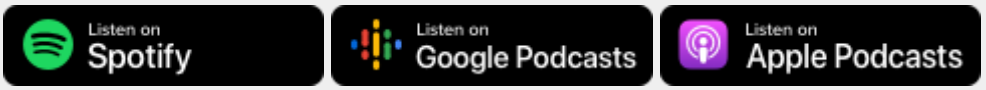
Pierwsze kroki w IT

PODCAST



Narzędzia AI w pracy programisty: ChatGPT, GitHub Copilot, Midjourney i inne – część 1

Gość: Karol Horosin

Wszystkie polecane materiały i linki znajdziesz na stronie odcinka:

https://devmentor.pl/b/narzedzia-ai-w-pracy-programisty-chatgpt-github-copilot-midjourney-i-inne-czesc-1

DEVMENTOR.PL
YOUR CODE COACH

Dziś moim gościem jest Karol Horosin. Karol opowie nam o wykorzystaniu narzędzi sztucznej inteligencji w pracy programisty. Karolu, dziękuję, że przyjąłeś moje zaproszenie na rozmowę.

Cześć, Mateusz, dzięki za zaproszenie. Mamy bardzo ciekawy temat przed sobą, więc mam nadzieję, że powiemy coś ciekawego dla Twoich słuchaczy.

Myślę, że będą same ciekawe rzeczy. Może zaczniemy po prostu od Twojej historii. Powiedz nam coś więcej o sobie i co łączy Cię z branżą IT.

Moje najmocniejsze powiązanie z pracą w IT w tej chwili to to, że pracuję jako engineering manager w start-upie, który zajmuje się cyberbezpieczeństwem. Firma nazywa się Netacea i zajmujemy się wykrywaniem zagrożeń ataków na różne systemy za pomocą AI – czyli trochę inne AI niż będziemy dzisiaj obgadywać, ale też, powiedzmy, pokrewna branża.

Nie mam bardzo *hands on* programowania w pracy w tej chwili. Ja zajmuję się zarządzaniem, od czasu do czasu zaprojektuję jakąś architekturę, zajmuję się produktem. Ale jak przychodzi do napisania czegoś mniej krytycznego, czegoś, gdzie nie mamy deadline'ów, to też napiszę kawałek kodu. Poza tym robię dużo rzeczy poza tym, więc jestem na bieżąco ze wszystkimi trendami i z za dużą ilością frameworków front-endowych.

No dobrze, to może powiedz nam jeszcze, jak zacząłeś wchodzić do tej branży, bo wspominałeś, że programowałeś zanim wszedłeś w temat taki bardziej menedżerski.

O tak, to jest długa historia. Trochę za długa, żeby opowiadać całą. Ale jak byłem mały, byłem w podstawówce, zastanawiałem się, jak działa Internet i to mnie wysłało na fora o tym: o HTML-u, o tym, jak działają serwery. Wtedy znalazłem sobie kolegę z klasy, który miał pomysł na content na stronę o grze Hitman – i zrobiliśmy naszą pierwszą stronę World of Hitman, zahostowaliśmy na darmowym hostingu. To były piękne czasy CSS-a, który działał zupełnie inaczej w każdej przeglądarce, więc teraz wszystkie narzekania młodych front-end developerów są dla mnie urocze. Od tego się gdzieś tam zaczęło.

Później kolejne strony internetowe, małe projekciki aż do jakiegoś pierwszego zlecenia, gdzie ktoś wiedział, że umiem pisać strony internetowe, więc podszedł do mnie: „słuchaj, masz tu 500 złotych, zrób mi stronę, która przypomina 9gaga". No i od takiego zlecenia się zaczęło. A później to już staż, junior developer, senior developer, kolejne projekty, kolejne branże i gdzieś przez to organizowanie, uczenie innych zawędrowałem do takiej bardziej liderskiej pozycji.

No dobrze, to tak z ciekawości: może chcesz podać adres tej strony z Hitmanem? Możemy sobie sprawdzić, bo jest taka opcja.

Próbowałem, próbowałem znaleźć na archive.org nawet przed tym podcastem, ale nie znalazłem. Nie jestem w stanie znaleźć tych plików. Mam je gdzieś na zewnętrznym dysku twardym, ale znalezienie tego zajmie mi pewnie chwilę.

No to odkopiemy je następnym razem.

A dzisiaj zajmijmy się może naszą sztuczną inteligencją. To takie pytanie na początek na rozgrzewkę: czym są narzędzia wykorzystujące sztuczną inteligencję i w jaki sposób różnią się od tego, co było dostępne w ciągu dwóch-trzech lat?

Na pewno różni się to, jak dużo się o nich mówi. Spotkałem się nawet wczoraj z taką małą analizą porównującą hype narzędzi AI do hype'u z blockchainem. Ponieważ media mainstreamowe wskoczyły na temat, bardzo głośno mówią: „Czy AI zastąpi cię w pracy? Czy to koniec jakichś tam zawodów?”. Wiesz, nagłówki, nagłówki, nagłówki. Ale z takiego technicznego punktu widzenia i tego, co my możemy wykorzystać, to mam wrażenie, że zmienił się trochę sposób myślenia o tych narzędziach i o AI ogólnie.

Mieliśmy takie wyspecjalizowane narzędzia jak na przykład, nie wiem… Ja korzystam z Grammarly do poprawiania tego, co piszę. Mieliśmy takie oczywiste narzędzia AI w stylu: YouTube miał automatyczne transkrypcje, mogliśmy mówić do naszych telefonów i były wyspecjalizowane silniki do analizy danych, ale to nie było dostępne dla konsumentów.

Mieliśmy filmy, w których się nas straszy AI, i jeszcze była ta sprawa, nie wiem, czy pamiętasz, jeszcze dwa-trzy lata temu była wielka afera, że nasze dane są wykorzystywane przez wielkie korporacje: AI jest używane do zrozumienia każdego naszego kroku, Facebook wie lepiej, czego chcemy niż my sami. Ale w tej chwili w ogóle o tym się nie mówi.

W tej chwili AI jest wielką szansą, przyjacielem, rewolucją, a nie szpiegiem.

To więc na pewno się zmieniło. No i oczywiście mamy tę dużą technologię, z którą teraz mamy do czynienia i o której dzisiaj powiemy więcej, czyli Large Language Models – duże modele językowe (LLM). Stały się one bardzo popularne i otworzyły bardzo dużo use case'ów. Myślę, że to jest taka główna zmiana. Wiesz, mógłbym o tym mówić w nieskończoność, bo interesuję się też historią tego. Te modele już były dostępne dwa-trzy lata temu. Same oryginalne badania na temat technologii wyszły niedługo wcześniej.

W tej chwili tak naprawdę duże firmy informatyczne mocno zainwestowały w to, żeby włożyć te narzędzia w nasze ręce. Niektórzy by powiedzieli, że jedyna różnica, jaka jest, to duża ilość pieniędzy i duża ilość kart graficznych. W tej chwili każdy może sobie wejść, zalogować się na chat.openai.com i pogadać z takim AI. Wcześniej to nie było możliwe, bo te modele są bardzo drogie do uruchomienia i nie zadziałają na żadnej karcie graficznej, którą może kupić zwykły śmiertelnik przez Internet, nawet jeśli ma dużo pieniędzy. Dostępność się więc zwiększyła. I oczywiście technologia trochę poszła do przodu.

Jak tak opowiadałeś o tej mocy obliczeniowej… W zasadzie nie mieliśmy na ten temat rozmawiać. Ciekawy jestem, czy myślałeś w ogóle o tym, co będzie, jeżeli faktycznie powstanie komputer kwantowy? Jak on zmieni całą sytuację? Czy w ogóle zastanawiałeś się nad tym, czy odpuszczamy temat?

Powiem Ci, że chwilę się zastanawiałem, ale jestem daleko od tego, jak będą wyglądały kwantowe sieci neuronowe. Muszę Ci jednak przyznać,

że jeszcze na studiach miałem taki cudowny przedmiot obliczenia na komputerach kwantowych, gdzie projektowaliśmy te algorytmy i potencjalnie widziałem projekty odpalenia sieci neuronowych na komputerach kwantowych, ale jest to bardzo teoretyczne.

