictwem PhoneGap umożliwiają uruchamianie ich jako natywnych aplikacji wybranego mobilnego systemu operacyjnego.

Pisząc jeden, uniwersalny kod źródłowy aplikacji można uzyskać pliki instalacyjne dla różnych platform mobilnych. W chwili pisania tego tekstu PhoneGap umożliwia kompilację do natywnych aplikacji dla:

- Android
- iOS
- Blackberry OS 6.0+
- Blackberry 10
- WebOS
- Windows Phone 7 + 8
- Symbian
- Bada

Sercem PhoneGap jest otwarty projekt Apeche Cordova, umożliwiający dostęp do niskopoziomowych warstw sprzętowych za pomocą wysokopoziomowych abstrakcji w języku JavaScript.

Jaśniej mówiąc, istnieje możliwość wykorzystania m.in. aparatu, akcelerometru, kompasu, odbiornika GPS, głośników i mikrofonu urządzania, na którym uruchamiana jest aplikacja, a także możliwy staje się dostęp do systemu plików z możliwością odczytu i zapisu, ustawień karty sieciowej urządzenia, listy kontaktów oraz emitowania zdarzeń takich jak powiadomienia zarówno dźwiękowych, jak i wibracyjnych.

[1] http://phonegap.com
[2] http://www.adobe.com/pl/

Rozdział 3

Architektura systemu

3.1 Serwer

-

3.2 Klient

Tu opisać najpierw 3 podejścia do tworzenia aplikacji mobilnych - Przechowywanie w localstorage

3.3 Model bazy danych

3.4 Panel zarządzania

- Routing

3.5 Komunikacja Klient-Serwer - API

- API - WYmiana JSON

Rozdział 4

Uruchomienie aplikacji

4.1 Aplikacja serwerowa

4.2 Aplikacja mobilna na platformę Android

Rozdział 5

Panel zarządzania - konfiguracja

5.1 Zarządzanie listą placów manewrowych

5.2 Zarządzanie listą instruktorów

Rozdział 6

Klient dla systemu Android - korzystanie

6.1 Przykładowy przepływ

6.2 Pogląd na listę placów

6.3 Zajmowanie i zwalnianie placu

Rozdział 7

Podstawowe elementy

To jest rozdział.

7.1 Sekcje dokumentu

W Latexu w klasie dokumentów **book** wyróżniamy rozdziały (**chapter**), podrozdziały **section**, podpodrozdziały **subsection**, podpodpodrozdziały **subsubsection** i paragrafy (**paragraph**). Podpodpodrozdziały i paragrafy domyślnie nie są numerowane ani nie występują w spisie treści. Zachowanie to można zmienić poprzez funkcję **setcounter** umieszczaną w preambule. Wykomentowany przykład można znaleźć w kodzie tego dokumentu.

Obecnie znajdujemy się na poziomie podrozdziału. Pozostałe przykłady poniżej.

7.1.1 Podpodrozdział

To jest podpodrozdział.

Podpodpodrozdział

To jest podpodpodrozdział. On nie jest domyślnie numerowany.

Paragraf A to jest paragraf. On również nie jest domyślnie numerowany.

7.2 Podstawowe elementy typograficzne

7.2.1 Twarda spacja

Twarda spacja jest bardzo istotnym elementem, gdyż zabrania Latex'owi łamanie linii w miejscu jej wystąpienia, a tym samym pozwoli na niejako „sklejenie" wyrazów ze sobą. Dzięki temu możemy uniknąć tzw. sierot (pojedynczych znaków na końcu wiersza). W Latex twardą spację umieszcza się wstawiając znak tyldy (~). Zapisujemy to więc np. tak: „dokument, w~którym".

7.2.2 Formatowanie tekstu

Aby zapewnić poprawny wygląd tekstu należy pamiętać o kilku rzeczach:

