

DEVELOPMENT OF A CHATBOT APPLICATION WITH ARTIFICIAL INTELLIGENCE FOR VOICE ASSISTANT AND PROGRAMMING ASSISTANT

*Assoc. Prof. PhD Dimitar Minchev, Burgas Free University, mitko@bfu.bg
Zdravko Zhelyazkov, graduate student, zdravko.zhel@gmail.com*

Abstract: This publication examines a chatbot application that interacts with artificial intelligence and has built-in voice communication, which reduces the time programmers spend searching for information on the internet, so that their time and focus on writing code is not wasted.

Keywords: Python, PyCharm IDE, Google Generative AI, Flask, SQLite, SQL Alchemy, REST API

РАЗРАБОТВАНЕ НА ЧАТ-БОТ ПРИЛОЖЕНИЕ С ИЗКУСТВЕН ИНТЕЛЕКТ ЗА ГЛАСОВ АСИСТЕНТ И ПОМОЩНИК ПО ПРОГРАМИРАНЕ

*доц. д-р Димитър Минчев, Бургаски свободен университет, mitko@bfu.bg
Здравко Желязков, дипломант, zdravko.zhel@gmail.com*

Абстракт: Настоящата публикация разглежда чат-бот приложение, контактуващо с изкуствен интелект и с вградена гласова комуникация, което да намали времето на програмиста прекарано в търсене на информация из интернет, за да може неговото време и фокусът му върху писането на кода да не се губят..

Ключови думи: Python, PyCharm IDE, Google Generative AI, Flask, SQLite, SQL Alchemy, REST API.

Въведение

Актуална тема през последните години е изкуствения интелект, който се превърна в една неразделна част от съвременните технологии и нашето ежедневие. Чат-бот приложенията вече не са само част от научната фантастика, която виждаме по телевизията, а и от реалния живот на хората. Инструменти, които улесняват комуникацията, достъпа до информация и изпълнението на различни задачи.

Въпреки напредъка в разработката на тези чат-бот приложения, все още липсва в много от наличните асистенти подпомагането на специфични дейности като програмирането. За начинаещи програмисти, липсата на бърза, изпълнена с контекст помощ и разбираемо обяснение, често е сериозна пречка при усвояването на нови знания. Затова възниква необходимостта от интелигентна система, която комбинира чат-бот приложенията с гласово взаимодействие, за една по-ефективна асистенция при писане на код.

Избор на използвани технологии

Изборът на технологиите е направен на базата на техните функционални възможности, съвместимост с целите на проекта, лекота на използване, налична документация и поддръжка на общността. Всяка технология ще бъде разгледана чрез описание, роля в проекта и причини да бъде избрана.

Python [1] е високоефективен, обектноориентиран програмен език, създаден от Гуидо ван Росум през 90-те години, отличаващ се с четим синтаксис и богата екосистема от библиотеки. Той осигурява бърза разработка и поддръжка на големи приложения, включително интеграция с API и изкуствен интелект, като е особено подходящ за изграждане на сървърни решения и потребителски интерфейси, благодарение на широката си съвместимост и документация.

PyCharm [2] е интегрирана среда за разработка на Python, създадена от JetBrains върху платформата IntelliJ, която поддържа Windows, macOS и Linux. Тя предлага анализ на кода, графичен дебъгер, интегрирани тестове, интеграция със системи за контрол на версиите и уеб разработки с Django. Използвана за разработка, тестване и отстраняване на грешки в Python кода, като осигурява лесна навигация между сървърната и клиентската част. Средата предоставя функции като автоматично довършване, рефакторизиране, инспекция на кода в реално време и удобен, персонализируем интерфейс, което значително повишава продуктивността и е основна причина за избора ѝ за този проект.

Gemini [3] е серия мултимодални големи езикови модели, разработени от Google DeepMind като наследник на PaLM, които са в основата на модерната генеративна AI архитектура на Google. В рамките на проекта Gemini служи като главния езиков модел за чат-бота, обработвайки различни типове данни, но основно генерира текст, разпознава намерения и отговаря на въпроси. Изборът на този модел е обусловен от безплатния достъп чрез API по време на разработка, дълбокото разбиране на контекста, високата точност при разпознаване на намерения, поддръжката на много езици и наличието на Python библиотека за лесна интеграция с приложението.

