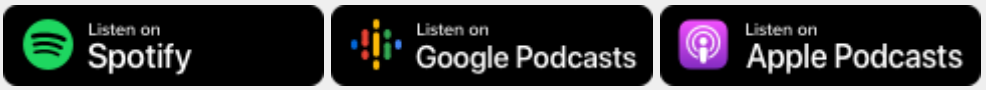
Pierwsze kroki w IT

PODCAST



Jak zostać programistą aplikacji mobilnych

Gość: Krzysztof Baranowski

Wszystkie polecane materiały i linki znajdziesz na stronie odcinka:

https://devmentor.pl/b/jak-zostac-programista-aplikacji-mobilnych

Dziś moim gościem jest Krzysztof Baranowski. Krzysztof opowie nam o tym, jak zostać programistą aplikacji mobilnych. Krzysztofie, dziękuję, że przyjęłeś moje zaproszenie na rozmowę.

Cześć Mateusz, bardzo mi miło.

To może zacznijmy od twojej osoby. Powiedz nam, co łączy Cię z branżą IT i jakie do tej pory miałeś z nią doświadczenie.

Jak miałbym tak w sumie popatrzeć, to chyba całkiem sporo, bo generalnie dzisiaj będziemy rozmawiać o aplikacjach mobilnych i będziemy się poruszać wokół tego tematu. Mam wrażenie, że urodziłem się z komputerem pod pachą, tak że odkąd pamiętam komputer i ja byliśmy jednym. Ale miałem też znajomych – to nie było tak, że nikt nie chciał się ze mną widzieć.

W branży tak oficjalnie jestem od 2016 roku, a zajawka do programowania, pierwsza książka to 2010-2011 rok, coś około tego. Od 2016 to są w zasadzie już tylko aplikacje mobilne i wokół tego typu projektów się poruszam. Pojawiło się tam kilka technologii – był np. Flutter. Swoją przygodę z IT rozpocząłem od Xamarina – jest to podejście cross-platform, o którym na pewno potem porozmawiamy i powiemy, czym to się różni od innych podejść.

No i potem już w przeciągu kolejnych lat mieszałem Xamarina z Flutterem i podejściem natywnym. Przy czym jeżeli chodzi o podejście natywne, to był tylko Android – na pewno jeszcze do tego przejdziemy i o tym opowiem. Obecnie od dwóch lat pracuję w krakowskim oddziale StoneX jako Technical Lead, również w zespole mobilnym przy aplikacji pisanej w Xamarinie.

Po godzinach piszę swojego bloga, może nie tyle „piszę”, co jakoś tam prowadzę i buduję wokół niego społeczność. To, co jeszcze jest dla mnie ważne, to to, że uwielbiam rozmawiać z ludźmi – nuby dość proste, ale dla mnie bardzo ważne. Tak że wydaje mi się, że tyle o mnie.

No dobrze, to w takim razie zaczniemy od pierwszego pytania z zakresu aplikacji mobilnych. Chyba takie najistotniejsze na początek: jak rozpoznać, że programowanie aplikacji mobilnych to nasza pasja, gdy tak naprawdę dopiero zaczynamy przygodę z programowaniem i chcemy się jakoś ukierunkować, na przykład wybrać odpowiedni język?

Ja jestem zdania, że ciężko to stwierdzić, odpowiem: to zależy. To zależy od tego, co nam się podoba. Ja tak naprawdę też dość długo szukałem swojej ścieżki – próbowałem aplikacji webowych… Jak patrzyłem na swoją historię zakupową, to tam jest dosłownie wszystko, więc trochę mi zajęło, by zdecydować, że chcę robić aplikacje mobilne. I to nawet nie „zdecydować", tylko po prostu tak się okazało – zatrudniłem się jako Junior C# Developer i na miejscu w pierwszy dzień pracy okazało się, że będę rozwijał aplikacje mobilne.

U mnie więc było to kompletnie przypadkowe, ale wydaje mi się, że trzeba po prostu spróbować. Na miejscu takiej osoby – która w tym momencie szuka swojej ścieżki i nie ma pojęcia, czy to będą mobilki, czy to będzie web, może trochę back endu, może to będą gry – po prostu bym siadł i poszukał. Mamy internet, dostęp do wszystkich informacji, więc ja zwyczajnie bym spróbował. Jest to może taka dość ogólna odpowiedź, ale ciężko jest mi powiedzieć, jakie wymagania taka osoba musiałaby spełniać i czym się interesować, aby móc powiedzieć, że aplikacje mobilne to jej pasja.

Wydaje mi się, że jeżeli chcielibyśmy sprawdzić, czy aplikacje mobilne są dla nas, to dobrym pomysłem jest wejście do internetu, sprawdzenie, nie wiem, na przykład designów aplikacji i zobaczenie, czym one się

różnią od takich webowych. Czy chciałbym pisać coś takiego, czy może jednak aplikacje webowe, które składają się z większej ilości elementów, które widzimy w danym momencie na ekranie? Ja bym podszedł do tego w ten sposób. Wydaje mi się, że tę kwestię rozwiniemy jeszcze później. Powiedziałbym więc, że wystarczy spróbować, poszukać wiedzy, a o konkretach opowiem później.

To może jeszcze takie dodatkowe pytanie: czy uważasz, że od razu na początku trzeba wybrać tę ścieżkę, czy można przejść z jednej dziedziny do drugiej? I czy jest taki kierunek, w którym to dość naturalnie wychodzi?

Można. Generalnie w czym byś nie programował – czy to będą aplikacje mobilne, web – to paradygmaty są w zasadzie te same. W programowaniu chodzi o coś bardzo podobnego: możemy używać różnych narzędzi, możemy sobie coś trochę inaczej zaprogramować, dany problem możemy rozwiązać na tysiąc sposobów, ale generalnie w każdym chodzi mniej więcej o to samo.

Na przykład do mojego zespołu, w zasadzie w tym tygodniu, dołączyła nowa osoba, która wcześniej była wyłącznie programistą .NET i nigdy nic nie słyszała o aplikacjach mobilnych. Ale przez to, że pracujemy z Xamarinem i technologia jest ta sama (używamy C#), to uznała, że teraz będzie robić aplikacje mobilne, mimo że kompletnie nie wie, jak działają pod spodem. Może jest to taki przykład nie do końca w kontekście twojego pytania, ponieważ ta osoba pracowała wcześniej w C#, teraz też będzie pracować w C#, ale docelowo będzie robiła coś innego, nie będzie robiła back endu, tylko będzie robiła coś, co można zobaczyć (wizualnie), prawda?

Wydaje mi się więc, że można sobie to zmienić. Znowu wrócę do tego, że ja na samym początku nie wiedziałem, że chcę robić aplikacje mobilne, i wiele miesięcy spędziłem z JavaScriptem, z ASP.NET, próbowałem po godzinach wielu różnych tematów… „Po godzinach" – ja wtedy nie pracowałem, więc miałem godziny wolne cały czas. Więc można.

Jakie doświadczenia pomogą nam wystartować w aplikacjach mobilnych? Może powinniśmy mieć zajawkę na przykład na punkcie smartfonów? Czy to nam pomoże?

Ja bym powiedział, że musisz sobie zadać jedno megaważne pytanie: co chcesz zrobić? A potem zacząć to robić. To jest bardzo powiązane z jednym z poprzednich pytań – po prostu wziąłbym jakąkolwiek aplikację na telefonie, zaczął się nią bawić i stwierdził, czy chciałbym pisać coś podobnego. Ktoś może zadać sobie pytanie: czy często korzysta z aplikacji mobilnych, a może raczej częściej korzysta z weba i woli odpalać aplikacje webowe na telefonie (bo aplikacje webowe to dzisiaj już też są aplikacje), czyli po prostu używając przeglądarki, a może woli natywne – te, które instaluje z Google Play albo z App Store? Może to jest jakiś punkt, o który warto się zahaczyć: czy ktoś chciałby robić coś takiego, a może coś takiego?