- Linia poprzedzona procentem to komentarz.
- Poprzedzaniu spacji występującej po kropce kończącej skrót znakiem ucieczki, odstęp będzie wtedy taki, jak odstęp między wyrazami a nie między zdaniami. Przykładowo zapis „np. tekst" vs. „np. tekst". Ten drugi jest poprawny, a zapisany został tak: „np.\ tekst".
- Skróty pisane wielkimi literami kończące zdanie powinny posiadać \@ przed kropką kończącą zdanie, np. OCS\@. Spowoduje to potraktowanie spacji jako spacji międzyzdaniowej z nie międzywyrazowej.
- Cudzysłowie zawsze tworzymy używając podwójnego przecinka jako symbolu otwierającego cudzysłów, oraz podwójnego apostrofu zamykającego cudzysłów.
- Kursywę uzyskujemy za pomocą słowa kluczowego \textit, co w efekcie daje *tekst kursywą*. Pogrubiony **używamy słowa kluczowego textbf**. Każdorazowo tekst mający być napisany danym krojem otaczamy nawiasami klamrowymi.
- Myślnik (–) tworzymy poprzez umieszczenie bezpośrednio po sobie dwu kresek (minusy). Różnica między nimi jest zasadnicza. Pojedynczy myślnik generuje krótką kreskę (-), podwójny długą (–), potrójny najdłuższą (—).
- Odwołania do różnych elementów dokumentu robimy poprzez słowo kluczowe **ref()**. Jako jego parametr wstawiamy nazwę zdefiniowaną za pomocą słowa kluczowego **label()**. Należy pamiętać, że odwołanie zwraca jedynie numer elementu, słowo opisowe, jak np. rozdział czy rysunek należy dodać samodzielnie. Polecam tutaj przyjąć jakąś konwencję i się jej trzymać w całym dokumencie. Tak samo należy postępować w przypadku etykiet.
- Latex doskonale radzi sobie z dzieleniem wyrazów na końcach linii, jednak czasami zachodzi konieczność wymuszenia podziału w określonym miejscu. W tym celu należy zastosować konstrukcję \-. Latex takiego ukośnika nie wydrukuje dopóty, dopóki rzeczywiście w tym miejscu nie zostanie wykonane przeniesienie części wyrazu. Możliwe jest dodanie wielu podziałów w jednym wyrazie. Użyte wtedy zostanie to, które spowoduje wygenerowania „najładniejszego" tekstu.

7.3 Podział linii i paragrafy

Nowy paragraf rozpoczyna się poprzez wstawienie jednej wolnej linii. Latex automatycznie wygeneruje wcięcie. Należy pamiętać, że pierwszy paragraf, zgodnie ze standardami drukarskimi, nie ma wcięcia! Możemy tym sterować za pomocą poleceń **noindent** (brak wcięcia) oraz **indent** (dodatkowe wcięcie).

Jeżeli chcemy po prostu zrobić nową linię, bez tworzenia nowego paragrafu używamy konstrukcji \\. Efekt będzie taki, że paragraf
będzie kontynuowany w nowej linii. Nie spowoduje to jednak rozciągnięcia poprzedniej linii. Zostanie ona przerwana tam gdzie tego sobie zażyczymy i kontynuowana w nowej linijce.

A co w przypadku, gdy chcemy z jakiegoś powodu przerwać linię, ale wymusić justowanie tekstu? Weźmy dla przykładu fragment:

Trzecim istotnym aspektem jest stosowana w trakcie wytwarzania ontologii metodologia pracy [**?** **?**]. Zastosowanie jednej z uznawanych metodologii, takich jak Methontology, NeOn czy metodologia opracowana przez Noy i McGuiness, znacząco wpływa na jakoś uzyskanego produktu. Wspomniane metodologie w dużej mierze uwzględniają potrzebę przyszłej integracji wiedzy, a w połączeniu z narzędziami typu Protégé czy OCS [**?** **?** **?**], pozwalają na tworzenie spójnych i formalnie oraz logicznie poprawnych ontologii.

Tekst zostaje bardzo brzydko złamany w środku odnośników do cytowań. Użycie podwójnego po słowie „metodologia" w pierwszym zdaniu ukośnika da nam natomiast taki efekt:

Trzecim istotnym aspektem jest stosowana w trakcie wytwarzania ontologii metodologia pracy [**?** **?**]. Zastosowanie jednej z uznawanych metodologii, takich jak Methontology, NeOn czy metodologia opracowana przez Noy i McGuiness, znacząco wpływa na jakoś uzyskanego produktu. Wspomniane metodologie w dużej mierze uwzględniają potrzebę przyszłej integracji wiedzy, a w połączeniu z narzędziami typu Protégé czy OCS [**?** **?** **?**], pozwalają na tworzenie spójnych i formalnie oraz logicznie poprawnych ontologii.

Też nie ładnie, gdyż linijka jest niewyjustowana. Z pomocą przychodzi nam tutaj komenda **linebreak**[], gdzie w nawiasie kwadratowym podajemy liczbę od 1 do 4 określająca jak bardzo zależy nam na tym, by linia została złamana w tym miejscu (4 to najwyższa wartość). Efekt jest następujący:

Trzecim istotnym aspektem jest stosowana w trakcie wytwarzania ontologii metodologia pracy [**?** **?**]. Zastosowanie jednej z uznawanych metodologii, takich jak Methontology, NeOn czy metodologia opracowana przez Noy i McGuiness, znacząco wpływa na jakoś uzyskanego produktu. Wspomniane metodologie w dużej mierze uwzględniają potrzebę przyszłej integracji wiedzy, a w połączeniu z narzędziami typu Protégé czy OCS [**?** **?** **?**], pozwalają na tworzenie spójnych i formalnie oraz logicznie poprawnych ontologii.

