# НАЦИОНАЛЕН ЕСЕНЕН ТУРНИР ПО ИНФОРМАЦИОННИ ТЕХНОЛОГИИ

# „ДЖОН АТАНАСОВ“

# 2023/2024

1. **ТЕМА:** TypoChecker
2. **АВТОРИ:**

Иван Христов Иванов

ЕГН: 0645268764

адрес: гр. Шумен, ул. Никола Вапцаров №6

телефон: 0885448632

е-mail: vankata384@gmail.com

училище: СУ „Сава Доброплодни“

клас: XI клас

Мартин Миленов Жечков.

ЕГН: 0651048727

адрес: гр. Шумен ул. Алеко Константинов №4,

телефон: 0895105052

е-mail: martin.jechkov@gmail.com

училище: СУ „Сава Доброплодни“

клас: XI клас

1. **РЪКОВОДИТЕЛИ:**

Тонислав Троев

телефон: 0894545844

е-mail: tonyslav\_troeff@outlook.com

длъжност: учител по информатика

Станаила Нейкова-Карагаева

телефон: 0877544182

е-mail: stanaila.neykova@gmail.com

длъжност: учител по информационни технологии и информатика

1. **РЕЗЮМЕ:**
   1. **Цел:**

Приложение, което автоматично да открива правописни грешки или непознати за българския език думи в PDF документ или изображение със сканиран текст.

Нашата разработка се отличава с възможността за визуализация на сегментите от документа, в които има допуснати неточности (или такива с не достатъчно добро качество).

* 1. **Основни етапи в реализирането на проекта:**
* Проучване за реализирани вече подобни проекти;
* Събиране на необходимата информация;
* Проучване на Microsoft Azure Document Intelligence API;
* Създаване на API:
  + Извличане на думи от българският език и техните форми от онлайн речници;
  + Имплементация на прочитането на текст от PDF документ или снимка чрез Microsoft Azure Document Intelligence API;
  + Създаване на логика за сравняване на всяка една извлечена дума с базата ни от думи в българският език;
  + Връщане на даден PDF документ или изображение с всяка една неразпознаваема или неправилна дума маркирана със съответен цвят;
  + Добавяне към резултата статистиките за подадените PDF документи и/или изображения във файлов формат JSON.
* Тестване, отстраняване на грешки и споделяне в GitHub;
* Изготвяне на материали за презентиране на проекта.
  1. **Ниво на сложност на проекта:**

Проектът изисква успешното съчетаване на различни технологии - от извличане на думи до създаване на ефективен алгоритъм за сравнение и маркиране на грешките.

* За извличане на думи, реализирахме два отделни scraper компонента, които оперират чрез Selenium. Този процес сме го разделили условно на две стъпки – извличане на основните форми и след това извличане на всички форми на думите. За да ускорим втората стъпка, сме заложили в кода възможност за многонишково изпълнение, като по този начин паралелно ще се координират много scraper инстанции.
* Разгледахме два подхода, за имплементация на алгоритъма за откриване на грешки в текста, базирани на две различни структури от данни – хеш таблица и префиксно дърво. За да изберем най-подходящия, ги сравнихме с помощта на benchmark тестове.
* За маркирането на грешки използваме две различни библиотеки – SkiaSharp (за работа с изображения) и iText7 (за работа с PDF документи). Въпреки приликите в реализацията, работата с двете библиотеки е коренно различна – изискват се различни познания за структурата на файловите формати и начина, по който можем да въвеждаме промени върху вече съществуващото съдържание. Например – при работа с изображения се използва мерната единица pixel (px) и координатите са базирани спрямо горния ляв ъгъл на снимката; при работа с PDF документи се използва мерната единица point (pt) и координатите са базирани спрямо долния ляв ъгъл на документа.
* За разчитане на текста от сканиран PDF документ или изображение, изпълняваме OCR върху съдържанието. Услугата, която използваме за целта, е Microsoft Azure Document Intelligence API.
* Разработихме Web API, като използвахме framework-а ASP.NET Core. Тъй като нашето API може да работи с много файлове едновременно, отново сме заложили възможност за многонишково изпълнение, с което целим да спестим време на нашите клиенти – няколко файла могат да бъдат обработвани едновременно, без да се изчакват. Извличаме статистики за качеството на съдържанието на документа, които след това записваме в JSON формат.