W tej chwili nie mogę Ci powiedzieć, czy to sprawi, że nagle będziemy mieli do czynienia z kwantowym AI i z rzeczami, które wykonają wielkie skoki. Myślę, że jeszcze jesteśmy od tego daleko i branża o tym dużo nie mówi, więc ja też nie mam nic odkrywczego w tej chwili na ten temat.

Jasne, to ja może tylko tak dodam od siebie, że jak mieliśmy ten skok z ChatGPT 3.5 do 4, to pokazywali taką małą kuleczkę i taką wielką kulę. I to jest różnica między jednym a drugim. Ponoć jeśli chodzi o moc obliczeniową komputerów kwantowych będzie ona X razy większa niż ten skok z ChatGPT 3.5 do 4, więc jeżeli faktycznie by się to udało osiągnąć, to może być ciekawie, ale może jeszcze upłynąć trochę czasu.

Tak, ale to jest bardzo ciekawy temat. Tylko z wypowiedzi choćby liderów firmy OpenAI, która dała nam ChatGPT, wynika, że oni w tej chwili zaczynają trochę odchodzić od tego rozmiaru obliczeń. W ostatnim wywiadzie, którego słuchałem – Sama Altmana, CEO OpenAI – wspominają, że już nie będą szli w rozmiar, że kolejne rewolucje i poprawy w tej technologii dużych modeli językowych będą się brały z lepszych danych i z nowych osiągnięć inżynieryjnych i naukowych. Może dlatego, że nie ma sprzętu, który jest to w stanie odpalić, a może dlatego, że gdzieś powyżej pewnego rozmiaru to już ma mniejszy sens. Ale zobaczymy, bo tak naprawdę ten temat otworzył tyle nowych

ścieżek, że są niewyeksplorowane. Zaskoczyłeś mnie tym pytaniem. Zobaczmy co się stanie.

A danych jest coraz więcej. Patrząc po tym, jak to wszystko wygląda i ilu ludzi korzysta, to jest z czego wybierać.

To prawda. Chociaż są też pewne obawy, że Internet, w którym istnieją teraz te modele AI i ten cały content… Czy AI uczący się na contencie AI to jest coś, co chcemy, czy w pewnym momencie po prostu ten progres zastopuje, bo będzie tyle kontentu AI w Internecie, że będziemy karmili modele AI outputem AI i one nie będą w stanie stać się lepsze? Ale zobaczymy.

No to jeszcze powiem taką ciekawostkę, że ostatnio widziałem na LinkedInie informację o tym, że Google zaczyna banować blogi, które są generowane na podstawie danych z AI, więc może będą w stanie sobie z tym poradzić. Za dużo wątków chyba na początek.

Ale jak widzisz i pewnie widzą słuchający nas, temat jest rozległy. My skupimy się nad tym, jak te rzeczy mogą się przydać, więc myślę, że poza takim inspirującym aspektem, który możemy tutaj zarysować, powiemy też dużo praktycznych rzeczy.

Tak, więc ja od razu przechodzę do kolejnego pytania.

Jakie narzędzia przydatne dla programistów ostatnio wysunęły się według Ciebie na prowadzenie? Czy możesz je krótko omówić i powiedzieć, w jaki sposób ułatwiają nam pracę?

Jasne, postaram się tutaj nie doktoryzować nad tym, co to znaczy dla programistów, bo temat jest szeroki. To, co wchodzi w zakres naszej pracy, jest dyskutowane bardzo, bardzo szeroko. Co powinien robić junior? Co powinien robić mid? Co powinien robić regular? Ale tak naprawdę jest dużo interdyscyplinarnych aspektów tego, czego potrzebujemy: jako programista muszę napisać dokumentację, muszę też czasem coś wymyślić, napisać maila do klienta, do managera, wyjaśniając to, co robię. Więc do tych zastosowań mamy różne narzędzia.

Obecnie takim, o którym pewnie każdy słyszał, jest ChatGPT. ChatGPT to jest narzędzie OpenAI oparte na dużym modelu językowym. Jego zaletą jest generowanie tekstu – mamy to dostarczone przez super prosty interfejs chatowy.

Jakie zastosowanie tam widzę? Samo programowanie, pisanie tekstu, generowanie prezentacji, testowanie swoich odpowiedzi rekrutacyjnych, code review, pisanie testów – wszystko, co jest związane z tekstem. Pewnie ChatGPT może nam w tym pomóc. Jest to więc na pewno narzędzie, na które warto patrzeć, ale są też alternatywne rozwiązania od konkurencji. Mamy w tej chwili Bing Chat, czyli w zasadzie opakowany przez Microsoft ChatGPT, który ma już dostęp do Internetu w darmowej wersji, z czego ja czasem korzystam, jeśli potrzebuję podsumowania jakiegoś aspektu. Kiedy byś tam wpisał „napisz mi dwa akapity o Mateuszu Bogolubowie”, dostałbyś tę informację, której nie dostałbyś pewnie w ChatGPT albo byłaby ona opisana w nieprecyzyjny sposób, z nieaktualnymi informacjami. Więc Bing Chat zdecydowanie.

No i mamy Barda od Google'a, czyli te takie interfejsy chatowe. Poza tym na tej technologii modeli językowych od paru lat mamy dostępny GitHub Copilot, czyli pomocnika w naszym IDE, w naszym środowisku do programowania. Według mnie w zastosowaniach, w których ja z niego korzystałem, to jest taka trochę lepsze autouzupełnianie – podpowiada nam, jak się skończy linijka, czasem w taki bardzo straszny sposób: wciskasz Enter, a on się wydaje już wiedzieć, co chcesz napisać. Jest to więc zdecydowanie ułatwienie i przyspieszenie. Bardzo często też nie musimy znać bibliotek, których musimy użyć, bo jeśli zaczynamy pisać funkcję, która ma coś robić, on nam podpowiada funkcje biblioteczne, których nie musimy już googlować. Jest to więc przyspieszenie, ale nie zastępuje nas bardzo.

Ja bym tutaj może dodał jedną rzecz. Osoby, z którymi współpracuję, korzystają właśnie np. z GitHub Copilota i mówią (ja też to widzę), że generuje on taki kod, powiedziałbym, mało optymalny. Nawet nie chodzi o szybkość, tylko zamiast zrobić odpowiednią strukturę danych i wykorzystać pętle do wygenerowania odpowiedniej formy, to on po prostu zrobi Ci obiekt, który zawiera milion tablic. I tak to będzie.

Warto więc o tym wspomnieć, że – przynajmniej w mojej ocenie – nie jest to idealne na takim poziomie, który byśmy oddawali na rekrutacji czy w pracy. Nie wiem, czy też to widzisz, czy w Twoim przypadku tak ma.

Zdecydowanie. Ludzie i te sensacyjne artykuły w przypadku Copilota pisały „Czy to koniec pracy programistów?" albo „AI Cię zastąpi". Ja

korzystam z tego Copilota i widzę: OK. Nie. To napiszę sam, to przeskoczę. To mi pomaga, ale jednak zrobiłbym to inaczej.

Trzeba pamiętać, skąd się bierze intuicja czy wiedza w tych narzędziach. To jest wszystko, co było na GitHubie, wszystko, co było w Internecie, wszystko, co było na Stack Overflow. Ten kod nie zawsze jest dobry, nie zawsze jest optymalny.

Ale w przypadku programowania z ChatGPT, kiedy piszemy dokładnie, co chcemy, zwykle dostaniemy poprawne rozwiązanie. Niekoniecznie optymalne, tak jak mówisz. Nie powiedziałbym więc, że ChatGPT jest ekspertem w programowaniu, ale na pewno może być całkiem okej programistą.

To może tylko dodam jedną rzecz, bo też w zasadzie nagrywałem odcinek podcastu na temat sieci neuronowych. To tutaj ważna informacja: jakość danych wchodzących daje wynik. Czyli jeżeli są te dane, na podstawie których uczy się ta sztuczna inteligencja, są mega, to wtedy taki sam jest wynik. Wiecie, większość rzeczy na GitHubie nie jest idealnych, idealna jest mniejsza ilość, co powoduje, że ten kod, który jest generowany przez sztuczną inteligencję, też jest średniej jakości. O tym trzeba zawsze pamiętać.