Można byłoby jeszcze zadać sobie takie pytanie: czy myśl o robieniu aplikacji takich skrojonych idealnie pod konkretne urządzenie mnie w jakiś sposób jara? Czy chciałbym robić coś takiego, co będzie idealnie skrojone pod ekosystem Apple czy pod ekosystem Androida? Wydaje mi się, że można do tego podejść w ten sposób.

Czy zajawka na punkcie smartfonów? Też może jest to jakiś punkt zaczepienia. Trudno mi powiedzieć, bo ja zwyczajnie wróciłbym do tego, co powiedziałem wcześniej, czyli: spróbujmy i zobaczmy, czy nam się to w ogóle podoba. Zawsze po miesiącu czy dwóch możemy wrócić i spróbować kompletnie innej technologii, tym bardziej, że sam – jestem tego zdania i żyję w ten sposób – próbuję wielu rzeczy: trochę poczytam o marketingu, o aplikacjach mobilnych, o architekturze. Wydaje mi się, że każdy z takich klocków w pewnym momencie zacznie nam się łączyć w większą całość. Nie jest więc też tak, że te dwa czy trzy miesiące spędzone na przykład z Flutterem, z Xamarinem czy React Native'em, będą czasem straconym. Na pewno coś z tego nam się przyda, nawet jeśli uznamy, że jednak to nie jest ta nasza bajka i chcemy spróbować innej technologii.

No dobrze, to uznajmy, że nasz słuchacz troszkę się pobawił – czy to w programowaniu strony internetowej, czy programów desktopowych – i teraz myśli o zabraniu się za aplikacje mobilne. Na co się ma nastawić? Czym różnią się początki programowania aplikacji mobilnych od innych form programowania?

Każda z platform, na które tworzymy, rządzi się swoimi prawami – i tu mam na myśli Android, iOS, ale również aplikacje webowe lub nawet desktopowe, czyli takie, które są pisane na komputery osobiste na systemy macOS czy Windows. Każda z nich ma jakieś swoje założenia, prawda? Pewne paradygmaty są dokładnie te same, ale czym się różni to tworzenie?

Na samym początku warto powiedzieć o tym, że każda z platform docelowych rządzi się swoimi prawami. Tworząc na desktop, zrobimy to trochę inaczej niż w przypadku Androida, na iOS-a jeszcze inaczej. Aplikacja webowa również rządzi się swoimi prawami. Jeżeli więc chodzi o aplikacje mobilne, to czym różnią się te początki od innych? Nie wiem, bo nigdy nie tworzyłem tak wprost produkcyjnie aplikacji webowych czy desktopowych, zawsze to było coś tam z boku. Natomiast w przypadku aplikacji mobilnych trzeba się zastanowić, czy chcemy robić tylko Androida, czy tylko iOS-a, a może użyjemy technologii, która pozwoli nam za pomocą jednego kodu stworzyć aplikację na obie z tych platform w jednym czasie. To jest ten punkt startowy: musimy się zastanowić, co będzie naszą platformą docelową.

W przypadku weba raczej nie musimy się zastanawiać, czy użytkownik to uruchamia na MacBooku, Windowsie czy na urządzeniu mobilnym. Pomijam kwestię interfejsu użytkownika, który musi być responsywny. Natomiast narzędzie będzie dokładnie to samo.

Na początku więc musimy się zastanowić nad platformą. Po wyborze platformy musimy użyć konkretnych narzędzi. W przypadku Androida będziemy używać Android Studio, który jest rekomendowanym środowiskiem pracy. W przypadku iOS-a będzie to Xcode. Swego czasu JetBrains – firma, która też dostarcza środowiska – rozwijała aplikację, która (jak dobrze pamiętam) nazywała się AppCode. Wydaje mi się, że ona nie już jest rozwijana, więc w przypadku iOS-a będzie to Xcode. Po wybraniu platformy musimy więc użyć innych narzędzi.

Czym jeszcze różnią się początki tworzenia aplikacji mobilnych? Musimy zdawać sobie sprawę, że na urządzeniu docelowym – mam na myśli

smartfon – mamy ograniczoną ilość zasobów. To jest też tożsame z aplikacją webową, natomiast jak by nie patrzeć, komputery są znacznie mocniejsze od smartfonów, chociaż mamy już teraz takie monstra, które czasami działają nawet szybciej niż komputery.

Generalnie natomiast zasoby są bardziej ograniczone niż w przypadku weba, więc musimy mieć to na względzie, tworząc aplikacje mobilne. Należy szczególną uwagę zwrócić na wydajność: żeby aplikacja była płynna, żeby przyjemnie i responsywnie się z niej korzystało. Użytkownicy mają tendencję do tego, że jeżeli zainstalują jakąś aplikację, a ona się zacina, jest nieresponsywna, długo odpowiada czy się crashuje, to bardzo szybko ją instalują i skorzystają z rozwiązania, które działa lepiej.

Kolejną rzeczą jest to, że istnieje ogromna liczba różnych rozdzielczości (rozmiarów ekranów) urządzeń. W przypadku iOS-a jest to, powiedzmy, trochę łatwiejsze do ogarnięcia, bo tych urządzeń jest ograniczona ilość. W przypadku Androida w zasadzie marek, rozdzielczości i rozmiarów telefonów jest mnóstwo. Budując interfejs użytkownika, musimy być tego świadomi. To jest kolejna rzecz.

Interakcja z użytkownikiem w przypadku aplikacji mobilnej też różni się od tej w aplikacji internetowej. W aplikacji mobilnej mamy na przykład obsługę gestów. Użytkownicy są przyzwyczajeni do tego, że mogą coś przesunąć, dwukrotnie kliknąć. Mogą przesunąć jeden element z miejsca w drugie miejsce. Są to więc rzeczy, o których musimy pamiętać. Na webie nie mamy raczej czegoś takiego jak podwójne tapnięcie (plus minus – to nie jest takie domyślne zachowanie

użytkownika). W przypadku użytkownika aplikacji mobilnej są po prostu inne zachowania. Musimy o tym wiedzieć, tworząc taką aplikację.

Teraz kwestia techniczna dla iOS-a i Androida: ciągle powstają nowe wersje oprogramowania na obie platformy i pojawiają się nowe funkcjonalności, zmiany w bibliotekach obu systemów – mam na myśli biblioteki programistyczne. Musimy być świadomi tego, że kod napisany dzisiaj jutro może się zmienić, bo okaże się, że iOS od dzisiaj zachowuje się trochę inaczej na przykład w przypadku plików albo Android od dzisiaj inaczej współpracuje z kamerą i potrzebuje dodatkowych uprawnień.

Musimy być też świadomi tego, że w tych wersjach pojawiają się nowe rzeczy, inne zostają usunięte… Może nie tyle „usunięte”, co zmieniają swoje zachowanie i pojawiają się ulepszone wersje jakichś tam konkretnych rozwiązań. Trzeba więc wiedzieć, że aplikacje, które tworzymy, musimy zwyczajnie na bieżąco aktualizować i być świadomi tych zmian.

Wersje oprogramowania są bardzo ważne i co ciekawe, wydaje mi się, że (co nie jest tak wprost widoczne) w przypadku iOS-a chyba większość użytkowników przechodzi na nowe wersje oprogramowania bardzo szybko – Apple bardzo mocno tego pilnuje i wymusza tę aktualizację. W przypadku Androida zdarza się, że użytkownicy mają urządzenie z 2016-2018 roku i to nie jest problem, ale wersja oprogramowania jest bardzo stara, sprzed 4-5 lat – od tego czasu dużo się zmieniło. Musimy więc być świadomi tego, że część naszych użytkowników może nie mieć chęci lub nawet możliwości zaktualizowania swojego Androida do nowszej wersji. Nie każdy

producent to umożliwia i nie każdy producent tak długo wspiera swojego urządzenia.