Flask [4] е лек и гъвкав микро уеб фреймуърк на Python, който предоставя основните инструменти за създаване на уеб сайтове и API без допълнителни вградени компоненти, но позволява лесно разширяване чрез допълнения. Той изпълнява ролята на бекенд, реализирайки комуникацията между потребителския интерфейс и изкуствения интелект, обработвайки заявки, управлявайки връзката с езиковия модел и базата данни. Основната причина за избора му е доброто познаване и опит на дипломанта с Flask, както и неговата лекота на използване, интуитивен синтаксис, разширяемост, бърза интеграция с API и широка документация.

SQLite [5] е лека релационна база данни с отворен код, реализирана като библиотека, която съхранява цялата информация в един файл, без необходимост от сървър или мрежова връзка. Използва се широко в продукти на компании като Adobe, Apple, Mozilla и Google, а в разглеждания проект служи за съхраняване на историята на разговорите и потребителските данни. Основните ѝ предимства са лесната интеграция, подходяща е за малки и средни проекти, напълно безплатна е и позволява локална работа без интернет връзка, като Python има вграден модул за работа с нея (“sqlite3”).

SQLAlchemy [6] е библиотека за Python, която осигурява ORM функционалност и позволява обектно-ориентирана работа с релационни бази данни, без необходимост от

писане на ръчен SQL код. В рамките на проекта тя се използва за дефиниране на таблици чрез Python класове, за извършване на CRUD операции и за управление на връзката между чат-бот приложението (Flask) и базата SQLite. Основните ѝ предимства са лесната интеграция с Flask, възможността за бъдеща миграция към други бази данни без промяна на логиката, както и повишената сигурност благодарение на автоматичното генериране на параметризирани заявки.

Kivy [7] е мултиплатформена, отворена библиотека за създаване на графични потребителски интерфейси с Python, подходяща както за десктоп, така и за мобилни устройства благодарение на единната кодова база и собствен рендеринг механизъм. Kivy е използвана за изграждане на интерфейс за чат-бота, включващ бутони, полета за въвеждане и визуализиране на диалога, като основните ѝ предимства са възможността за работа на различни платформи, разработка на целия UI на Python, лесна интеграция с логиката на приложението и гъвкав контрол върху дизайна и визуалните елементи, включително поддръжка на жестове и анимации.

Google Text-To-Speech (TTS) и **Google Speech-To-Text (STT)** са облачни AI услуги, част от Google Cloud Platform, които осигуряват надеждно преобразуване на текст в реч и обратно, с висока точност и поддръжка на български език. В разработеното решение TTS превръща отговорите на чат-бота в естествено звучаща реч, а STT разпознава речта на клиента и я превръща в текст за обработка. Технологиите са избрани заради лесната интеграция с Python, възможностите за персонализация и естественото звучене на синтезираната реч.

Pygame [8] е мултиплатформен набор от Python библиотеки, базиран на SDL, използван основно за създаване на видео игри и мултимедийни приложения, включително графични интерфейси. В рамките на разработеното приложение, Pygame служи за възпроизвеждане на аудио, генерирано от Google Text-To-Speech, като аудио се проиграва сякаш е стартирана мини игра, тъй като не е намерен по-ефикасен начин за реализиране на този ефект в използваната среда.