Jeżeli z jakiegoś powodu potrzebujemy nową linię to używamy komendy **newpage**.

Tekst występujący po niej znajdzie się na nowej stronie. Rozdziały itp. automatycznie generują nową stronę, przy czym w układzie dwustronnym nowy rozdział zawsze zacznie się od nieparzystej strony.

7.4 Środowisko matematyczne

Środowisko matematyczne otwieramy i zamykamy znakiem $. Niektóre funkcje można używać tylko wewnątrz takiego środowiska. Przykładem niech będzie funkcja **mathcal** zamieniająca duże litery w symbole o charakterystycznym kroju, stosowanym do opisywania stałych, np. $\mathcal{O}$ czy $\mathcal{R}(\mathcal{D},\mathcal{P},\mathcal{T},\mathcal{S},\mathcal{U},\mathcal{I})$. Pamiętać należy, że zamienione zostaną wszystkie litery w wyrażeniu występującym wewnątrz nawiasów klamrowych.

Niektóre konstrukcje, np. równania, automatycznie włączają tryb matematyczny. Równania dobrze jest opisać, przykład przedstawia Równanie 7.1.

$$\mathcal{O}(\mathcal{K},\mathcal{B},\mathcal{C},\mathcal{R}) \tag{7.1}$$

gdzie:
$\mathcal{K}$ - zbiór klas wchodzących w skład ontologii,
$\mathcal{B}$ - zbiór bytów wchodzących w skład ontologii,
$\mathcal{C}$ - zbiór komentarzy przypisanych do klas $\mathcal{K}$ i bytów $\mathcal{B}$ wchodzących w skład ontologii,
$\mathcal{R}$ - zbiór relacji wiążących elementy ontologii.

W równaniach możemy stosować różne dodatkowe symbole oraz np. wyrównywać je do określonego miejsca. Służy do tego blok typu **split**, a sam punkt wyrównania określony jest ampersandem (&). Przykład zastosowania prezentuje równanie 7.2.

$$\begin{aligned} \forall x_1 \leq y_1, x_2 \leq y_2 : f(x_1 + x_2, y_1 + y_2) \\ = \frac{y_1}{y_1 + y_2} f(x_1, y_1) + \frac{y_2}{y_1 + y_2} f(x_2, y_2) \end{aligned} \tag{7.2}$$

7.4.1 Twierdzenia i dowody

Linie 75 – 93 nagłówka dokumentu definiują nowe nazwy sekcji twierdzeń i dowodów, oraz znacznik końca dowodu (taki czarny kwadracik). Dzięki nim można uzyskać ładnie wyglądające twierdzenia jak poniżej (Twierdzenie 7.4.1). Zauważmy, że równanie dowodu nie jest równaniem numerowanym. Wszędzie tam, gdzie nie chcemy by rozdział czy dowolna inna sekcja była numerowana należy w jej nazwie użyć gwiazdki, np. **begin{equation*}**.

Twierdzenie 7.4.1 *Podobieństwo pomiędzy pojęciami A i B opisane jest stosunkiem ilości informacji niezbędnej do opisania ich wspólności znaczeniowej oraz ilością informacji niezbędnej do ich opisania (Równanie 7.3).*

$$sim_{lin}(A,B) = \frac{\log P(common(A,B))}{\log P(description(A,B))} \tag{7.3}$$

Dowód

$$\begin{aligned} f(x,y) &= f(x+0, y+(y-x)) \\ &= \frac{x}{y} * f(x,x) + \frac{y-z}{x} * f(0, y-z) \\ &= \frac{x}{y} * 1 + \frac{y-z}{x} * 0 \\ &= \frac{x}{y} \quad \blacksquare \end{aligned}$$

Inna ciekawa konstrukcja wykorzystująca tryb matematyczny do zapisania pewnego stwierdzenia:

Niech $A \subseteq T$, $C = N_y(A) \neq W$, a $\alpha_y = \min_{a \in A, b \notin C}\{y(a) + y(b) - q(a,b)\}$ oraz

$$y'(v) = \begin{cases} y(v) - \alpha_y & \text{jeżeli } v \in A \\ y(v) + \alpha_y & \text{jeżeli } v \in C \\ y(v) & \text{w innych przypadkach} \end{cases}$$

Zapis ten, acz skomplikowany, pozwala na reprezentację złożonych reguł matematycznych w postaci ładnie ułożonych i wyrównanych wierszy. Reguły **left** oraz **right** pozwalają na utworzenie nawiasów klamrowych, których rozmiar będzie automatycznie dostosowywany do rozmiaru elementu, jakie mają zawierać.