Dokładnie tak, jak mówisz. To powiedzenie w wersji angielskiej brzmi mniej przyjaźnie, czyli „garbage in, garbage out". „Śmieci wchodzą i śmieci wychodzą". Jak będziemy później mówili o zastosowaniu tych narzędzi, to to samo ma zastosowanie do tego, jak my pracujemy z tymi modelami językowymi: czy z chatem, czy z generowaniem grafik, czy z

czymkolwiek innym. Ten input, który damy modelowi, opis zadania, które ma być dla nas wykonane, ma bardzo duży wpływ na to, co dostaniemy.

Jeśli wpisujemy prośbę do ChatGPT, żądanie, zadanie: „napisz mi funkcję, która robi to” i napiszemy to tylko jednym zdaniem, to nie powinniśmy się dziwić, że wynik będzie nieprecyzyjny. To jest zupełnie inny aspekt niż przy trenowaniu, ale nie możemy się dziwić, że wynik będzie nieprecyzyjny, bo nie daliśmy więcej instrukcji.

Z kolei jak damy dużo instrukcji, możemy się spodziewać, że output, który dostaniemy, będzie zawierał dużą część z nich. Polecam więc wkładać wysiłek w pracę z AI, tak jakbyśmy tłumaczyli człowiekowi, samemu sobie, co ma się stać. Spodziewanie się, że dostaniemy idealnie to, co chcemy, bez powiedzenia, czego chcemy, nie działa ani w relacjach międzyludzkich ani w pracy z AI.

A co do relacji międzyludzkich to ja to zawsze porównuję do tego, jak prosimy kogoś na forum, żeby nam pomógł, albo na czacie. Jeżeli podrzucimy: „Wiecie co, mam taki błąd. Co to może być?”, to dostaniemy pewnie hejt albo coś w stylu: „no jak mamy ci pomóc, jak nie znamy kodu?” Jeżeli wrzucimy kod, napiszemy, co chcieliśmy osiągnąć, co się nie udało, co spróbowaliśmy, to tak samo inny efekt otrzymamy na takim forum. Więc podobnie jest tutaj z rozmową ze sztuczną inteligencją.

Tak. W przypadku ludzi co prawda pamiętajmy o tym, żeby najlepiej kod udostępnić przez jakieś narzędzie, które ma kolorowanie składni i umożliwia kopiowanie kodu, bo wklejanie screenshota małych literek niekoniecznie wzbudzi sympatię. Ale do ChatGPT możemy wkleić dość

niesformatowane rzeczy i nie ma to dla niego aż takiego znaczenia, więc mała różnica.

To ja muszę się przyznać, że byłem zdziwiony. Napisałem sobie bota, który korzysta z ChatGPT i na Slacku pomaga moim podopiecznym. Byłem zdziwiony, że jak wrzuciłem do ChatGPT kod, który używa backticków, żeby oznaczyć, że to jest kod, to on w odpowiedzi też mi zwracał kod w ten sam sposób. Nawet nie musiałem o tym mówić, tylko on to zrobił sam. To mnie akurat zdziwiło, że był w stanie sobie tak wywnioskować: że skoro ja mu tak wrzuciłem, to on mi też tak odpisze.

Zgadza się. Ja w ogóle raz wykorzystałem ChatGPT do poskładania mi tekstu, który w zasadzie był nie do odczytania, co jest ciekawym eksperymentem. W nowych komputerach Apple'a możemy otworzyć jakiekolwiek zdjęcie. Jeśli jest na nich tekst, możemy skopiować cały tekst. Jest automatyczne rozpoznawanie tekstu. Tylko że taki tekst, jak go skopiujemy, zwykle jest nieustrukturyzowany. Kolejność słów się miesza, linijki się mieszają. Wziąłem taki tekst, wkleiłem totalnie pomieszany tekst do ChatGPT i prosiłem: pomóż mi poskładać te puzzle, złożyć tekst z tych porozrzucanych słów. I co jest ciekawe, dostałem wynik, który był bardzo podobny do tego, co było rzeczywiście na kartce.

To może trochę mówić o tym, jak działają te modele językowe, o tym, że sekwencja nie ma aż takiego znaczenia. I nie wiem, czy widziałeś, czasem są na LinkedInie takie posty, czy takie eksperymenty, ćwiczenia umysłowe, gdzie wrzuca się tekst, ale tylko pierwsza i ostatnia litera wyrazu jest dobrze, wszystko pomiędzy jest pomieszane. Dla ludzi to nie

ma znaczenia, bo czytamy słowami. Tak samo AI, z którymi pracujemy, też ma pewne swoje charakterystyki i np. niekoniecznie wszystkie błędy mają znaczenie dla tego modelu językowego.

No dobra, stanęliśmy na tych wersjach tekstowych, ale może coś powiesz o innych wersjach sztucznej inteligencji, która nam może pomóc?

Tak, ciężko nie mówić o tekście, bo to jest to, co wzbudza teraz dużą ekscytację ludzi. Ale mamy też narzędzia graficzne. One do niedawna służyły głównie do zabawy, do tworzenia sztuki. A działo się tak dlatego, że sztuka dopuszcza błędy, a błędy są wręcz pożądane. Błędy to wyjątkowość. Błędy to właśnie to, co czasem sprawia, że coś jest inne i artystyczne. Dlatego pierwszym, co tak naprawdę zaimponowało ludziom, było generowanie obrazów.

W tej chwili takim wiodącym narzędziem jest Midjourney, więc jeśli ktoś ze słuchaczy chciałby sobie wygenerować obrazek, zobaczyć ulubioną gwiazdę w innym świetle i w innej sytuacji, wyobrażać sobie rzeczy, które mogłyby się stać, to Midjourney generuje bardzo realistyczne grafiki. Może generować anime, może generować ikony, może generować bardzo dużą ilość dobrej jakości grafik, ale też oczywiście robi czasem dziwactwa. Jeśli pobawimy się generowaniem dłoni, to możemy zobaczyć, że dłonie bardzo często wyglądają dość wynaturzenie.

Mamy stable diffusion, czyli taki oryginalny, imponujący model do generowania obrazków. On jest open-source'owy, więc jeśli chcielibyśmy go wykorzystać we własnym projekcie, możemy to zrobić. Możemy też

trenować ten model samemu do swoich zastosowań, o czym jeszcze, mam nadzieję, będę mógł powiedzieć później.

Z takich narzędzi, które ja polecam i jedynych darmowych dostępnych obecnie, jest Bing Image Generator. Jest to silnik do generowania obrazków oparty na OpenAI DALL·E. Wystarczy wygooglować czy wybingować – teraz nie jestem pewien, co powinniśmy mówić – Bing Image Generator" i możemy wpisywać i generować sobie różne ciekawe rzeczy: ikony, obrazy. Te efekty są coraz, coraz lepsze.

Poza tym mamy dużo takich wyspecjalizowanych narzędzi do generowania konkretnych rzeczy: grafiki wektorowej, do pracy z kodem, do pracy z testami. Mamy rzeczy, które są trenowane po to, żeby generować specyficzne typy contentu, tekstu.

Ja w naturalny sposób jestem trochę subiektywny, jeśli chodzi o te narzędzia pod kątem kodu i pod kątem tego, co robiłem, więc zachęcam każdego do zastanowienia się, popatrzenia na te narzędzia: „jak mogę je wykorzystać" albo „co może mi dać". Myślę, że jedną z największych zalet dla mnie personalnie w pracy z tymi narzędziami, był nie tyle output z tego AI, tylko samo zastanawianie się nad tym, co mogą mi dać te narzędzia – to sprawiło, że zacząłem myśleć o programowaniu na nowe, kreatywne sposoby. Myślę, że jeśli AI samo w sobie nie pomoże nam w zastosowaniu, to sama praca z nim, brainstormowanie, wymyślanie nowych rzeczy jest na tyle interesujące, żeby spróbować.