No i właśnie, czym od innych różni się aplikacja mobilna w momencie, w którym chcemy ją wydać do sklepu? No właśnie: do sklepu. W przypadku weba sami decydujemy o tym, kiedy chcemy to zrobić. Chcemy dzisiaj wydać nową wersję oprogramowania lub dodać nowe funkcjonalności dla użytkownika? Możemy to zrobić tu i teraz. Nie musimy pytać o zgodę właściciela internetu, tylko to robimy i nowa wersja jest dostępna dla użytkowników.

W przypadku Androida i iOS-a, jeżeli nasze aplikacje są dostępne w sklepie, musimy przejść proces weryfikacji i cały proces publikacji takiej aplikacji. Jest on dość długi. Wydaje mi się, że możemy ten temat omówić później i opisać cały proces. Natomiast jest to też główna różnica: w webie wydajemy aplikację tu i teraz, a w przypadku Androida i iOS-a musimy przejść dodatkowy proces, który trwa, co oznacza, że nie możemy od razu udostępnić aplikacji użytkownikom.

To może od razu dopytam: czy jesteśmy w stanie jakoś to przyspieszyć? Może jakaś dodatkowa opłata albo znajomość odpowiedniej osoby to umożliwia?

Chciałbym, bo niejednokrotnie mieliśmy w swoim zespole do wydania hotfixy, czyli coś, co już teraz musi być na produkcji, bo na przykład jakaś funkcjonalność kompletnie nie działa. Bardzo często nie możemy tego zrobić tu i teraz. Chciałbym mieć takie wtyki, że dzwonię do Apple (tak, szczególnie do Apple, bo to tam trwa dłużej) i mówię: „dobra, to może byście przepchnęli nam to teraz, bo mamy ważną rzecz do wydania”. No niestety nie, trwa to czasami jeden dzień i to jest super –

czasami jesteśmy w stanie się zamknąć w ciągu jednego dnia z wydaniem aplikacji do sklepu: od momentu wyrzucenia jej do panelu do upublicznienia jej dla klientów. Czasami jednak są to średnio dwa dni.

A najdłużej ile to trwało?

Wiesz co, wydaje mi się, że było to chyba w zeszłym miesiącu (ja mam dość krótką pamięć, więc powiem, co było w zeszłym miesiącu). Nasza aplikacja ma dwa brandy i jeden z nich był zawsze akceptowany. Nie było nigdy żadnych problemów technicznych, nigdy nie było nam nic zwracane. I w tym momencie okazało się, że pomimo tego, że nie wprowadziliśmy żadnej znaczącej zmiany i nic nie zmieniło się w regulaminie sklepu, nasza aplikacja została odrzucona. Wydaje mi się, że komunikacja z nimi – żeby nowa wersja aplikacji została wydana na tym jednym z brandów – trwała chyba dwa tygodnie. Tam weszły już bardziej prawne rzeczy. Tak że dwa tygodnie też może to trwać, przy czym druga aplikacja, która ma dokładnie tę samą funkcjonalność, tylko inaczej jest pokolorowana, poszła bez tego procesu. Nie musiało to tak długo trwać, więc to zależy. Im większa aplikacja, tym pewnie dłużej.

Czyli nie zależy to od rodzaju zmian i ich ilości, tylko często… nie chcę mówić, że od osoby sprawdzającej, ale tak trochę wynika.

Ale taka jest prawda. Też zaczęliśmy analizować i okazuje się, że w zależności od tego, który brand, to ten ma u nas więcej problemów. Mamy poczucie, że chyba trafia to do jakichś innych regionów. Wydaje mi się, że Apple ma po prostu support na całym świecie i zależnie od tego, gdzie to trafi, kto na to popatrzy, to inaczej zinterpretuje niektóre tematy.

OK, to czy coś jeszcze do tego tematu chciałbyś dodać?

Wiesz co, wydaje mi się, że jest jeszcze jedna rzecz, taka dość śmieszna. Żeby być programistą aplikacji mobilnych, musisz posiadać chyba ze 100 różnych kabli do każdego rodzaju urządzenia, to znaczy do każdego wejścia. W przypadku Apple'a powiedzmy, że jest to w miarę oczywiste, ale w przypadku Androida to są micro USB, USB-C i tak dalej, więc zawsze jest pęk kabli w plecaku.

lus, jeżeli chcesz pokryć obie platformy – to znaczy iOS i Androida – to raczej musisz posiadać MacBooka. Właściwie nie „raczej", tylko na pewno – to też jest ograniczenie. W przypadku iOS-a aplikację możesz developować tylko na urządzeniu z systemem macOS, a w przypadku Androida nie ma takich ograniczeń – możesz to robić na obu platformach. Nie mając MacBooka, możesz tworzyć aplikację na Windowsie, ale musisz mieć dostęp do systemu macOS. Są różne techniki, żeby to zrobić. Więc tak: musisz poczynić inwestycje, jeżeli chcesz tworzyć aplikację na iOS-a.

No dobrze, to w ten sposób przeszliśmy do kolejnego pytania. Wspomnieliśmy już o tym, że my mamy Androida, iOS-a, mamy dwa różne sklepy – pewnie możemy to obsłużyć za jednym razem. Powiedz nam, jakie są podejścia do tworzenia aplikacji mobilnych, mam tu na myśli m.in. platformy hybrydowe czy cross-platformowe. Jakie są różnice i czy w ten sposób jesteśmy w stanie ominąć temat kupowania MacBooka, czy jednak nie?

Nie jestem specjalistą od każdego z podejść, obracam się wokół świata Xamarina, Fluttera i samego Androida w podejściu natywnym (za chwilę wyjaśnię, o co chodzi), więc nie chciałbym głęboko wchodzić w aplikacje webowe, bo nie chcę mądrzyć się na temat, którego kompletnie nie znam. To podejście przedstawię więc wysokopoziomowo.

Najbardziej domyślne jest podejście natywne. Oznacza ono, że wykorzystujemy narzędzia i język wspierany na konkretnej platformie. I tak w przypadku iOS-a mamy język Swift (wcześniej to był język Objective-C), a w przypadku Androida mamy Kotlin (wcześniej Javę). Kotlin z Javą są dość mocno wymienialne – to znaczy Kotlin na końcu i tak kompiluje się do kodu javowego. Kotlin jest jakby następcą Javy, jeżeli chodzi o ekosystem Androida. Tak, w przypadku podejściu natywnego używamy więc narzędzi dostarczonych przez twórców. Jest to jakby coś najbliżej platformy, to znaczy wydajność takich aplikacji jest najwyższa w porównaniu do wszystkich innych podejść i jest to najlepszy sposób na wykorzystanie funkcji specyficznych dla konkretnej platformy, z którą pracujemy.

Co najlepsze, na bieżąco mamy dostęp do najnowszych narzędzi – nie mamy żadnych nakładek, tylko korzystamy z tego, co bezpośrednio udostępnili nam twórcy. Jeżeli pojawi się nowa funkcjonalność, nowy sposób tworzenia widoków, to od razu mamy do tego dostęp, więc możemy z tego na bieżąco korzystać. Ale właśnie: zazwyczaj firmy tworzą aplikację na obie platformy w tym samym czasie, co oznacza, że prawdopodobnie musimy mieć plus minus dwa razy więcej programistów – bo połowa będzie pracowała z Androidem, a druga połowa z iOS-em. De facto oba zespoły pod względem funkcjonalności będą tworzyły tak

naprawdę tę samą aplikację. To jest potencjalna duplikacja. Problem ten rozwiązuje technologia cross-platform i podejścia hybrydowe.