Rozdział 8

Rysunki, tabele i inne konstrukcje

8.1 Rysunki

Obrazki wstawiamy do dokumentu za pośrednictwem sekcji **figure**. Przykładowy wyśrodkowany rysunek z etykietą i opisem to Rys. **??**.

W podanym przykładzie obrazek będzie wyśrodkowany, o szerokości odpowiadającej 70% szerokości tekstu. Plik obrazka znajduje się w katalogu img i nazywa się obrazek.png. Latex dołoży starań, żeby umieścić go w tym miejscu (znacznik h w definicji figury). Wykrzyknik oznacza, że Latex bardzo się postara by obrazek tu był. Opisy obrazków zawsze znajdują się pod obrazkiem.

Latex potrafi wygenerować spis wszystkich numerowanych figur i umieścić go w miejscu, gdzie użyjemy polecenia **listoffigures**.

8.1.1 Wielostronicowe obrazki

Czasami zdarza się, że obrazek jest tak duży, że umieszczenie go na jednej stronie powoduje, że stanie się nieczytelny. Latex umożliwia pocięcie obrazka na kawałki, przeskalowanie i przetworzenie tych kawałków, oraz wyświetlenie ich w postaci następujących po sobie figur, gdzie tylko pierwsza wskazana będzie numerowana i uwzględniona w spisie rysunków. Podpis pod każdą częścią będzie odmienny, numeracja będzie wspólna, w spisie wskazanie będzie do pierwszego z obrazków. Po szczegóły zapraszam do dokumentu w celu przeczytania komentarzy.

Należy zauważyć, że Latex dołoży wszelkich starań, by obrazek umieścić tam gdzie go sobie zażyczyliśmy. Czasami jest to jednak niemożliwe, ze względu na zbyt małą ilość tekstu i niestety trzeba pokombinować. Typowe zabiegi to próba dodania znacznika [!h] przy definicji figury, wstawienie obrazka wcześniej w tekście czy też lekka zmiana jego rozmiaru.

8.1.2 Wymuszanie renderowania obrazków do pewnego miejsca

Czasami zdarza się, że Latex zbyt długo zwleka z renderowanie obrazków, przez co w pamięci podręcznej zaczyna brakować miejsca. Najczęściej problem ten występuje, gdy dokument zawiera bardzo dużo obrazków, a raczej niewiele tekstu, lub relacje pomiędzy rozmiarami obrazków a liczbą słów są zachwiane i nie jest możliwe ich poprawne rozlokowanie.

Wszystkie zaległe obrazki zawsze renderowane są na koniec rozdziału, jednak ich duże nagromadzenie może spowodować błąd przepełnienia pamięci. Czasami też koniec rozdziału jest zbyt odległym miejscem i lepiej po prostu wymusić renderowanie zaległych ramek w określonym miejscu. Służy temu komenda **clearpage**. Jej zastosowanie spowoduje renderowanie wszystkich, do tej pory nierozlokowanych obrazków.

8.2 Listingi

Tworzenie listingów umożliwia pakiet **listings**. Należy zwrócić uwagę, że etykiety i opisy listingów dodaje się w dośc specyficzny sposób jako parametry do bloku **lstlisting** (patrz kod dokumentu). Używając funkcji **lstset** możliwa jest zmiana kroju czcionki na inną niż reszty dokumentu. W przykładzie (Listing 8.1) zmniejszono czcionkę do rozmiaru indeksu dolnego.

Listing 8.1: Przykładowy listing

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
 <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
 <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
 <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<rdf:RDF xmlns="http://kask.eti.pg.gda.pl/securityA.owl#"
 xml:base="http://kask.eti.pg.gda.pl/securityA.owl"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <owl:Ontology rdf:about="http://kask.eti.pg.gda.pl/securityA.owl"/>

 <!--
 ///////////////////////////////////////////////////////////////////////////////////////
 //
 // Classes
 //
 ///////////////////////////////////////////////////////////////////////////////////////
 -->

 <!-- http://kask.eti.pg.gda.pl/securityA.owl#DSA -->

 <owl:Class rdf:about="http://kask.eti.pg.gda.pl/securityA.owl#DSA">
  <rdfs:subClassOf rdf:resource="http://kask.eti.pg.gda.pl/securityA.owl#asymetryczne"/>
 </owl:Class>