No dobrze, to zanim wejdziemy w te tematy jeszcze głębiej, bo i tak już się mocno rozgadaliśmy, to ciekawy jestem kwestii

prywatności. Kto tak naprawdę ma dostęp do tych danych wprowadzonych do tych narzędzi? Czy w ogóle bezpieczniej jest wprowadzać te dane, dane pracodawcy, dane naszego kodu, dane naszej aplikacji, może klientów? Jak to wygląda? Perspektywa prawna, prywatności itd.

Dobre pytanie. Musiałem się z nim ostatnio zmierzyć właśnie w pracy. Zostaliśmy poproszeni przez naszego security officera o podanie narzędzi, z których korzystamy. Razem z nimi pracowałem nad wewnętrzną polityką używania takich narzędzi AI. Przeczytaliśmy regulaminy, przeczytaliśmy polityki prywatności i te rzeczy są czasem napisane w trochę mylący sposób. Jeśli zobaczymy choćby politykę prywatności GitHub Copliota, to spotkamy dużo różnych zapisów, które nie do końca mówią, gdzie te dane idą. Albo spotykamy różne stwierdzenia i te różne stwierdzenia czasem biorą się z tego, że wersje prywatne i wersje biznesowe tych narzędzi mają zupełnie inne zasady przetwarzania danych. Bo o ile biznes nie może sobie pozwolić na zakup narzędzia, które wykorzystuje jego dane, regularni użytkownicy mogą przeoczyć takie zapisy albo mogą się na to zgodzić przez same korzyści, które płyną z tych narzędzi.

Chciałbym więc tylko uczulić wszystkich, że tak jak powiedzieliśmy wcześniej, dużą wartością dla firm są dane z naszego użytkowania tych narzędzi, czyli to, że poprosimy AI o wygenerowanie czegoś, a potem poprosimy o wygenerowanie jeszcze raz, bo nam się nie podobało. To jest bardzo cenny feedback dla każdej firmy, która oferuje takie narzędzie, więc te dane są na wagę złota. Jeśli pracujemy w firmie i chcemy np. wrzucać kod firmowy do ChatGPT, musimy zdać sobie sprawę, że te dane w tym momencie, kiedy robimy „Ctrl+V, wyślij”

opuszczają rewir, nad którym ma kontrolę nasza firma i nasz dział IT. Idealnie by było, gdyby ten dział IT miał politykę, która pozwala na korzystanie z niektórych narzędzi, a przed niektórymi uczula. Tak pewnie nie będzie, więc musimy pamiętać, że my bierzemy odpowiedzialność za te dane i za to, co wychodzi.

Na szczęście przez ostatnie dwa miesiące, po tym, jak Włochy zbanowały ChatGPT na miesiąc, firmy dały trochę opcji customowania tego, co się dzieje z naszymi danymi. Więc np. jeśli korzystamy z ChatGPT, wrzucamy tam jakieś dane i nie chcemy, żeby zostały one wykorzystane do trenowania, jest taka opcja, żeby wyłączyć historię czatów. Jest to mniej wygodne, bo nie mamy zapamiętanego tego outputu, musimy go sobie skopiować, ale wtedy OpenAI obiecuje nam, że nie trzyma tych danych do trenowania. Trzyma je tylko 30 dni dla celów prawnych, jeśli np. zrobilibyśmy coś złego z tymi danymi, jakiś rząd może do nich przyjść i zapytać o to, co robiliśmy. Ale wciąż trzymają to przez 30 dni dla celów, jak to oni nazywają, bezpieczeństwa. Są to rzeczy napisane bardzo małym druczkiem w polityce prywatności.

Ja bym był skłonny zaufać tym ustawieniom danych, ale musimy się zastanowić, czy możemy sobie na to pozwolić. Jeśli rozmawiamy o znajomych, prosimy o porady prywatne albo piszemy rzeczy, co do których nie chcielibyśmy, żeby kiedyś wypłynęły, to zwróćmy na to dużą uwagę. Ja, generując teksty np. o mojej firmie, zmieniałem nazwę firmy tak, żeby nie dało się w danych później wyszukać tej nazwy firmy. I potem z powrotem wracałem do nazwy firmy, używając „znajdź i zastąp” w moim edytorze kodu. Bardzo mały krok, który można wykonać, ale później ktoś szukający danych nie będzie mógł tego zrobić, albo przypadkiem, pytając o firmę, nie będzie mógł trafić na taką informację.

Ogólnie nie chcę straszyć nikogo, bo te narzędzia są super, ale upewnijmy się, że dane, które wrzucamy do tych narzędzi, nie są wrażliwe. A jeśli są, to że dział IT wie, że korzystamy z tego. I lobbujcie, zachęcajcie dział IT do kupowania Wam tych narzędzi, bo licencje biznesowe gwarantują to, że te dane nie zostaną nigdzie wykorzystane. Licencja biznesowa Copilota jest dwa razy droższa dla firm właśnie dlatego. A licencji biznesowej OpenAI i chatu w tej chwili nie ma. Uważajmy więc na to, co robimy, szczególnie z danymi firmowymi, które mogą nam przysporzyć kłopotów.

Ja może bym dodał jeszcze od siebie taką historię, którą chyba nieraz przytaczałem odnośnie do tego, że kiedyś na podstawie zakupów opłacanych kartą kredytową firmy były w stanie wywnioskować na przykład, czy żona już jest w ciąży, gdy mąż jeszcze nie wiedział. Trzeba pamiętać, że zmiana nazwy firmy to jest dobry pierwszy krok, ale jeżeli pytamy się o różne rzeczy ChatGPT, to on na podstawie tych kilku zapytań też może wyciągnąć jakieś wnioski i powiązać fakty. Może to być więc taki temat, na który warto zwrócić uwagę. Co myślisz?

Oczywiście, zmiana nazwy niedużo da, jeśli bylibyśmy profilowani. Nie chcielibyśmy raczej dostać kuponów na pieluszki dla dzieci, tak jak ludzie z tej historii. Bardzo dobrze ją kojarzę. Myślę, że ostrożnie. Zachęcające jest np. przetestować ChatGPT jako terapeutę, ponieważ możemy sobie porozmawiać o naszych problemach, dostać kilka pomysłów i pewnie nic się nie stanie. Prawdopodobieństwo, że cokolwiek z tego wycieknie jest małe.

Ale już przez to, że ChatGPT został wydany bardzo szybko, widzieliśmy bug, który sprawił, że nawet 10% danych, które weszły do ChatGPT, mogło być widoczne dla innych. Był taki tydzień, około miesiąc temu, kiedy wprowadzono bug na wersję produkcyjną ChatGPT. To oczywiście nie jest niczyja wina ani czyjeś złe działania, ale pamiętajmy: ograniczone zaufanie. Jak w każdym narzędziu – nie wklejamy czegoś, co do czego bardzo nie chcielibyśmy albo mielibyśmy problemy gdyby wypłynęło.

Tak. To może teraz przejdźmy do tematu samego oprogramowania. Jakie narzędzia i w jaki sposób pomagają nam tworzyć kod? Trochę o tym wspomniałeś, ale może chciałbyś coś więcej na ten temat powiedzieć. Pamiętajmy też, że programowanie to nie sam kod, ale może być to coś więcej, o czym też wspominałeś, czyli dokumentacja albo nawet sama architektura aplikacji. Jak tutaj możemy się wspomagać?

Tak się składa, że przychodzę do Ciebie do podcastu pełny inspiracji, bo w zeszłym tygodniu uczestniczyłem i mówiłem na konferencji Code Europe, która była w Krakowie i w Warszawie. Był tam taki *keynote speaker* – Andrei Alexandrescu z NVIDIA – i rozmawialiśmy dużo o narzędziach. Jego *keynote* też wspominał o ChatGPT. Rozgraniczył on umiejętności czy aspekty programowania na trzy sfery: umiejętności programistyczne, kreatywność i gust (dobry smak).