Zacznijmy od cross-platformu. Te aplikacje są tworzone przy użyciu takich frameworków, jak na przykład React Native. Jeżeli ktoś pracuje z webem, to na pewno coś tam już o nim słyszał. Kolejnym przedstawicielem aplikacji cross-platform jest Xamarin oraz Flutter. Wszystkie one pozwalają nam napisać kod jednokrotnie i przygotować aplikację dla obu platform właśnie za pomocą tego jednego kodu. To jest super, bo tworzymy funkcjonalność raz, utrzymujemy ją w tym samym czasie, a jeżeli pojawi się jakiś błąd, który psuje jakąś funkcjonalność na obu platformach, fiksujemy wyłącznie raz. Z założenia potrzebujemy mniej osób, żeby taką aplikację rozwijać, a mamy dwa urządzenia docelowe. I co ciekawe wiele z tych platform pozwala też tworzyć aplikacje na inne platformy, chociażby na desktop. Flutter również pozwala uruchomić taką aplikację internetową, więc to też jest super.

Ich skuteczność polega na tym, że piszemy kod raz i uruchamiamy go na wielu platformach. Z grubsza te narzędzia działają w ten sposób, że tłumaczą kod, który napisaliśmy w konkretnym języku dla konkretnego frameworka, na kod natywny. Czyli de facto trochę tłumaczą to, co napisaliśmy w tym konkretnym języku, na kod natywny, czyli ten, który jest bliski platformie. Zazwyczaj zapewniają wydajność bardzo zbliżoną do aplikacji natywnych i na pewnym poziomie skomplikowania aplikacji różnica nie będzie widoczna, natomiast trzeba być świadomym, że to nie jest dokładnie to samo, co podejście natywne. Wydaje mi się jednak, że w przypadku większości aplikacji jest to dość pomijalne. Korzyść z zaoszczędzonego czasu i pieniędzy (dzięki temu, że będziemy potrzebować mniej osób i mniej zasobów), jest na tyle duża, że możemy

pozwolić, by ta aplikacja minimalnie gorzej chodziła. Wydaje mi się więc, że to jest całkiem akceptowalne.

Mamy też aplikacje webowe, czyli takie, które tworzymy w JavaScripcie, w jakiś tam frameworku, i możemy uruchomić je po prostu w przeglądarce swojego telefonu. To też jest aplikacja. Ona nie jest, co prawda, instalowana na urządzeniu… Chociaż – tu popraw mnie, Mateusz, jeśli się mylę – możemy zrobić taki skrót na pulpicie urządzenia, żeby móc taką aplikację uruchomić, prawda?

Tak jest.

Nie jestem specem od tego, ale mam takie poczucie, że można to zrobić, więc jest to trochę takie udawanie, że aplikacja jest zainstalowana.

Ostatnim rodzajem są aplikacje hybrydowe – one trochę łączą podejście natywne i sposób tworzenia aplikacji internetowych. W większości przypadków korzystamy z JavaScriptu i opakowujemy go w pewien kontener. Sam kontener jest uruchamiany w podejściu natywnym, a aplikacja napisana w JavaScripcie komunikuje się z tym kontenerem i pozwala na używanie rzeczy, które są dostępne w systemie – jest to takie trochę uproszczenie. Wydaje mi się, że aplikacje hybrydowe są najmniej popularne, ale też nie miałem okazji z nimi pracować.

Mamy więc różne podejścia i każdego z nich używamy zależnie od tego, jaką aplikację tworzymy i jaki jest cel firmy, w której taki projekt jest tworzony.

Tak zastanawiam się nad minusami rozwiązań cross-platformowych, bo pewnie jakieś są, skoro nie wszyscy wykorzystują tego typu rozwiązania.

Tak. I znowu: to zależy od projektu, który tworzymy. Jakie są różnice względem tych różnych podejść? Podejście natywne, jak wspomniałem wcześniej, dostarcza nam możliwość korzystania z tego, co twórcy przygotowali tu i teraz. Wypuszczają nową wersję – my już mamy do tego dostęp. W przypadku aplikacji cross-platform i hybrydowych zazwyczaj musimy trochę poczekać na to, żeby twórcy tych rozwiązań dostarczyli nam nową wersję środowisk, abyśmy byli w stanie korzystać z tych nowych rozwiązań, które twórcy systemów wprowadzili w ostatnich aktualizacjach. Więc to jest ta jedna różnica: zazwyczaj w cross-platformie będziemy trochę opóźnieni względem podejścia natywnego, jeżeli chodzi o nowe funkcjonalności.

Co więcej, mamy pewnego rodzaju ograniczenia, jeżeli chodzi o używanie funkcjonalności telefonu, chociażby jakichś sensorów, kamery, np. w przypadku Apple będzie to Face ID, Touch ID i tak dalej. To są rzeczy, które w przypadku podejścia hybrydowego mogą być trochę problematyczne w używaniu. W przypadku cross-platformu istnieją już gotowe rozwiązania, z których możemy skorzystać, i one komunikują się jakby pod spodem. Jest jednak czasami pewien problem, żeby w podejściu cross-platform i hybrydowym używać tych rzeczy specyficznych dla platformy. Natomiast teraz platformy są już na tyle dojrzałe, że raczej jest to pomijalne. Trzeba jednak mieć na uwadze to, że w przypadku hybrydy i cross-platformu musimy czekać nowe wersje.

Wcześniej wspomniałam o wydajności. Jest ona raczej pomijalna dla większości aplikacji, więc wydaje mi się, że zwyczajnie trzeba być tego świadomym, że podejście natywne będzie prawdopodobnie najszybsze. Cross-platform będzie troszeczkę wolniejsze, ale to wszystko zależy też od tego, jakimi jesteśmy programistami, jak umiemy pracować i na ile znamy daną platformę i konkretne podejście. Wydajność więc też jest różnicą pomiędzy tymi podejściami.

Wydaje mi się, że to są takie główne różnice, też trzeba być świadomym, że na przykład w przypadku Xamarina korzystamy z innego języka programowania. W przypadku Fluttera korzystamy z innego języka programowania. I Flutter, i Xamarin to przedstawiciele nurtu cross-platform. Jeśli chcemy tworzyć aplikacje za pomocą JavaScriptu, to to jest inny język niż Android wykorzystujący Kotlin i iOS wykorzystujący Swift – więc znowu: różnica jeżeli chodzi o języki. Jeżeli zdecydujemy się na jedną z platform, to poniekąd – gdy będziemy chcieli zmienić podejście – to pewna wiedza zostanie (jak mówiłem na początku), ale jest jeszcze różnica językowa. Wydaje mi się więc, że to są takie główne różnice, plus cross-platform daje oszczędność czasu przy samym wytwarzaniu.

W takim razie uznajmy, że chcemy programować aplikacje mobilne. W jaki sposób powinniśmy podejść do naszej ścieżki nauki i jakie sposoby mógłbyś polecić?

Wziąłbym pierwszą lepszą książkę na temat którejkolwiek z platform czy podejścia (to już zostawiam osobie, która chciałaby zacząć tworzyć aplikacje mobilne) – żeby po prostu wejść w ten temat. Najlepiej, by była

ona z dobrymi ocenami. Jestem fanem płatnych materiałów. Wydaje mi się, że osoby, które chcą, abyśmy im za to zapłacili, raczej muszą dać od siebie trochę więcej. Zakładam, że takie materiały są bardziej jakościowe. Darmowej wiedzy w internecie jest mnóstwo, tylko czasami trzeba spędzić trochę więcej czasu, żeby sobie to wszystko poukładać i wytyczyć konkretną ścieżkę.