 <!-- http://kask.eti.pg.gda.pl/securityA.owl#RSA -->

 <owl:Class rdf:about="http://kask.eti.pg.gda.pl/securityA.owl#RSA">
  <rdfs:subClassOf rdf:resource="http://kask.eti.pg.gda.pl/securityA.owl#asymetryczne"/>
 </owl:Class>


 <!-- http://kask.eti.pg.gda.pl/securityA.owl#asymetryczne -->

 <owl:Class rdf:about="http://kask.eti.pg.gda.pl/securityA.owl#asymetryczne"/>


 <!-- http://kask.eti.pg.gda.pl/securityA.owl#symetryczne -->

 <owl:Class rdf:about="http://kask.eti.pg.gda.pl/securityA.owl#symetryczne"/>
</rdf:RDF>
```

8.3 Algorytmy

Pakiety **algorithmic** oraz **algorithm** dostarczają środowiska do definiowania algorytmów. Można za ich pomocą definiować zarówno pseudokod jak i algorytmy ogólne. Pseudokod przedstawia Algorytm 1. Przykładową ogólną reprezentację algorytmu Levenshteina obrazuje Algorytm 2.

Algorytm 1 Pseudokod prezentujący jakiś bezsensowny algorytm

```
1: program FancyProgram {
2: function superFunkcja(List listA) {
3: for all element : listA.getElements() do
4:    int result = storeElement(element);
5:    if result == SUCCESS then
6:       double number = result*element;
7:    else if result == FAILURE then
8:       double number = result/element;
9:    end if
10: end for
11: }
12: input List elements;
13: output number;
14: superFunkcja(elements);
15: return  number;
16: }
```

Algorytm 2 Algorytm Levenshteina

1. Niech $n = \textit{długość}(s)$ a $m = \textit{długość}(t)$.
2. Jeżeli $n == 0$ zwróć m i zakończ działanie.
3. Jeżeli $m == 0$ zwróć n i zakończ działanie.
4. Utwórz macierz d o $m + 1$ wierszach indeksowanych od 0 do m oraz $n + 1$ kolumnach indeksowanych od 0 do n, pierwszy wiersz zainicjuj wartościami od 0 do n a pierwszą kolumnę od 0 do m.
5. Porównaj każdy znak pierwszego ciągu, indeksowany za pomocą i, z każdym znakiem drugiego ciągu, indeksowanym za pomocą j. Dla każdego porównania:
   (a) jeżeli $s[i]$ jest identyczne z $t[j]$, $koszt = 0$, w przeciwnym wypadku $koszt = 1$,
   (b) ustaw wartość komórki $d[i, j]$ macierzy na wartość minimalną spośród:
   - $d[i - 1, j] + 1$,
   - $d[i, j - 1] + 1$,
   - $d[i - 1, j - 1] + koszt$.
6. Odczytaj odległość $dist_{lev}(s, t)$ z komórki $d[n, m]$.

Podobnie jak w przypadku rysunków, algorytmy są automatycznie rozlokowywane w tekście przez system Latex.

8.4 Tabele

Latex pozwala na konstruowanie bardzo rozbudowanych tabel. Pełny opis możliwości znaleźć można w sieci Internet, a samo zagadnienie jest bardzo skomplikowane. Prosty przykład prezentuje Tabela 8.1.

Tablica 8.1: Opis tabelki

Pole 1	Pole 2	Pole 3
1	Pojęcie	Opis tegoż pojęcia
2	Pojęcie	Opis tegoż pojęcia

Pakiet **longtable** pozwala na generowanie ogromnych tabel rozłożonych na kilka stron.

Tablica 8.2: Bardzo długa tabela (główna etykieta)

Lemma B	Agency					Area				
Lemma A	$max(P_{lex}, P_{sem})$	P_{kom}	P_{str}	P_{sk}	Wynik	$max(P_{lex}, P_{sem})$	P_{kom}	P_{str}	P_{sk}	Wynik