ChatGPT i inne narzędzia programistyczne w tej chwili mogą nam pomóc głównie w tej kwestii umiejętności czy znajomości bibliotek oraz generowania kodu, który później będziemy musieli sprawdzić. Tak jak

wspominałeś: inni mówili Ci, że to rozwiązanie, które dostają, nie jest do końca optymalne i Ty jako programista musisz wiedzieć, że ono nie jest do końca optymalne. Więc mała gwiazdka do tego wszystkiego: ChatGPT i GitHub Copilot generują zwykle całkiem działający kod albo są bardzo przydatne, ale to od naszego gustu czy intuicji programistycznej zależy, czy zaakceptujemy to rozwiązanie.

Musimy też wykazać pewną kreatywność, ponieważ ChatGPT jest ograniczony w tym, jakie rozwiązania może wymyślić. Tak jak mówisz: bazą do trenowania tych modeli jest ogół Internetu, jest GitHub, więc ChatGPT będzie nam proponował rzeczy, które już zostały stworzone, albo jakąś średnią wynikową rzeczy, które już zobaczył w Internecie. Jeśli chcemy coś zrobić inaczej, ChatGPT czy GitHub Copilot, czy jego alternatywa od Amazonu CodeWhisperer nam w tym nie pomogą.

Ale mówiąc już o konkretnych narzędziach, taką podstawą, którą możemy sobie zainstalować jest GitHub Copilot. Mała uwaga: żeby korzystać z GitHub Copilota, musimy zapłacić za wersję premium GitHuba. Ale mamy też alternatywę od Amazonu, czyli Amazon CodeWhisperer, która jest za darmo i możemy z niej korzystać.

Znam parę osób, które testowały te narzędzia. Nie mam jakichś naukowych wyników tych badań, ale CodeWhisperer radził sobie delikatnie gorzej. Miał jednak też zalety, które imponują choćby ludziom, którzy muszą pracować w korporacjach i spełniać jakieś guideline'y. Na przykład CodeWhisperer jest nam w stanie wskazać, w których bibliotekach open-source i na jakich licencjach wystąpiły podobne linie kodu – tak żebyśmy przypadkiem nie naruszyli praw autorskich. Wiem, że powtarzam tutaj rzeczy takie co do bezpieczeństwa, polityki

prywatności, ale to będzie ważne, jeśli programujemy nie dla siebie, tylko dla firmy. Tak jak wspomnieliśmy: te narzędzia pomagają, przyspieszają, wciąż wymagają dużo naszego inputu.

No i tutaj wchodzi nasz super generator tekstu, czyli ChatGPT. Możemy go wykorzystać i do napisania kawałków kodu, i do dokumentacji, i do testów, ale też do pracy koncepcyjnej. Tak jak wspominaliśmy, są różne aspekty tworzenia oprogramowania, tak samo też są różne wyniki: mamy kod, mamy dokumentację, mamy testy, mamy też czasem specyfikację (co prawda od niej się powinno zaczynać).

I teraz pytanie: które z nich będziemy pisać? Bo jeśli otworzymy sobie ChatGPT, musimy od czegoś zacząć. Więc mówimy: „hej, chciałbym napisać kawałek kodu, który zrobi to, to i to". Jest to już jakaś specyfikacja i możemy dostać kod, możemy popracować nad tym kodem.

Co jest istotne, bardzo rzadko od razu dostaniemy dobre rozwiązanie, więc z ChatGPT czasem trzeba przejść przez wiele wariantów tego kodu, dawać uwagi i być takim kuratorem. To będzie rozmowa, w której my jesteśmy tak naprawdę oceniającym kod, a rozmawiamy z jakimś programistą. Mówimy więc: „Odpaliłem ten kod nie działa, dostaję taki błąd. A poza tym mam wrażenie, że zapomniałeś o tym i tym". Musimy więc pracować nad tym iteracyjne. A w momencie, gdy stworzymy taki kod albo kiedy mamy już kod, z którym pracujemy, możemy poprosić: „hej, czy mógłbyś mi napisać testy?". I na tyle, na ile kod ma widoczne zastosowania, ChatGPT napisze testy, ale niekoniecznie pokryje nam wszystko, co chcielibyśmy, żeby pokrył.

Ale co do samego generowania kodu powiedziałbym, że i ja, i wiele innych osób skorzystało z ChatGPT z bardzo dobrymi efektami. Tylko pamiętajmy: jeśli opisujemy, co trzeba zrobić, opiszmy to w jak największych szczegółach, ponieważ jeśli zadamy ogólne pytanie, dostaniemy ogólne rozwiązanie, a jeśli doprecyzujemy to, czego potrzebujemy, to dostaniemy coś, co idealnie wpasuje się w nasz przypadek użycia.

OK, to teraz skupmy się na dokumentacji. Czy jesteśmy w stanie wspomóc się sztuczną inteligencją na tym etapie? Wiadomo, że jest to uciążliwe. Programiści raczej tego nie lubią albo nawet tego nie piszą. Potem coś się zmienia, trzeba zaktualizować, ale różnie z tym bywa. Jak widzisz tutaj ten aspekt usprawnienia?

Ja miałem okazję skorzystać z ChatGPT do wygenerowania kawałków dokumentacji w sytuacji, w której potrzebowałem, żeby coś bardzo technicznego zostało przełożone na bardzo prostą terminologię. Mogłem to zrobić, mogłem to podyktować, mogłem to wpisać do dokumentu samemu, ale chciałem zobaczyć jak ChatGPT sobie poradzi. I w zasadzie napisał wszystko to, co chciałem napisać. Mogę więc powiedzieć, że ChatGPT pomoże nam wygenerować dobrą dokumentację.

A jeśli mówimy o aktualizowaniu, to jak najbardziej poradzi sobie z takim zadaniem, że wklejamy stary kawałek dokumentacji, wklejamy nowy kod, albo nawet wiemy, co się zmieniło i ChatGPT będzie w stanie nam zaktualizować dokładnie to, czego potrzebujemy. Jest więc na pewno dobry do takiego zastosowania, z tym że jeśli mówimy o

dokumentowaniu kawałka kodu, jeśli mamy kod z dobrze nazywanymi zmiennymi, jeśli mamy kod, w którym proces działania tego kodu jest jasny, to ta wygenerowana dokumentacja będzie bardzo dobra. Jeśli będziemy mieli zmienne nazwane literkami, to ChatGPT też nie wymyśli, co się dzieje w tym kodzie.

Jeśli więc piszemy samodokumentujący się kod i potrzebujemy opisu tego wszystkiego, wystarczy ChatGPT. Może czasem, jeśli ten schemat kodu przypomina to, co wcześniej było na GitHubie, to, co było w danych treningowych, nawet jeśli zmienne są literkami, to ChatGPT jest w stanie odcyfrować, co się w tym kodzie stanie.

Nie wiem, jak Ty patrzysz na moją wypowiedź, ale ja, jak to wszystko mówiłem, mam takie pytanie z tyłu głowy: czy w takim razie jest sens pisać dokumentację, skoro jedyne, co robię, to biorę kod, wklejam do ChatGPT i mówię: „proszę, opisz mi to do dokumentacji technicznej”? Czy w takim razie w ogóle pisanie takiej dokumentacji przez programistów ma sens? I to jest coś, co wzbudziło moje zainteresowanie, i zacząłem trochę o tym czytać.

Są narzędzia, do których możemy wczytać w zasadzie całą bazę kodu i wygenerować dokumentację dla całego projektu lub po prostu rozmawiać z tym projektem, rozmawiać z kodem. Jeśli mamy konkretne pytanie, nie generujemy tej dokumentacji live, nie generujemy tekstu, tylko prosimy AI, które ma świadomość tego, co jest w całej bazie kodu, o wytłumaczenie nam pewnego konceptu.