Jestem więc fanem tego, żeby kupić sobie jakiś kurs online, jakąś książkę. Na sam początek polecam kursy na Udemy – one są naprawdę fajne, tylko że niektóre tematy poruszają bardzo pobieżnie, więc traktowałbym je jako punkt wejścia.

To, co sam polecam zaraz po tych pierwszych rzeczach (książka, kurs), to swój własny projekt. I to jest coś, co ja megadobrze wspominam. Uwielbiam rozwijać swoje projekty. Stwórz własny projekt, wymyśl sobie, co chciałbyś stworzyć, i po prostu zacznij to tworzyć. To jest takie proste, ale dzięki temu poznasz zasadę tworzenia takiego projektu; będziesz wiedział, z czym to się je; dowiesz się, jakie są problemy przy takim projekcie. Jednocześnie będziesz product ownerem, czyli właścicielem takiego produktu, więc będziesz mógł wytyczyć jakąś ścieżkę tego projektu. Będziesz jednocześnie gościem, który musi zaprojektować interfejs użytkownika takiej aplikacji, a to jest coś, nad czym ja swoją drogą spędzałem zawsze najwięcej czasu: „a ten przycisk może 3 piksele tu, 5 pikseli tam”. Jest to więc świetna zabawa. Będziesz jednocześnie marketingowcem, jeżeli uznasz, że taką aplikację chcesz wypuścić do szerszego grona publiczności. Trzeba będzie się dowiedzieć, jak taką aplikację sprzedać, jak ją pokazać. To wszystko będzie fajnie się ze sobą zazębiać i w końcu sprawi, że będziesz tym milionerem (mam nadzieję).

Nauka przez praktykę to jest coś, co ja bardzo rekomenduję, bo wydaje mi się, że można przeczytać 10 książek o programowaniu aplikacji mobilnych, a na końcu i tak nic nie wiedzieć, bo tego nie przećwiczyłeś. Zacząłbym więc własny projekt nawet zaraz po tym, jak poznałem podstawy jakiejś platformy, jakiegoś frameworka, i na tym się skupił. To też oznacza, że budujemy sobie wiedzę – i to jest świetne. Osoba, która prowadzi rekrutację na stanowisko programisty, czuje, czy kandydat już coś robił, czy tylko zna teoretyczną wiedzę z książek. To się zwyczajnie czuje. To jest wyrażone w języku, w sposobie myślenia i w formułkach, które wielu rekruterów już słyszało – wiedzą, że to pochodzi z takiej i takiej strony, że w ten sposób tam było napisane. Więc praktyka, praktyka i praktyka.

Wydaje mi się, że taka osoba, która chce tworzyć aplikacje mobilne, może poszukać projektów open-source na GitHubie – jest tego mnóstwo, więc można szukać np. konkretnej platformy po jakichś hashtagach, po słowach kluczowych. To też jest jakiś pomysł w kontekście tworzenia własnego projektu.

Można też spróbować odtworzyć jakąś istniejącą aplikację – Facebook, Messenger, cokolwiek, co jest popularne i wiem, jak działa. Dzięki temu łatwiej będzie przenieść taką aplikację do swojego kodu. Nie będę musiał się zastanawiać, co stworzyć, tylko po prostu to skopiuję. Może brzmi to jak takie bezmyślne kopiowanie, ale pewna faza – czyli szukania projektów, projektowanie interfejsu użytkownika – jest już za nami, więc możemy się skupić na tym, żeby lepiej poznać platformę za pomocą jakiejś konkretnej aplikacji.

W kontekście tego, co powiedziałem na samym początku: swego czasu uczyłem się w takim serwisie, który teraz nazywa się Kodeco. Co prawda materiały są dość drogie, ale jeżeli chodzi o aplikacje mobilne, to wydaje mi się, że jest to trochę takie imperium. Jest tam mnóstwo tematów z Fluttera, iOS-a, Androida. Chyba znalazło się tam nawet Unity do tworzenia gier. Jest tam mnóstwo materiałów i co ciekawe, również ścieżek. Myślę, że warto tam zajrzeć: kodeco.com.

Przede wszystkim więc praktyka, ale to chyba dość oczywiste. Chociaż mam wrażenie, że osoby, które zaczynają naukę, zapominają o tym i wpadają w tak zwany *tutorial hell*, czyli tylko: oglądamy, czytamy i w zasadzie nie praktykujemy. Przestrzegamy przed tym, bo to zazwyczaj nie prowadzi do osiągnięcia celu, czyli znalezienia pracy. Trzeba praktykować.

Tak, wydaje mi się, że to jest też trochę taki strach: „chyba nie do końca dobrze napiszę ten kod, nie do końca jeszcze pamiętam, jak to było, jak użyć tej składni, jak użyć tej kontrolki". Ten strach z tego wynika – z takiej niewiedzy i niepewności – ale zwyczajnie trzeba spróbować i pozwolić sobie na błędy. Bo tych błędów będzie mnóstwo przez najbliższe lata, nawet jak już będziesz doświadczoną osobą, więc trzeba po prostu nauczyć się z nimi żyć.

Tak, zastanawiam się, czy w ogóle da się programować bez błędów. Ja już trochę programuję i ciągle mi się pojawiają.

Chyba nie.

No dobrze, to powiedz, proszę, jakie języki, narzędzia cieszą się największą popularnością, jeśli chodzi o aplikacje mobilne. Czy widzisz zmieniające się trendy? Może coś warto odpuścić, warto pójść w inną stronę?

Generalnie nie jestem gościem, który jakoś specjalnie obserwuje rynek. Poruszam się w dwóch podejściach – czyli natywnym i cross-platformowym – i wydaje mi się, że to jest coś, co przez najbliższe lata będzie przodowało. Czuję więc trochę, że jestem zabezpieczony, jeżeli chodzi o moją wiedzę i to, czy będę miał przy czym pracować. Natomiast jak z czystej ciekawości sprawdziłem popularność, np. Fluttera i React Native (bo wydaje mi się, że to są jedne z popularniejszych podejść cross-platformowych), to zauważyłem, że one są bardzo do siebie podobne – to znaczy popularność jest stosunkowo bardzo podobna w Google Trends. Jeżeli więc chodzi o same frameworki javascriptowe, to wydaje mi się, że większość rynku pracuje właśnie z tym. W związku z tym React dla osób, które już znają JavaScript i chcą wejść w świat aplikacji mobilnych, będzie na sam początek lepszym rozwiązaniem.

Moim zdaniem Flutter to jest najbliższa przyszłość. Co go wyróżnia względem innych technologii? Możemy pisać na kod cross-platform – to już wiadomo – ale on umożliwia skorzystanie z projektów natywnych, czyli dokładnie takich samych, jakich byśmy używali do tworzenia aplikacji w podejściu natywnym. Dzięki temu możemy wprost używać bibliotek gotowych już na danej platformie – czyli tych milionów bibliotek, które są już gotowe i rozwijane przez lata. Możemy ich użyć w naszej aplikacji Flutter. Może nie tak bezpośrednio, natomiast jest bardzo fajny mechanizm dostępny we Flutterze, który nam na to pozwala.

I to jest ogromna rzecz, jaka na przykład mi, gościowi, który wywodzi się z Xamarina, mega odpowiada, bo w Xamarinie to jest w zasadzie prawie niemożliwe. Nie chcę mówić „kompletnie niemożliwe”, ale jest bardzo problematyczne. Nie możemy tak po prostu sobie uznać, że bierzemy jakieś gotowe rozwiązanie na iOS-ie i na Androidzie i używamy w aplikacji Xamarine. Musimy przejść cały proces konfiguracji, przygotowywania takiej biblioteki i bardzo często okazuje się, że jest to niemożliwe. Jest nawet taki żart wśród programistów aplikacji pisanych w Xamarinie, że proces przygotowania takiej biblioteki natywnej do użycia we frameworku Xamarine jest estymowany od jednego dnia do plus minus nieskończoności. I to jest prawda, już niejednokrotnie się o tym przekonałem.