Asset	0,30	0,00	0,06	0,02	Różne	0,40	0,00	0,06	0,02	Różne
Attack	0,33	0,00	0,09	0,03	Różne	0,33	0,00	0,09	0,03	Różne
Circumstance	0,37	0,00	0,00	0,00	Różne	0,85	X	X	X	Rodzeństwo
Continuity	0,31	0,00	0,13	0,04	Różne	0,42	0,00	0,13	0,04	Różne
Error	0,32	0,00	0,08	0,03	Różne	0,35	0,00	0,08	0,03	Różne
Event	0,46	0,00	0,10	0,03	Różne	0,40	0,00	0,10	0,03	Różne
Group	0,60	0,00	0,00	0,00	Różne	0,55	0,00	0,00	0,00	Różne
Harm	0,37	0,00	0,00	0,00	Różne	0,42	0,00	0,00	0,00	Różne
Mission	0,46	0,00	0,10	0,03	Różne	0,19	0,00	0,10	0,03	Różne
Operation	0,83	X	X	X	Rodzeństwo	0,43	0,00	0,50	0,15	Różne
Organization	0,72	X	X	X	B podrzędne A	0,35	0,00	0,10	0,03	Różne
Potential	0,36	0,00	0,00	0,00	Różne	0,41	0,00	0,00	0,00	Różne
Risk	0,32	0,00	0,17	0,05	Różne	0,20	0,00	0,17	0,05	Różne
Security	0,65	0,00	0,00	0,00	Różne	0,40	0,00	0,00	0,00	Różne
Threat	0,28	0,00	0,00	0,00	Różne	0,50	0,00	0,00	0,00	Różne
Value	0,30	0,00	0,11	0,03	Różne	0,47	0,00	0,11	0,03	Różne
Vulnerability	0,31	0,00	0,00	0,00	Różne	0,42	0,00	0,00	0,00	Różne
Weakness	0,34	0,00	0,11	0,03	Różne	0,44	0,00	0,11	0,03	Różne

Lemma B	Asset					Attack				
Lemma A	$max(P_{lex}, P_{sem})$	P_{kom}	P_{str}	P_{sk}	Wynik	$max(P_{lex}, P_{sem})$	P_{kom}	P_{str}	P_{sk}	Wynik
Asset	1,00	X	X	X	Tożsame	0,17	0,00	0,39	0,12	Różne
Attack	0,17	0,00	0,85	0,25	Różne	1,00	X	X	X	Tożsame
Circumstance	0,32	0,34	0,00	0,24	Różne	0,25	0,00	0,00	0,00	Różne
Continuity	0,27	0,00	0,50	0,15	Różne	0,26	0,00	0,25	0,08	Różne
Error	0,48	0,25	0,47	0,31	Różne	0,45	0,00	0,27	0,08	Różne
Event	0,32	0,20	0,64	0,33	Różne	0,51	0,00	0,42	0,13	Różne
Group	0,13	0,00	0,00	0,00	Różne	0,31	0,00	0,00	0,00	Różne
Harm	0,33	0,26	0,00	0,18	Różne	0,51	0,00	0,00	0,00	Różne
Mission	0,29	0,00	0,44	0,13	Różne	0,77	X	X	X	Rodzeństwo
Operation	0,31	0,00	0,15	0,05	Różne	0,95	X	X	X	B podrzędne A
Organization	0,55	0,31	0,35	0,33	Różne	0,76	X	X	X	Rodzeństwo
Potential	0,32	0,22	0,00	0,16	Różne	0,46	0,00	0,00	0,00	Różne
Risk	0,20	0,29	0,46	0,34	Różne	0,40	0,00	0,18	0,05	Różne
Security	0,31	0,00	0,00	0,00	Różne	0,39	0,00	0,00	0,00	Różne
Threat	0,33	0,23	0,00	0,16	Różne	0,40	0,00	0,00	0,00	Różne
Value	0,64	0,00	0,38	0,11	Różne	0,20	0,00	0,14	0,04	Różne
Vulnerability	0,27	0,31	0,00	0,22	Różne	0,08	0,00	0,00	0,00	Różne
Weakness	0,47	0,22	0,38	0,27	Różne	0,13	0,00	0,14	0,04	Różne

Lemma B	Circumstance					Countermeasure				
Lemma A	$max(P_{lex}, P_{sem})$	P_{kom}	P_{str}	P_{sk}	Wynik	$max(P_{lex}, P_{sem})$	P_{kom}	P_{str}	P_{sk}	Wynik
Asset	0,32	0,00	0,11	0,03	Różne	0,14	0,00	0,00	0,00	Różne

Tablica 8.2: (ciąg dalszy – ten tekst pokazywać się będzie na kolejnych stronach)

Attack	0,25	0,00	0,18	0,05	Różne	0,31	0,00	0,00	0,00	Różne
Circumstance	1,00	X	X	X	Tożsame	0,21	0,00	0,00	0,00	Różne
Continuity	0,33	0,00	0,25	0,08	Różne	0,29	0,00	0,00	0,00	Różne
Error	0,27	0,00	0,17	0,05	Różne	0,21	0,00	0,00	0,00	Różne
Event	0,99	X	X	X	A podrzędne B	0,26	0,00	0,00	0,00	Różne