I gdybym miał robić jakieś predykcje, to myślę, że skończymy raczej z czymś, co będzie działało w podobny sposób. Nie będziemy proszeni jako programiści o pisanie dokumentacji, która powtarza to, co jest w

kodzie, tylko jeszcze bardziej istotne stanie się pisanie samoopisującego kodu, bo z takiego kodu bardzo łatwo uzyskamy dokumentację. Jeśli zmienne nazywane są dobrze, jeśli tok tego, co się dzieje, jest zrozumiały, jeśli damy małe ilości komentarzy, które wyjaśniają bardziej newralgiczne kawałki kodu, to taka dokumentacja, taki minimalny opis tego, co się dzieje w kodzie, będzie bardzo łatwo przetwarzany przez ChatGPT na coś, co mogą czytać zwykli użytkownicy, co mogą czytać nietechniczni Product Ownerzy czy nasi nowi koledzy z zespołu.

Chciałbym tylko tutaj wspomnieć, że jeśli mamy w tej chwili pracę, w której ktoś nam każe pisać bardzo dużo dokumentacji i niekoniecznie zawsze to ma sens (czasem podążamy za rutyną. Niektórzy Project Managerowie chcą widzieć wszystko bardzo dobrze udokumentowane albo klient tego wymaga). My potencjalnie jako jednostki mamy złoty czas, ponieważ dokumentacja, którą generuje ChatGPT jest zwykle bardzo dobra, szczególnie, jeśli ją sprawdzimy. Myślę, że krótkoterminowo, zanim pojawią się te narzędzia, które generują tę dokumentację same, to jako programista ktoś może się znaleźć w bardzo dobrym momencie, ponieważ dostaje dwa dni na napisanie dokumentacji, a ta dokumentacja jest napisana w parę minut przez ChatGPT.

Chciałbym tylko w tym momencie przypomnieć, że jako profesjonaliści bierzemy odpowiedzialność za to, co wysyłamy, i za to, co commitujemy. Więc jeśli wygenerujemy dokumentację w ten sposób i będą tam błędy, to to nasza odpowiedzialność, nie OpenAI, która dostarcza modele.

Tak jest, tak że ciekawa perspektywa. Też mi się spodobała taka opcja. Ja może bym tylko dodał jeszcze jedną rzecz dla tych, którzy

nas słuchają, że jeżeli teraz, po tym co powiem, ktoś wymyśli takie rozwiązanie, wdroży i zarobi milion, to dla mnie i Karola przynajmniej po 10%.

Oczywiście.

Bo może będzie takie rozwiązanie na zasadzie: jak mamy Gita, to po prostu wrzucamy tam zmiany i widzimy, co się zmieniło. Może będzie dodatek, np. do Gita, który będzie generował całą dokumentację. Czyli na początku mamy dokumentację, a potem każda zmiana to jest aktualizacja tej dokumentacji w automatyczny sposób na podstawie ostatnich zmian, które zostały wprowadzone. To byłoby super rozwiązanie. Może już takie jest, nie wiem, ale to by mi się bardzo spodobało.

Powiem Ci, Mateusz, że nie widziałem jeszcze tego rozwiązania, ale myślałem o tym, co bym chciał powiedzieć w rozmowie z Tobą. Miałem dokładnie tę samą myśl, miałem takie: „Może warto by zrobić ten start-up, żeby przygotować gotowy klocek do Continuous Integration, który każda firma może sobie wrzucić po zbudowaniu kodu, który automatycznie aktualizuje dokumentację na podstawie kodu, ewentualnie dodaje kolejny krok do sprawdzenia i zaakceptowania przez kogoś?”. To by było idealne. Myślę, że narzędzia w tej chwili otwierają bardzo dużo małych zastosowań, więc jeśli ktoś ze słuchających nas lubi tworzyć małe aplikacje, lubi tworzyć różne zastosowania i ma pomysły, to zachęcałbym, żeby publikować jak najszybciej.

Tak jest.

To może teraz przejdźmy do trudnego tematu, może nie z perspektywy AI, ale z perspektywy programisty – tzw. *legacy code*, czyli kod, który my musimy utrzymywać, a niekoniecznie chcemy. I co tutaj zrobić? Czy myślisz, że w tym temacie może nam pomóc sztuczna inteligencja, która podpowie nam, jak to utrzymywać, jak to zrozumieć i jak to dalej rozwijać, żeby się nie okazało, że wszystko się posypie?

Rzecz z *legacy code* jest też taka, że każdy kod w momencie zacommitowania staje się legacy code'em. Bądźmy więc wyrozumiali dla naszych kolegów, którzy kiedyś zrobili kod, który jest teraz naszym problemem. Ale ChatGPT zdecydowanie może nam pomóc wyjaśnić kod, na który patrzymy. Nie pomoże nam jednak w odpaleniu projektu. Nie wiem, czy spotkałeś się z tym, Mateusz, ale jak dostaje się stary projekt albo taki ciężki projekt, gdzie bardzo dużo kodu zostało napisane szybko, czasem problemem jest odpalenie go – szczególnie jeśli ktoś nie przygotował środowiska albo nie przygotował docker file'a, który nam pozwoli bardzo dobrze go odpalić.

Wiesz, ja tak sobie teraz pomyślałem, że warto mieć w zasięgu jakiś taki komputer, który jest z tego roku, co ten kod, i po prostu na nim go uruchomić.

To by było czasem najprostsze wyjście. Dlatego też polecam każdemu, kto musi zrobić coś takiego, zainteresować się Dockerem, ponieważ możemy odpalić system Linux np. sprzed 5-10 lat, zainstalować bardzo stare dependencje i być może wtedy projekt ruszy.

To może tylko dodam, że o Dockerze był już odcinek, więc odsyłam.

Tak, zdecydowanie polecam. To technologia, którą warto znać. Nieważne czy się interesujemy DevOps-owymi rzeczami, czy nie. Na pewno też pomoże w dostaniu wszelkiej pracy – Docker jest pewnym standardem w tej chwili w wielu zastosowaniach.

W każdym razie weźmy pod uwagę taki niedawny *legacy code*, czyli kod, który napisali nasi koledzy jakiś czas temu, i totalnie nie rozumiemy, co się tam dzieje. Mnie zdarzyło się wziąć funkcję, której nie rozumiem, wkleić do ChatGPT i poprosić: „wytłumacz mi, co się tam dzieje". I to, co dostaniemy, to zwykle krok po kroku opis słowny, co dzieje się w takiej funkcji. Czyli wracamy trochę do tego, jak dobry ChatGPT jest w dokumentowaniu kodu. I ja np. nauczyłem się, w jaki sposób działa jeden ze skomplikowanych hooków w Reakcie, prosząc właśnie ChatGPT o wyjaśnienie mi, jak działa ta funkcja, ponieważ nie rozumiałem schematu działania, nie rozumiałem, kiedy dzieją się te asynchroniczne eventy. To zostało mi wytłumaczone.

Jeśli więc widzimy kawałek kodu, którego absolutnie nie rozumiemy, dajmy jakiś kontekst ChatGPT, wklejmy ten kawałek kodu i możemy dostać wyjaśnienie, które powie nam, co się dzieje. Ograniczenie jest tylko wtedy, gdy np. pracujemy z programowaniem systemowym, niskopoziomowym, piszemy jakiś algorytm w C++ – dostaniemy zapewne wyjaśnienie, co robi ten kod, ale będzie o to trudno, jeśli pod spodem dzieją się jakieś operacje na pamięci, które są niezrozumiałe, albo są przesunięcia bitowe, są rzeczy, które dzieją się na etapie optymalizacji procesora.

ChatGPT zwykle nie jest w stanie zrobić tego ćwiczenia, które robi się często na kursach czy na studiach, czyli przejścia na tablicy przez to, co

się dzieje z inputem do funkcji, aż do wyjścia z tej funkcji. Modele językowe w tej chwili (co prawda niektórzy spekulują, że mają jakieś modele świata i sposób rozumienia zaszyty w tych setkach tysiącach neuronów) nie są w stanie zwykle przejść przez kompletny proces logiczny tak skomplikowany, jak choćby długa funkcja. Więc jeśli poprosimy ChatGPT o symulowanie działania funkcji dla danego inputa, to nie zawsze dostaniemy dobre wyniki. Mamy więc tutaj takie małe ograniczenie.