Wydaje mi się więc, że cross-platform to jest najbliższa przyszłość. Znowu: wybór technologii zależy od projektu, który tworzymy. To jest jasne. Natomiast wydaje mi się, że większość aplikacji jest na tyle prosta, że nie ma sensu tworzyć ich oddzielnie na obu platformach. Lepiej wziąć taki framework, jak na przykład Flutter, i spędzić mniej czasu oraz pracować z tym prościej – tym bardziej, że większość aplikacji nie jest na tyle skomplikowana, by potrzebować super wodotrysków i integracji bezpośrednio z urządzeniem. A praca z Flutterem jest świetna.

Zauważyłem, że w ostatnim czasie (to też było widać na iOS-ie i Androidzie) pojawił się tak zwany deklaratywny sposób tworzenia interfejsu użytkownika. Jest to swego rodzaju nowinka, bo mamy dwa style tworzenia: deklaratywny i imperatywny. Imperatywny to taki nasz domyślny – wydajemy instrukcję za instrukcją i mówimy, co będzie przypisane do czego. Do tej pory tak były tworzone interfejsy

użytkownika na platformie Android i iOS: tworzyliśmy kontrolkę, stylowaliśmy, dodawaliśmy i było super.

Wydaje mi się, że Flutter spopularyzował podejście deklaratywne. React Native też chyba miał to wcześniej, lecz Flutter spopularyzował to w kontekście mobilek (choć może jestem zamknięty w złotej klatce). W tym podejściu deklarujemy niejako to, jak ten UI będzie wyglądał – robimy pewien szkielet i mówimy, jak ten UI ma odzwierciedlać dane, które dostanie. Te dwa style programowania są trochę inne, trzeba kompletnie inaczej spracować, natomiast to pozwala na zdecydowanie szybsze tworzenie interfejsu użytkownika. Sposób i myślenie są inne, ale wydają mi się znacznie bardziej skuteczne i szybsze. Zaraz po tym iOS prowadził Swift UI, który opiera się na podejściu deklaratywnym. Android również ma już swój Android Compose, wykorzystujący funkcje, które budują UI w bardzo podobny sposób jak ten deklaratywny we Flutterze. Wydaje mi się więc, że to też jest coś, co w najbliższym czasie będzie przodowało.

No dobrze, to może teraz przejdźmy do pierwszego poważnego kroku. Załóżmy, że siadamy przy komputerze i chcemy poczynić pierwsze kroki w programowaniu naszej aplikacji mobilnej. Co zainstalować? Jak szybko zobaczymy wizualne efekty działania naszego kodu? Uznajmy, że to będzie Flutter – wspominasz, że to jest najlepsze rozwiązanie, więc się na nim skupmy.

Wydaje mi się, że potrzebujesz dosłownie pół godziny, żeby móc uruchomić swoją pierwszą aplikację – i to pół godziny w najgorszym przypadku. Decydujesz się na środowisko, masz w zasadzie do wyboru

trzy: będzie to IntelliJ od JetBrains, Android Studio i być może Visual Studio Code. Każdy z nich ma pluginy, które umożliwiają Ci pracę z Flutterem.

Instalujesz SDK, pobierasz go ze strony i w zasadzie za pomocą jednej komendy tworzysz pierwszy projekt, żeby pracować z jedną czy drugą platformą. Może dość mocno to upraszczam, ale w przypadku Androida jesteś w stanie to bardzo szybko zrobić. Być może gdzieś tam po drodze musisz jeszcze zainstalować Android Studio, ale to jest wszystko. Wydaje mi się, że jest to mega, bo pamiętam, że jak kiedyś chciałem stworzyć jakąś aplikację w Android Studio (kiedyś, kiedyś – to było parę lat temu), to zajmowało mi to znacznie więcej czasu, było to znacznie bardziej skomplikowane, a teraz z Flutterem jest to mega proste. Instalujesz środowisko, pobierasz SDK i finito. I jedziesz z tematem.

Ciekawą rzecz powiedziałaś, bo w zasadzie „podłączasz urządzenie i już", ale wiemy, że nie ma jednego urządzenia, tylko jest wiele. Chciałbym więc przejść do tematu, jak sprawdzać działanie naszych aplikacji mobilnych, gdy mamy tylko właśnie jeden smartfon. Tutaj zaznaczam, że znowu skupiamy się na tych rozwiązaniach, nad którymi Ty pracujesz. Są jakieś narzędzia, które imitują zachowanie różnych modeli i systemów operacyjnych? Na ile się one sprawdzają?

Albo Android, albo iOS – na obu z nich masz albo symulator w przypadku iOSa, albo emulator w przypadku Androida. One mają swoje różnice, ale na takim codziennym poziomie nie jest to ważne. Możesz więc stworzyć sobie na swoim komputerze takie wirtualne urządzenia, które będzie spełniało takie czy inne wymagania – mam na myśli, że

będzie odwzorowywało jakiś realny sprzęt. Możesz więc mieć takie wirtualne urządzenie i bezpośrednio na nim uruchamiać konkretną aplikację.

Nie musisz mieć stosu miliona telefonów, żeby testować swoje rozwiązania, ale jak by nie patrzeć, na koniec dnia warto by mieć jakieś rzeczywiste urządzenie. Są rzeczy, których nie przetestujesz na takim wirtualnym urządzeniu, przykład push notyfikacje. To jesteś w stanie zrobić tylko na urządzeniu rzeczywistym. Oczywiście to wszystko możesz sobie jakoś symulować, natomiast notyfikacje to już raczej na urządzeniu. Plus jeżeli na przykład Twoja aplikacja korzysta z jakichś sensorów, to znacznie łatwiej się to testuje, jeżeli masz urządzenie. Tak to wygląda: można pracować z symulatorem i emulatorem, da się to zrobić.

Czy myślisz, że jeżeli chcielibyśmy wypuścić swoją aplikację, to wystarczą nam właśnie te emulatory czy symulatory, czy jednak warto by przetestować to na największej ilości urządzeń – tak z perspektywy osoby, która po prostu chce coś wypuścić, coś testowego, czym może się pochwalić na rekrutacji. Nie mówię tutaj o komercyjnym rozwiązaniu.

Jeżeli na rekrutacji, to bym się jakoś dłużej nad tym zastanawiał. Stworzyłbym taką aplikację, uruchomiłbym na emulatorze, symulatorze – i to jest spoko. Wydaje mi się, że to jest wystarczające, natomiast jak rzeczywiście rozwija się coś swojego, to lepiej jest to jednak przetestować na urządzeniu. Nie zawsze ufałbym emulatorowi lub symulatorowi, urządzenie to jednak urządzenie.

No dobrze. Mamy już pomysł na tę naszą aplikację i coś zaczęliśmy robić. Czy Twoim zdaniem trzeba poznać tematy takie jak UX, UI czy design, może coś na temat grafiki komputerowej, czy możemy skorzystać z jakichś gotowych rozwiązań typu dość popularny Material UI.

Fajnie by było, ale nie uważam, że jest to jakoś szczególnie wymagane. Dzięki dodatkowej wiedzy na pewno bardziej poznasz konkretną platformę: jakimi się rządzi prawami, czym twórcy się kierują, dodając nowe funkcjonalności do interfejsu użytkownika. Fajnie więc jest to rozumieć, szczególnie jak tworzysz własny projekt i chcesz wydać do sklepu własną aplikację – fajnie jest wiedzieć, jak to zrobić dobrze.