Едно от основните предизвикателства в Scraper и API под-проектите е работата с потоци от данни (Stream) и по-конкретно – работа с файлове. От гледна точка на паралелното изпълнение на задачи координираната обработка на файлове е една ключова функционалност на нашия проект.

* 1. **Логическо и функционално описание на решението**



Решението ни се състои от 4 под-проекта – API, Scraper, Benchmarks и Core. В API се съдържат всички необходими компоненти, за да работи нашето Web API, както и методи за маркиране на грешките в PDF документи и изображения. В Scraper се съдържат всички класове и методи, които способстват работата на съответния компонент, с помощта на който ние извличаме информация от речници в Интернет. В Core сме поставили интерфейси и съответните имплементации, които задвижват алгоритмите за откриване на грешки. В Benchmarks имаме няколко теста за производителността, чрез които сравняваме различните алгоритми и на база които се основава решението за това кой точно да използваме.

* 1. **Реализация − обосновка за използвани технологични средства, алгоритми, литература, програмни приложения и др.**
* Технологични средства:
  + Visual Studio 2022
  + .NET 8, C# 12
  + ASP.NET Core
  + BenchmarkDotNet
  + GitHub[[1]](https://github.com/TryAtSoftware/TypoChecker)
  + Microsoft Azure Document Intelligence .NET API
  + Selenium .NET Library
  + SkiaSharp .NET Library
  + iText7 .NET Library
* Речници:
  + Wiktionary[[2]](https://bg.wiktionary.org/wiki/Уикиречник:Начална_страница)
  + Slovored[[3]](https://slovored.com/)
  1. **Описание на приложението – как се стартира и/или инсталира, как се използва, как се поддържа**

Официалната документация на проекта заедно с инструкции за инсталация и употреба се намират в repository-то[[1]](https://github.com/TryAtSoftware/TypoChecker) в GitHub. Там също така може да откриете историята на проекта – всички commit-и pull request-и. Примерни документи може да намерите в test\_data папката.

* 1. **Заключение – какъв е основният резултат, дали има приложения до момента, какви възможности съществуват за развитие и усъвършенстване**

С помощта на нашата разработка успешно можем да маркираме грешки в сканиран текст от PDF документи или изображения. Една от основните цели, която постигнахме, беше да подсигурим производителността на различните процеси чрез оптимално разпределение на ресурсите.

Това, което отличава TypoChecker, е пряката визуализация на неточностите във файловете, с които работим. Също така основен фокус на нашата разработка е работата с неструктурирани файлови формати, каквито са изображенията и повечето (най-често сканирани) PDF документи – те изискват предварителна обработка (OCR), за да разчетем текстовото им съдържание.

Заложили сме в имплементацията възможността за персонализация на различните процеси – може да се подменя речника с думи, да се добавят нови речници, да се контролира връзката с Document Intelligence API-то и т.н. Плановете ни за бъдещо развитие включват идеи за създаване на графичен интерфейс, чрез който работата с TypoChecker API-то ще бъде по-удобна. Също така очакваме да имплементираме един цял нов слой от функционалности, стъпващи върху вече съществуващите, с които да реализираме специфични задачи за различни отрасли – медицина, изкуствен интелект, счетоводство и пр.

\*1 TypoChecker GitHub - <https://github.com/TryAtSoftware/TypoChecker>

\*2 Wiktionary - <https://bg.wiktionary.org/wiki/Уикиречник:Начална_страница>

\*3 Slovored - <https://slovored.com/>

Автор: Ръководител:

/И. Иванов/ /Т. Троев/

/М. Жечков/ /С. Нейкова-Карагаева/