Tak jak wspominałem, świetnie się to sprawdza do wytłumaczenia czy zrozumienia konkretnych funkcji czy funkcjonalności. Obecnie jednak ciężko jest znaleźć narzędzie, które pozwoli nam popatrzeć na cały kod, czyli powiedzmy otwieramy projekt, w README mamy tylko nazwę projektu, ewentualnie mamy trochę lepszą dokumentację, ale nie wiemy od czego zacząć i chcielibyśmy skorzystać z narzędzia AI, żeby wytłumaczyła nam, o co chodzi w tym projekcie. Niestety obecnie ChatGPT nam w tym nie pomoże. Nowa wersja GitHub Copilota czy Copilot X, czy Copilot Chat nie pozwalają patrzeć na całość kodu. Copilot Chat różni się od zwykłego Copilota tym, że daje nam okienko chatu w naszym edytorze kodu i pozwala rozmawiać z kawałkami kodu, prosić o akcje na kawałkach kodu bez kopiowania ich do przeglądarki.

To już jest dostępne?

Copilot X w tej chwili jest dostępny w becie. Można zapisać się na waiting listę. Ja dostałem wstęp na taką waiting listę i testuję, ale nie jestem zachwycony – właśnie głównie przez to ograniczenie. Czyli mam okienko chatu po lewej od mojego edytora, mogę zaznaczyć kawałek kodu i poprosić np. żeby chat dodał mi jakąś funkcjonalność do tego

kawałka kodu, ale nie mogę pracować z kontekstem całej aplikacji w taki sposób, w jaki bym chciał.

Wczytywałem się, jak działa Copilot, żeby zrozumieć, o co mogę go prosić, i pod spodem, jeśli skorzystamy z Visual Studio Code, Copilot bierze plik, który przypomina kod, który piszemy w naszym projekcie, wyciąga z niego najpotrzebniejsze rzeczy za pomocą jakiejś heurystyki i wysyła razem z linijką kodu, którą ma uzupełnić, albo z plikiem, który ma uzupełnić. I tym sposobem on się czasem zachowuje tak, jakby widział inne pliki w naszym projekcie. To nie jest prawda, bo widzi zwykle jeden, który jest w stanie wczytać.

Tak samo w ChatGPT – nie skopiujemy tam całego wielkiego projektu, bo mamy ograniczenie długości kontekstu. W tej chwili ChatGPT jest w stanie widzieć i generować ciągi tekstu o długości 4096 tokenów, gdzie token to jest zlepek 3-4 liter. Nie wkleimy więc bardzo długiego kodu. Nie wkleimy też całego projektu.

Próbowałem znaleźć narzędzie, które by na to pozwoliło. W tej chwili jest dostępnych kilka. Takie jedno, które mi wyskoczyło, to było Adrenaline. Jeśli wczytamy tam projekt, indeksuje ono cały projekt. Robi coś, co w AI nazywamy „embeddingi”, czyli umiejscowienie tekstu w przestrzeni liczb, która reprezentuje tekst i w trakcie pytania pozwala takiemu modelowi wyszukać w tej bazie zaindeksowanego kodu konkretne fragmenty, które może użyć do udzielenia odpowiedzi. Jest to jednak dość specyficzne rozwiązanie, pewnie nie wszyscy po nie sięgną, więc radziłbym obserwować, co się będzie działo, lub ewentualnie skorzystać z takiego narzędzia, jeśli bardzo by nam pomogło.

Czy korzystałeś więcej z tego narzędzia? Ja dzisiaj odpaliłem i wrzuciłem tam projekt z GitHuba, i wydaje mi się, że do zrozumienia kodu jest to całkiem fajna opcja. Jeżeli np. mamy jakieś zadanie, mamy nowy projekt albo czyjś projekt i chcemy zobaczyć, co tam się dzieje, to w zasadzie po wrzuceniu takiego projektu zostało mi opisane na górze każdego pliku, co tam się dzieje. Dużo więc łatwiej da się zrozumieć, gdzie mogę wprowadzić ewentualne zmiany.

Starałem się też podpytać, gdzie co mogę zmienić, i tak powiedziałbym… no prawie mu się udało. Nie było to skomplikowane.

Ciekawy projekt, ale w ogóle zero informacji. W zasadzie tam jest chyba Discord, tylko nie ma napisane, ile co kosztuje, i na co można się decydować, czy np. jest jakieś API. Dość mało informacji. Chyba dopiero jest to rozwijane.

Super w takim razie, że to działa, bo tak jak mówię, ja Copilota zapytałem w tym okienku chatu, jak wygląda struktura projektu, na co powinienem patrzeć. Dostałem odpowiedź: „nie wiem, jak wygląda struktura tego projektu, ale patrząc na plik, który właśnie masz otwarty, jest to projekt w Reakcie i powinien mieć taką strukturę plików”. Czyli totalnie nieprzydatna rzecz dla *legacy code*. Cieszę się więc, że są takie narzędzia, które już działają.

Ale tak jak mówisz: te narzędzia są bardzo świeże. Bardzo często nie mamy dokumentacji, mamy Discorda, gdzie musimy przeskanować czasem tysiące wiadomości, żeby coś znaleźć. To jest wszystko świeże i myślę, że o ile są takie narzędzia, które jeszcze chwilę z nami zostaną

(jak ChatGPT), to bardzo dużo narzędzi będzie się pojawiało i znikało, eksperymentując z tym, czy można to w ogóle monetyzować i czy zastosowania są możliwe. Ale skoro Adrenaline działa, to pewnie pojawią się jacyś gracze, którzy to wdrożą, żeby dla nas działało. Dobrze więc, patrzymy na to z nadzieją.

No dobrze, to przejdźmy teraz do tematu testów. Wiemy, że testy to świetna sprawa, bardzo dobra praktyka. Fajnie byłoby pokryć nasz projekt testami, żeby wiedzieć, czy coś się nie popsuło, gdy coś dodamy. Ale na pewno zajmuje to czas. Czy jesteśmy w stanie jakoś usprawnić swoje działanie, wykorzystując sztuczną inteligencję?

Na pewno tak, bo tak jak wspomniałem, ChatGPT jest dobry w tłumaczeniu dokumentacji na kod i kodu na dokumentację, nie ma więc żadnego problemu, żeby był w stanie napisać do naszego kodu testy jednostkowe lub nawet bardziej skomplikowane.

Kwestia jest tylko taka, czy te testy też będą kiedyś potrzebne, ponieważ może powstać kolejne narzędzie (też sugerujemy komuś, żeby je zrobił i żeby pamiętał o tym, że pomysł był nasz), czyli coś, co wrzucamy w pipeline, i prosimy AI o wygenerowanie testów, puszczenie tych testów i zgłoszenie ewentualnych błędów.

Pamiętam moje pierwsze przygody z testami – absolutnie nie miałem pojęcia, co testować. Nie miałem pomysłu, do czego napisać test, co warto sprawdzać. Będąc na początku albo już nawet bardziej zaawansowanym w programowaniu, zapytajmy ChatGPT, jakie

napisałby testy do tego kawałka kodu. Zapewniam Was, że dostaniecie przynajmniej pomysły testów, które warto napisać, ale też działające testy do Waszego kodu. Taka uwaga: testy front endu są bardzo skomplikowane, ponieważ dużo zmienia się na stronie, zmienia się stan, czasem musimy zasymulować klikanie przycisków, żeby wykonały się jakieś funkcje. I w takich przypadkach, kiedy mamy wielokomponentowe strony czy komponenty, które mają zagnieżdżone inne, mniejsze komponenty, ChatGPT robi dużo błędów. Jesteśmy jednak w stanie dojść do satysfakcjonującego rozwiązania, do testów, które pokryją sporo funkcjonalności.

W przypadku back endu jest trochę prościej, bo zwykle staramy się przynajmniej pisać funkcje w taki sposób, żeby nie modyfikować niczego pod spodem, żeby nie miały side effectów. Jak wiadomo, nie każdy jest dobry w programowaniu funkcyjnym albo czasem mamy wielkie klasy, które modyfikują swój stan wewnątrz bardzo często. Takie rzeczy są ciężkie do testowania nie tylko dla AI, ale też dla ludzi.