Jeżeli dołączysz do projektu, to komunikacja z zespołem designerskim na pewno będzie łatwiejsza i znacznie bardziej skuteczna – nie będą musieli tłumaczyć Ci każdej jednej rzeczy. Wydaje mi się więc, że to jest bardzo pomocne.

I teraz tak: w internecie jest mnóstwo materiałów. Mega polecam stronę material.io. Jest to ogromne kompendium wiedzy o systemie Material Design – są tam konkretne przykłady kodu, które możesz użyć na iOS-ie, Androidzie Flutterze i być może jeszcze na jakimś innym frameworku – jest to do sprawdzenia. Ogromna baza wiedzy o tym, jak projektować z Material Design.

Jak tak opowiadasz, to zastanawiam się, na ile programista aplikacji mobilnych jest bliski front-endowcowi, a na ile back-endowcowi. Czy jesteśmy w stanie jakoś to porównać typu: 20% back end, 80% front end?

Ja patrzę na to tak, że to jest jedno i drugie, chociaż jak tworzysz cross-platform, to bardzo często używam takiego stwierdzenia jak „back end”. Ta część wspólna to jest trochę taki back end dziejący się gdzieś z tyłu. Może niektóre aplikacje nie mają tego zbyt skomplikowanego i dużego, lecz jest to trochę taki back end. No i jest też front – to się jedno z drugim łączy, to też jest super. Nie wiem, jak bym miał to rozdzielić. To właśnie jest w mobilkach fajne – że możesz trochę popracować tam z tyłu, porobić jakieś przeliczenia, jakąś zaawansowaną logikę, ale przychodzi dzień, że chcesz popracować z UI-em i możesz to zrobić. To jest fajne.

Na początku rozmawialiśmy trochę na temat projektów, więc teraz chyba to jest moment, w którym chciałbym Cię zapytać, jakie projekty powinniśmy zrobić do naszego portfolio, gdzie szukać inspiracji, by nie skończyć z jakąś nudną to-do listą, o której wszyscy mówią, że jest beznadziejna. Ja uważam, że jak byśmy ją podrasowali, to też byłoby OK, bo ta to-do lista może być zamieniona na każdą inną aplikację.

Czy możesz coś podpowiedzieć, coś z fajnych projektów? Wiem, że sam też rekrutujesz, to może powiesz, na co zwracasz szczególną uwagę?

Jeżeli chodzi o szukanie inspiracji, to ja polecam dwa serwisy. Pierwszy to jest Dribbble, a drugi to jest Behance. Mój kolega grafik śmieje się, że to są serwisy, gdzie graficy się dotykają. Jest tam mnóstwo inspiracji i każdy grafik raczej ma tam swoje konto i jakby chwali się tym, co tworzy. Jeżeli więc tworzysz własny projekt, możesz skorzystać z tego serwisu

jako inspiracji. Jest tam też mnóstwo projektów aplikacji mobilnych, różnych wizualizacji. To jest bardzo fajne miejsce, żeby poszukać pomysłów na to, co można byłoby zrobić. Bardzo fajnie się to łączy z jednym z poprzednich pytań o tym, żeby stworzyć własną aplikację mobilną. Można więc w tych serwisach inspiracji.

Kolejna rzecz – też o tym wspomniałem – czyli kopia jakiejś istniejącej aplikacji: wzoruję się i tworzę w tym konkretnym frameworku. To jest więc drugi przykład.

To jest może nie do końca związane z aplikacjami mobilnymi, ale bardziej ze sposobem nauki: żeby dokumentować swoją ścieżkę, np. otworzyć bloga. Nawet o nim nikomu nie mówić, ale pisać o tym, co właśnie tworzymy, i próbować wytłumaczyć komuś, kto uczyłby się czegoś podobnego, w jaki sposób robić to, co sami przed chwilą zrobiliśmy. To pozwala lepiej zrozumieć temat, którego się teraz uczymy, bo kiedy zaczynamy tłumaczyć, zaczynamy widzieć pewne połączenia i luki w naszym rozumowaniu. Nagle się okazuje, że coś, co wydawało nam się zrozumiałe, po próbie tłumaczenia nie do końca takie jest.

Podsumowując: te dwa serwisy plus próby odtworzenia jakiejś istniejącej aplikacji.

No dobrze, to może przejdźmy do tematu, który chyba jest coraz bardziej popularny. Co myślisz o platformach no-code i low-code do tworzenia aplikacji mobilnych? Czy tego typu platformy mają sens i jaki obecnie jest trend? Może obawiasz się tego, że stracisz pracę na rzecz takich rozwiązań?

Użycie technologii powinno zależeć od tego, jaki projekt tworzymy. Ja swego czasu mega się śmiałam z tych platform. Mówiłem: „jak coś takiego może w ogóle nazywać się programowaniem albo wytwarzaniem oprogramowania?”. Ale po jakimś czasie trochę to sobie ułożyłem w głowie i wydaje mi się, że jeżeli tworzysz mega prostą aplikację, która jest mega odtwarzalna (np. masz wyświetlić ekran, wyświetlić dane, przenawigować użytkownika do innego ekranu), to to jest super rzecz. Bo po co zatrudniać stado programistów i kazać im tworzyć jakieś oprogramowanie, robić sprinty itd., jeżeli możemy użyć czegoś, w czym trochę wyklikamy, i to też będzie spełniało nasze wymagania.

Ja nie pracowałem z tymi technologiami, więc nie chcę mówić o ich różnicach względem innych podejść, natomiast wydaje mi się, że jest to jakiś sposób na tworzenie aplikacji. Na pewno nie dla każdego, nie dla każdej firmy, bo niejednokrotnie chcemy, żeby aplikacje były spersonalizowane, żeby to było dokładnie to, czego oczekujemy my jako product ownerzy (mam na myśli biznes).

Ja nie mam styczności z tego typu platformami, jeśli chodzi o tworzenie aplikacji mobilnych, ale korzystam Make.com. Jest to właśnie platforma no-code i jestem zachwycony, że mogę coś takiego wykorzystać. Jeśli miałbym napisać program, który np. obsługuje mi ten podcast – wysyła maile, zbiera informacje o pytaniach i tak dalej – to pewnie zajęłoby mi to miesiąc albo parę. A na wyklikanie tego zeszło mi, powiedzmy, parę dni.

Pewnie dla takich wewnętrznych rozwiązań jest to super i ja z tego korzystam, ale tak się zastanawiam: czy aplikacje mobilne na tę chwilę są potrzebne wewnętrznie? To może być fajne rozwiązanie,

ale jeżeli miałbym wypuścić coś takiego dla klientów, to pewnie by się nie sprawdziło. Nie wiem, będę sprawdzał i mam nadzieję, że zobaczę wtedy, czy faktycznie ma to sens. Może na to też potrzeba trochę czasu – żeby te rozwiązania odpowiednio dojrzały i wtedy faktycznie będzie można robić aplikacje dla klientów.

No dobrze, to kolejny etap. Mamy już napisany projekt i teraz chcemy go opublikować. Jak wygląda taki proces publikacji aplikacji w najpopularniejszych sklepach?

Więc tak: spędzamy mnóstwo czasu, żeby stworzyć taką aplikację. Mamy ją gotową, mamy przygotowany plik, tak zwany artefakt, czyli to, co chcemy wrzucić do sklepu. Musimy stworzyć stronę aplikacji – uzupełnić w dashboardzie konkretnego sklepu listing naszej aplikacji, czyli opisać, czym ta aplikacja jest, do jakiej kategorii należy, wrzucić zrzuty ekranów. Jest cała masa rzeczy, które musimy przygotować przed, żeby w ogóle proces weryfikacji przeszedł pomyślnie. Wiele rzeczy jest rekomendowanych.

To dopytam: czy to musi być tak mega szczegółowo? Czy jeżeli napiszę, że to jest moja aplikacja testowa, chcę tylko zobaczyć, czy się opublikuje, to coś takiego przejdzie czy nie?