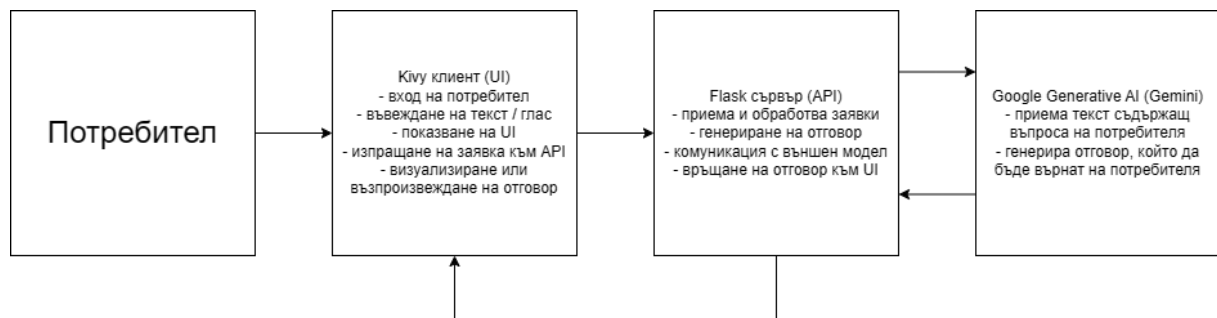
JSON Web Token (JWT) [9] е JSON-базиран отворен стандарт за създаване на компактни и подписани „жетони“, които съдържат определени твърдения и се използват основно за удостоверяване и идентификация на потребители, особено при уеб приложения и SSO. В разглежданото приложение JWT ограничава достъпа до API крайни точки във Flask, като при успешно логване сървърът генерира токен, който клиентът съхранява и използва за достъп до защитени ресурси. JWT гарантира сигурност чрез цифров подпис, не изисква поддръжане на сесии на сървъра и се интегрира лесно с Python и Flask, като опростява архитектурата и улеснява управлението на сесиите.

bcrypt [10] е функция за хеширане на пароли, създадена с цел висока сигурност чрез използване на сол и адаптивен алгоритъм, който позволява увеличаване на броя итерации за защита срещу brute force атаки. Тя е внедрена в множество програмни езици, включително Python, и се използва в разработеното приложение за хеширане на паролите при регистрация и логване на потребители, като основното ѝ предимство е устойчивостта срещу различни видове атаки и възможността за лесна интеграция и конфигуриране.

Разработка на платформата

Проектирана и реализирана бе собствена архитектура, която комбинира REST-базирана клиент-сървър комуникация с локално или онлайн изпълнение на гласов и текстов

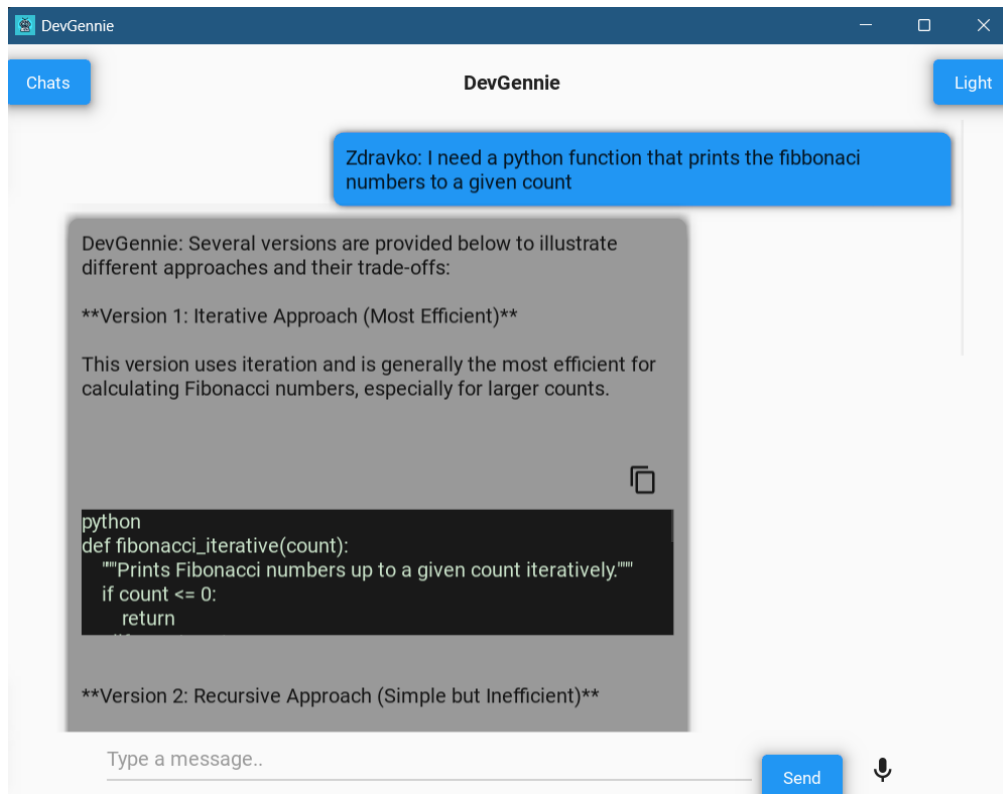
изкуствен интелект. Клиентската част е разработена с езика Python и модулът Kivy, за да може приложението да е мултиплатформено и с потребителски интерфейс балансиран между хубав дизайн и лесен за работа за потребителя. Сървърната част е отново разработена с езика Python, но с модулите Flask, SQLite и Google Generative AI. Тук се извършват всички функции на програмата като: регистриране, влизане, водене на чат с изкуствения интелект, запазване на информацията на потребителя и неговите чатове в база данни за по-нататъчна употреба. Модулите, които са използвани за генериране на текст от реч и обратно са разработени от Google, тъй като те са едни от малкото, които поддържат български език, а интеграцията с изкуствения интелект се случва чрез API заявка до изкуствения интелект на Google – Gemini.