Myślę, że jeśli macie zadanie pisania testów albo chcecie się nauczyć pisania testów, to ChatGPT Wam bardzo pomoże. Kwestia jest tylko taka, czy takie testy będą satysfakcjonujące, czy pomogą w pisaniu tego kodu. Myślę, że tak, bo jeśli mamy kawałek kodu i napiszemy regression tests – testy, które patrzą, czy ta oryginalna funkcjonalność działa dobrze – i później będziemy wprowadzać zmiany w innych kawałkach kodu, to te testy się przydadzą. Upewnią nas, że ta funkcjonalność, która była, pozostaje.

Jednakże ChatGPT nie będzie znał kontekstu biznesowego, tego, po co jest nasz kod, więc o ile mu go nie damy, nie dostaniemy testów, które

pokrywają właśnie takie zastosowania. Uczulałbym więc na to, że jednak wciąż mamy dużo wkładu w ten kod, bo to, czego nie pokażemy, AI nie będzie widziało.

Moje doświadczenie z tymi testami jest też takie, że testy, które dostawałem od ChatGPT, zwykle zawierały błędy, były niepełne i szczególnie właśnie dla front endu musiałem bardzo dużo ich poprawiać. Myślę, że w tej chwili ChatGPT może zmniejszyć nakład pracy ludzi, którzy są proszeni o pisanie projektów z bardzo dużym coverage'em testowym, bo te takie oczywiste testy ChatGPT można napisać za nas. My możemy skupić się na testach, które są naprawdę skomplikowane, które testują funkcjonalności biznesowe i które potencjalnie mogą wykryć najbardziej krytyczne błędy. Polecałbym więc z tego skorzystać.

Jeśli piszemy np. w metodologii TDD (Test Driven Development), możemy najpierw napisać testy i poprosić ChatGPT o zrobienie prototypu kodu na podstawie testów – to też jest możliwe. Testowałem to. Jeśli mamy dobre testy opisujące funkcjonalność, to tak naprawdę mamy bardzo dobrą specyfikację tego, co powinien robić kod. Myślę więc, że świetnym pomysłem jest też wzięcie ChatGPT i użycie go jako tego kolejnego kroku w Test Driven Development. Jeśli napiszemy dobre testy, które testują kod, ChatGPT może nam pomóc uzupełnić to, jak działa ten kod, i nasze pisanie testów będzie instrukcją dla sztucznej inteligencji i jednocześnie weryfikacją tego outputu. Sądzę, że w tym przypadku AI jest nawet bardziej przydatne niż w przypadku generowanie testów do już istniejącego kodu.

Myślę, że ta część może się przydać naprawdę dużej liczbie osób, bo raczej ludzie nie lubią pisać testów, Jak usłyszą, że mogą zostać

one napisane przez sztuczną inteligencję, a potem jeszcze na ich postawie ewentualnie wygenerowany kod, to już w ogóle super ekstra. Wiadomo, że nie jest to idealne, no ale wtedy i tak mamy mniejszą ilość pracy niż jak byśy mieli wszystko robić od początku, prawda?

Zdecydowanie tak. I jak patrzymy na to, dlaczego ChatGPT może być tak dobry w generowaniu testów, to dlatego, że w repozytoriach na GitHubie, które zostały wykorzystane do uczenia tych systemów, zwykle mieliśmy pliki testów obok plików kodu. Ten input więc powstał. Ale tak jak mówię: ten kontekst biznesowy wciąż jest ważny, więc pamiętajmy, żeby przetestować to, co jest ważne, a nie skupiać się tylko na coverage'u, na pokryciu kodu o stopniu 80%, 90%, 95%.

Wspominałeś o narzędziach, które generują grafiki. Myślisz, że są one w stanie pomóc w pracy programisty?

Zdecydowanie tak, ale jeśli chodzi choćby o web development, to nie mamy do czynienia z taką rewolucją jak w przypadku kodu. Jeśli poprosimy AI o wygenerowanie choćby projektu strony internetowej, te wyniki nie są tak super, ponieważ projektowanie stron internetowych jest w jakiś sposób schematyczne. Niektóre rzeczy trzeba zrobić w dany sposób. Wiemy, że przyciski, które mają być główne, mają mieć pewien kolor i w jakiś sposób odstawać. Wiemy, że przyciski, które mają drugorzędną funkcjonalność, nie powinny mieć tego koloru, tylko np. być obrysem albo być lekko wyciszone, tak żeby użytkownik nie kliknął ich mimochodem. To tak jak chcemy przejść dalej w ścieżce użytkownika i przycisk „dalej" będzie po lewej stronie i jeszcze będzie tylko

obramowany, to będzie dla nas mylące. Więc są zasady, których AI niekoniecznie miało szansę się nauczyć.

Jeśli poprosimy o wygenerowanie layoutu strony internetowej, możemy zobaczyć różne artefakty, krzywe linie, rzeczy, które nie nadają się do późniejszej pracy. W tym przypadku powiedziałbym więc, że jeszcze tam nie jesteśmy. Chociaż jest coraz więcej narzędzi, które umożliwiają generowanie grafiki wektorowej lub pracowanie Figmą, ale powiedziałbym, że jeszcze jesteśmy trochę od tego. Mimo że widziałem całkiem imponujące demo różnych takich funkcjonalności, to sam jeszcze nie miałem okazji pobawić się takimi rzeczami.

Możemy jednak generować grafiki i mamy tutaj dwa aspekty. W pracy programisty mamy grafiki takie jak choćby diagramy. Powiedziałbym, że warto próbować z technologią, która nazywa się Mermaid. Mermaid to są diagramy, które można załączać w Markdownie, czyli choćby pisząc readme na GitHubie, możemy użyć tagu *mermaid* i taki graf zostanie wygenerowany. Możemy zrobić flowcharty czy jakieś wykresy hierarchii. Dzięki temu, że taki wykres definiujemy tekstem, bardzo prostym językiem generowania zależności, powiązań pomiędzy blokami, możemy wykorzystać ChatGPT do wygenerowania takich grafik.

Jeśli chcemy wygenerować jakieś proste grafiki w SVG, proste grafiki wektorowe, to ich definicją też jest tekst, więc ChatGPT może nam w tym pomóc.

Są to oczywiście takie zastosowania, gdzie generujemy tekst. A co z tymi wszystkimi silnikami do tworzenia grafik, tych imponujących zdjęć itd.? Ja mam personalnie problem w znalezieniu takich mocnych zastosowań praktycznych w programowaniu w tej chwili. Jednym, które

znalazłem i z którego sam korzystałem, jest generowanie logo czy ikon aplikacji, bo te narzędzia sobie coraz lepiej z nimi radzą. Jeśli odpalimy takie Midjourney i poprosimy o wygenerowanie ikony na iPhone'a, dostaniemy bardzo dobre efekty. Z tym że wciąż te narzędzia mają problem choćby z prostymi liniami. Poprosimy o kwadrat, a dostajemy lekko prostokąt. Poprosimy o kółko, to kółko jest lekko wydłużone. Więc ta kreatywność, czyli niedoskonałość, która tworzy dobrą sztukę, w momencie, kiedy potrzebujemy precyzji, nie do końca się sprawdza.

Na pewno takim zastosowaniem, które już można wykorzystać w praktyce, jest tworzenie stock zdjęć, czyli zdjęć do promocji, do tła naszych stron internetowych, takich kreatywnych grafik, ponieważ one wyglądają świetnie. Jeśli potrzebujemy sprototypować jakiś concept art do robienia gier, to też będzie wyglądało świetnie. Ale to takie wykorzystanie graficzne wciąż się rozwija i wciąż jeszcze nie jesteśmy w takim miejscu, w którym powiedziałbym: „idźcie, używajcie, pomoże wam to bardzo”, więc graficy i UX designerzy mogą jeszcze chwilę spać spokojnie.

Była to pierwsza z dwóch części rozmowy o wykorzystaniu narzędzi AI w pracy programisty. W następnym odcinku będziemy rozmawiać m.in. o wyzwaniach, jakie czekają programistów w związku z rozwojem sztucznej inteligencji, wykorzystaniu narzędzi w konkretnych przypadkach oraz możliwych zmianach na rynku pracy.