Group	0,17	0,00	0,00	0,00	Różne	0,14	0,00	0,00	0,00	Różne
Harm	0,52	0,00	0,00	0,00	Różne	0,33	0,00	0,00	0,00	Różne
Mission	0,25	0,00	0,20	0,06	Różne	0,20	0,00	0,00	0,00	Różne
Operation	0,41	0,00	0,33	0,10	Różne	0,29	0,00	0,00	0,00	Różne
Organization	0,34	0,00	0,20	0,06	Różne	0,34	0,00	0,00	0,00	Różne
Potential	0,39	0,00	0,00	0,00	Różne	0,29	0,00	0,00	0,00	Różne
Risk	0,23	0,00	0,33	0,10	Różne	0,21	0,00	0,00	0,00	Różne
Security	0,49	0,00	0,00	0,00	Różne	0,73	X	X	X	Rodzeństwo
Threat	0,21	0,00	0,00	0,00	Różne	0,29	0,00	0,00	0,00	Różne
Value	0,33	0,00	0,22	0,07	Różne	0,21	0,00	0,00	0,00	Różne
Vulnerability	0,43	0,00	0,00	0,00	Różne	0,21	0,00	0,00	0,00	Różne
Weakness	0,36	0,00	0,22	0,07	Różne	0,14	0,00	0,00	0,00	Różne

Lemma B	Device					Event				
Lemma A	$max(P_{tex}, P_{sem})$	P_{kom}	P_{str}	P_{sk}	Wynik	$max(P_{tex}, P_{sem})$	P_{kom}	P_{str}	P_{sk}	Wynik
Asset	0,08	0,00	0,00	0,00	Różne	0,32	0,13	0,39	0,21	Różne
Attack	0,37	0,00	0,00	0,00	Różne	0,51	0,00	0,64	0,19	Różne
Circumstance	0,21	0,00	0,00	0,00	Różne	0,99	X	X	X	B podrzędne A
Continuity	0,27	0,00	0,00	0,00	Różne	0,32	0,00	0,25	0,08	Różne
Error	0,30	0,00	0,00	0,00	Różne	0,42	0,17	0,27	0,20	Różne
Event	0,34	0,00	0,00	0,00	Różne	1,00	X	X	X	Tożsame
Group	0,13	0,00	0,00	0,00	Różne	0,24	0,00	0,00	0,00	Różne
Harm	0,40	0,00	0,00	0,00	Różne	0,51	0,18	0,00	0,12	Różne
Mission	0,24	0,00	0,00	0,00	Różne	0,37	0,00	0,21	0,06	Różne
Operation	0,27	0,00	0,00	0,00	Różne	0,57	0,00	0,29	0,09	Różne
Organization	0,41	0,00	0,00	0,00	Różne	0,41	0,22	0,13	0,20	Różne
Potential	0,22	0,00	0,00	0,00	Różne	0,67	0,14	0,00	0,10	Różne
Risk	0,26	0,00	0,00	0,00	Różne	0,41	0,18	0,18	0,18	Różne
Security	0,83	X	X	X	Rodzeństwo	0,48	0,00	0,00	0,00	Różne
Threat	0,30	0,00	0,00	0,00	Różne	0,35	0,19	0,00	0,13	Różne
Value	0,18	0,00	0,00	0,00	Różne	0,32	0,00	0,14	0,04	Różne
Vulnerability	0,15	0,00	0,00	0,00	Różne	0,42	0,18	0,00	0,12	Różne
Weakness	0,25	0,00	0,00	0,00	Różne	0,35	0,23	0,14	0,21	Różne

Lemma B	Group					Harm				
Lemma A	$max(P_{tex}, P_{sem})$	P_{kom}	P_{str}	P_{sk}	Wynik	$max(P_{tex}, P_{sem})$	P_{kom}	P_{str}	P_{sk}	Wynik
Asset	0,13	0,00	0,00	0,00	Różne	0,33	0,00	0,00	0,00	Różne
Attack	0,31	0,00	0,00	0,00	Różne	0,51	0,00	0,00	0,00	Różne
Circumstance	0,17	0,00	0,00	0,00	Różne	0,52	0,00	0,00	0,00	Różne
Continuity	0,10	0,00	0,00	0,00	Różne	0,33	0,00	0,00	0,00	Różne
Error	0,20	0,00	0,00	0,00	Różne	0,45	0,00	0,00	0,00	Różne
Event	0,24	0,00	0,00	0,00	Różne	0,51	0,00	0,00	0,00	Różne
Group	1,00	X	X	X	Tożsame	0,11	0,00	0,00	0,00	Różne
Harm	0,11	0,00	0,00	0,00	Różne	1,00	X	X	X	Tożsame
Mission	0,46	0,00	0,00	0,00	Różne	0,26	0,00	0,00	0,00	Różne
Operation	0,17	0,00	0,00	0,00	Różne	0,39	0,00	0,00	0,00	Różne
Organization	0,87	X	X	X	A podrzędne B	0,48	0,00	0,00	0,00	Różne
Potential	0,11	0,00	0,00	0,00	Różne	0,40	0,00	0,00	0,00	Różne
Risk	0,32	0,00	0,00	0,00	Różne	0,28	0,00	0,00	0,00	Różne
Security	0,43	0,00	0,00	0,00	Różne	0,50	0,00	0,00	0,00	Różne