Фиг. 1. Схема на архитектурата на разглежданото решение.

Архитектурата на разглежданото решение е по-лека спрямо индустриалните микросървисно облачни системи, което я прави по-подходяща за персонални или образователни цели.

Екранът, в който се водят самите чатове с изкуствения интелект (фиг. 2). Основната част на този екран е поле, от което потребителя може да въвежда текстово съобщение, което да изпрати към изкуствения интелект, бутон, който изпраща съобщението при натискане от клиента към сървъра и бутон, който започва да слуша за глас при натискане. Освен тези елементи имаме и поле, което служи за визуализиране на отговорите и бутон, който ни отваря изскачащо меню, което ни позволява да видим и отворим всички чатове, които сме започнали, да преминем към екрана за настройване на профила на потребителя чрез бутон, който ще разгледаме в следващата подточка или да излезем от програмата със следващия бутон от менюто, както има и бутон, с който да стартираме изцяло нов чат.



Фиг. 2. Как приложението генерира отговор.

Използването на GitHub за управление на проекта носи множество ползи. Ето някои от конкретните начини, по които качването на всички промени в GitHub допринася за успешното управление и развитие на един проект. Качването на всички промени в GitHub означава, че имаме детайлна история на развитието на проекта. Това включва история на версиите и всеки комит може да бъде описан с коментар, който обяснява какви промени са направени и защо. Другото предимство е, че кода е безопасно съхранен в облака, което ни защитава от загуба на данни при проблеми с локалното съхранение. Също така лесно можем да се върнем към предишна версия на проекта, ако новите промени предизвикат проблеми.

Всички промени по време на разработката на платформата са качени в github репозитори. Като качваме всички промени в GitHub, ние не само осигуряваме централизирано съхранение на кода, но и използваме пълния потенциал на Github за подобряване на качеството и ефективността на проекта.

Използването на GitHub в проекта е отличен избор, тъй като осигурява ефективно управление на версиите, гарантира сигурността на кода чрез резервни копия, позволява лесна интеграция с инструменти за автоматизация и непрекъсната интеграция, и предоставя видимост и възможности за обратна връзка от общността. Това води до по-добро организиране, проследяване и подобряване на качеството на проекта.

Изследване и анализ на разработката

Тъй като изкуствения интелект, не е създаден специфично за разработката, а е готов голям езиков модел, който се достъпва чрез API заявки, това за което трябва да бъде тестван са надеждността, точността и практическата стойност на отговорите, които изкуствения интелект на компанията Google – Gemini генерира.

	Релевантност (0-5)	Коректност (0-5)	Яснота (0-5)	Последователност (0-5)	Устойчивост (0-5)
Базови въпроси	5	5	5	5	5
По-сложни/многослойни въпроси	5	4	3	3	4
Нееднозначни въпроси	3	5	4	3	4
Грешни/Шумни въпроси	4	3	1	4	3
Контекстни въпроси	5	4	5	5	4
Средна оценка на всеки критерии	22/25	21/25	18/25	20/25	20/25
Цялостна оценка на изкуствения интелект			20/25 = 80%		

Табл. 1. Сравнение на различните категории въпроси спрямо различните критерии.