Threat	0,17	0,00	0,00	0,00	Różne	0,25	0,00	0,00	0,00	Różne
Value	0,46	0,00	0,00	0,00	Różne	0,33	0,00	0,00	0,00	Różne
Vulnerability	0,09	0,00	0,00	0,00	Różne	0,44	0,00	0,00	0,00	Różne
Weakness	0,10	0,00	0,00	0,00	Różne	0,36	0,00	0,00	0,00	Różne

Rozdział 9

Inne przydatne konstrukcje

9.1 Symbole otoczone kółkiem

Latex pozwala na tworzenie własnych znaków i symboli. W mojej rozprawie doktorskiej potrzebowałem zestawu symboli zamkniętych w okrąg, np. $\oplus$, $\ominus$, $\textcircled{k}$. Za pomocą pakietu **tikz** zdefiniowałem dwa makra: **mycircalign** oraz **mycirc**. Pierwsze z nich otacza symbol kółkiem i wyrównuje powstały obrazek tak, by jego środek pokrywał się ze środkiem sąsiadujących znaków. Drugi układa elementy na linii tekstu, tak że spody znaków sąsiadujących i powstałego obrazka są wyrównane. Jak każde makro, również i te można używać zarówno w tekście jak i w środowisku matematycznym, tabelkach itp. (Równanie 9.1).

$$\mathcal{O}_3 = \mathcal{O}_1 \textcircled{+} \mathcal{O}_2 \tag{9.1}$$

9.2 Tymczasowa zmiana rozmiaru strony

Czasami zachodzi konieczność chwilowej zmiany rozmiaru strony, by np. udało się zmieścić jedną dodatkową linijkę tekstu. Możemy to wykonać za pomocą polecenia **enlargethispage{}**, gdzie jako parametr podajemy rozmiar oraz jednostkę. Należy pamiętać, że polecenie to musi zostać wydane odpowiednio wcześniej, by Latex zdążył zastosować nowy rozmiar strony. Najlepiej by te polecenie było bezpośrednio przed nową stroną, jednak zazwyczaj jest to kwestia poeksperymentowania. Wartość przekazana jako parametr może być także ujemna. Przykładowo strona zawierająca rozdział 8.2 została w niniejszym dokumencie pomniejszona o 5 cm wymuszając przeniesienie początku rozdziału 8.3 na następną stronę.

Strona zawierająca rozdział 8.1 powiększono o 20 punktów, co pozwoliło uniknąć samotnej, pojedynczej linii tekstu kończącego akapit (tzw. wdowy) na następnej stronie. Takie drobne modyfikacje rozmiaru strony zazwyczaj są niezauważalne dla czytelnika a poprawiają ogólny układ dokumentu.

9.3 Wpisy bibliograficzne

Wpisy bibliograficzne przechowujemy w odrębnym pliku z rozszerzeniem .bib. Przykładowy plik został dołączony do tego dokumentu. Do pliku takiego należy dodawać odpowiedni sformatowane wpisy. Latex automatycznie posortuje je po nazwiskach autorów oraz do finalnego dokumentu dołączy tylko te wpisy,które posiadają odwołania w tekście! Można więc stworzyć kompletną bazę publikacji, a Latex użyje tylko to co potrzeba. Dodatkowo, dzięki użyciu pakietu **natbib** z parame-

trem **sort** (patrz preambuła dokumentu), numerki w odwołaniach również zostaną posortowane, niezależnie od kolejności podania odwołań. Przykład podano w rozdziale 7.3.

Spis rysunków

Spis tablic

8.1 Opis tabelki 24
8.2 Bardzo długa tabela (główna etykieta) 24

Spis algorytmów

1 Pseudokod prezentujący jakiś bezsensowny algorytm 23
2 Algorytm Levenshteina 23