Самата част от приложението е сървър под формата на API разработен с модула Flask, който приема заявките от потребителския интерфейс (клиента), обработва ги и връща някакви отговори. За функцията на водене на чат с изкуствения интелект, ще направим няколко теста, с въпроси с различна сложност, за да определим някакво средно време за изпълнение.

	Получен код от сървъра	Очакван код от сървъра	Време за изпълнение (ms)
/register	200	200	0.4649360179901123
/login	200	200	0.4484076499938965
/delete	200	200	0.04807782173156738
/delete chat	200	200	0.0020139217376708984
/get info	200	200	0.0014998912811279297
/update info	200	200	0.6237411499023438
/get all	200	200	0.0020172595977783203
/chat	200	200	3.816447814305623

Табл. 2. Таблица с данните от тестването на сървъра.

Можем да пресметнем, че времето за изпълнение на всички операции една след друга е 5.407141527 ms, което в секунди е 0.0054071415 s. Като най-често, ще ни се налага да изпълняваме само операцията за водене на чат с изкуствения интелект, която статистически отнема най-много време, но дори и тя се изпълнява за по-малко от секунда.

На базата на тези изводи можем да стигнем до заключението, че сървърната част работи достатъчно правилно и достатъчно бързо – нито една от функциите на сървъра не се обърка, нито се провали в изпълнението на задачата си, а времето, за което всички се изпълни беше по-малко от секунда.

Заключение

Представеното в настоящата публикация приложение доказва възможностите на съвременните технологии за изграждане на интелигентни чат-бот системи. То съчетава различни, но взаимно свързани компоненти като изкуствен интелект, база данни,

клиент-сървър архитектура и гласово взаимодействие, като предоставя завършено решение с функционалната възможност за употреба в реална среда. На базата на всички резултати и направени анализи, можем да считаме че основата за бъдещо развитие и усъвършенстване е стабилна.

Използвана литература

- [1]. Wikipedia contributors, Питон, Python, Wikipedia, [Онлайн], Наличен на: <https://bg.wikipedia.org/wiki/Python>, [достъп 03.08.2025]
- [2]. Wikipedia contributors, PyCharm, Wikipedia, [Онлайн], Наличен на: <https://en.wikipedia.org/wiki/PyCharm>, [достъп: 03.08.2025]
- [3]. Wikipedia contributors, Gemini (language model), Wikipedia, [Онлайн], Наличен на: [https://en.wikipedia.org/wiki/Gemini_\(language_model\)](https://en.wikipedia.org/wiki/Gemini_(language_model)), [достъп 03.08.2025]
- [4]. Wikipedia contributors, Flask (web framework), Wikipedia, [Онлайн], Наличен на: [https://en.wikipedia.org/wiki/Flask_\(web_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework)), [достъп 03.08.2025]
- [5]. Wikipedia contributors, SQLite, Wikipedia, [Онлайн], Наличен на: <https://bg.wikipedia.org/wiki/SQLite>, [достъп 03.08.2025]
- [6]. Wikipedia contributors, SQLAlchemy, Wikipedia, [Онлайн], Наличен на: <https://en.wikipedia.org/wiki/SQLAlchemy>, [достъп 03.08.2025]
- [7]. Wikipedia contributors, Kivy (framework), Wikipedia, [Онлайн], Наличен на: [https://en.wikipedia.org/wiki/Kivy_\(framework\)](https://en.wikipedia.org/wiki/Kivy_(framework)), [достъп 03.08.2025]
- [8]. Wikipedia contributors, Pygame, Wikipedia, [Онлайн], Наличен на: <https://en.wikipedia.org/wiki/Pygame>, [достъп 03.08.2025]
- [9]. Wikipedia contributors, JSON Web Token, Wikipedia, [Онлайн], Наличен на: https://bg.wikipedia.org/wiki/JSON_Web-Token, [достъп 03.08.2025]
- [10]. Wikipedia contributors, Bcrypt, Wikipedia, [Онлайн], Наличен на: <https://en.wikipedia.org/wiki/Bcrypt>, [достъп 03.08.2025]