Niekoniecznie. Na pewno musisz wskazać, o czym ta aplikacja jest i nie może to być „test, test, test, test”. Jestem pewien, że to nie przejdzie. Nawet jeżeli jest to Twoja aplikacja testowa, to na pewno nie przejdzie do użytku publicznego, żeby była dostępna w sklepie. Chociaż może jest jakaś szansa, że ktoś Ci to przeklika, ale generalnie musisz uzupełnić

informacje o aplikacji. Na pewno to nie jest coś, co osoby, które recenzują tę aplikację przed jej wypuszczeniem i zatwierdzeniem, dokładnie czytają, ale muszą tam się znaleźć jakieś podstawowe informacje. Jest to więc taki proces przygotowawczy.

Potem jest proces review, o którym wspomniałem, czyli Apple Store i Google Play sprawdzają, czy aplikacja jest zgodna ze standardami, czy jest bezpieczna itd. Mają swoje procesy weryfikacyjne. Po tym, jak aplikacja została zatwierdzona, możemy ją udostępnić dla wszystkich użytkowników. Jeżeli została odrzucona, to musimy poprawić to, co zostało zgłoszone. Każdy ze sklepów umożliwia różne formy dystrybucji: możemy udostępnić aplikację dla konkretnych użytkowników, dla wszystkich, dla jakiejś części. Jest mnóstwo opcji personalizacji tego procesu, możemy go dla siebie dostosować.

Co ciekawe, na obu sklepach mamy komentarze – chyba każdy jest tego świadomy. Może to być zarówno świetna rzecz, jaki i bardzo destrukcyjna dla samej aplikacji, i to na długi czas. Mam tu na myśli to, że osoby, które dodały komentarze negatywne, raczej rzadko chcą je zmienić. Nawet jeżeli ten babol, o którym wspomnieli, został poprawiony, to prawdopodobnie ten komentarz zostanie na wsze czasy. Ma to też swój negatywny skutek. Aplikacje mobilne w sklepach mają system ocen, więc jeżeli raz taka aplikacja oberwie, to bardzo trudno jest jej się wykaraskać i wskoczyć na wyższy poziom. Jest to bardzo trudne. To trzeba mieć na uwadze. Jednocześnie to też jest jakiś benefit – możemy rywalizować z innymi na podstawie jakości.

Myślałem, że powiesz, że można to wyłączyć, ale rozumiem, że nie.

Nie, bardzo bym chciał, ale nie.

A czy płacimy za każdy review i tak dalej?

Nie, jeżeli chodzi o liczbę aplikacji, to nie, natomiast dostęp do sklepów jest płatny. I o ile w przypadku Google Play jest to chyba nadal dwadzieścia pięć dolarów za dostęp lifetime (na zawsze), o tyle w przypadku Apple jest to oczywiście troszkę droższe i kosztuje w podstawowym planie dziewięćdziesiąt dziewięć dolarów rocznie – jest to więc taka subskrypcja. Pozwala nam to udostępnić aplikację w sklepie.

Jest jeszcze wyższa wersja, która pozwala nam taką aplikację udostępnić na przykład wewnętrznie, we własnym sklepie, lub po prostu jako taki plik, który możemy zainstalować na swoim urządzeniu. To kosztuje więcej. W przypadku Androida nie musimy tego robić – po prostu budujemy swoją aplikację, wrzucamy na dowolny telefon i możemy ją zainstalować. W przypadku iOS-a nie możemy tego zrobić i zawsze musimy mieć któryś z płatnych planów. Są to więc takie koszty stałe lub jednorazowe w zależności od tego, na co tworzymy.

A czy chciałbyś coś powiedzieć na temat Huawei? Była taka historia, że Google zbanował urządzenia Huawei i tworzone dla nich aplikacje. Musimy teraz zainstalować dedykowaną aplikację (AppGallery) i wtedy przez nią instalować dodatkowe aplikacje. Czy się tym bawiłeś? Czy tutaj proces wygląda inaczej, czy w ogóle warto iść tą drogą?

Ja mam silne poczucie, że tak, warto, bo to jest trzeci ogromny sklep. Jak wchodzę sobie tam od czasu do czasu, to widzę, że te aplikacje mają miliony pobrań, nawet te polskie aplikacje. Wychodzi więc na to, że jest to dość spory rynek. Nawet ja sam jestem bardziej przyzwyczajony do tego, że aplikacja to raczej Google Play oraz App Store, i jeszcze nie

myślę o Huawei AppGallery. Wydaje mi się, że to jest miejsce, które jeszcze samo nie przychodzi nam na myśl, ale to jest coś, co chyba może się wkrótce zmienić, bo będzie coraz więcej użytkowników. To na pewno całkiem fajna alternatywa i konkurencja do tych dwóch zastanych sklepów, do App Store i Google Play.

Ale tak, nie miałam okazji dystrybuować żadnej aplikacji do tego sklepu. Był moment, w którym próbowałem zrozumieć, jak on działa, jak wygląda proces i na co zwracają uwagę, natomiast nie miałam okazji wydać pełnej aplikacji do sklepu Huawei.

Okej, no to mamy już tę naszą aplikację, zrobiliśmy pierwszy projekt, wszystko działa, ale pewnie nie raz popełniliśmy jakieś błędy. Czy możesz opowiedzieć nam o ewentualnych błędach przy nauce, podczas poszukiwania informacji, czy właśnie wypuszczania takiej aplikacji mobilnej, żebyśmy nie musieli ich popełniać sami?

Skupię się trochę na osobie początkującej, jeżeli chodzi o samą naukę programowania apek mobilnych. Wydaje mi się też, że to będzie coś bardzo podobnego, nawet jak byśmy się uczyli jakiejś innej technologii.

Na pewno pierwszym błędem jest nauka wszystkiego naraz: „ja chcę już poznać iOS-a, Androida, podejście za Xamarin, Flutter, jeszcze React, bo muszę być ze wszystkim na bieżąco”. To jest ogromny błąd. Fajnie jest to poznać, tak jak wcześniej wspomniałem: połączyć te kropeczki i tak dalej, natomiast na początku skupiłbym się na jakiejkolwiek z platform. Polecam cross-platform, bo siłą rzeczy prędzej czy później i tak

będziemy musieli dotknąć kodu natywnego. Na pewno jednak skupiłbym się na jednej platformie lub jakimś podejściu i przez jakiś czas pracował tylko z tym, a po drodze uczył się, jak to działa na iOS-ie, na Androidzie, ale też nie wchodziłbym bardzo w szczegóły.

Na pewno nie skupiałbym się na każdej jednej nowince, która się pojawia w świecie mobilnym, bo mam wrażenie, że czasem jest to taki miód na uszy: fajnie byłoby pracować z najnowszymi rzeczami, Android właśnie wypuścił coś nowego, tu iOS ma taki fajny framework, jest taka fajna biblioteka… Jestem zdania, że fajnie być świadomym, że coś takiego jest, że coś takiego powstało, wiedzieć, po co to jest, ale nie traciłbym za każdym razem czasu, żeby już teraz się tego uczyć. Warto zbierać sobie takie informacje i wiedzieć, że jak będzie potrzeba, to sobie tego użyję. W ten sposób bym do tego podchodził.

Plus teoria ponad praktykę: mniej czytania, więcej robienia. Miałam powiedzieć, na czym się można skupić na samym początku, jeżeli chodzi o podstawy na jakiejkolwiek platformie – to są to takie niezmienne klocki, z których składają się aplikacje. Nauczyłbym się, jak stworzyć podstawowy widok; jak taki widok skomunikować z kawałkiem jakiejś logiki biznesowej; dowiedziałbym się, jak zaktualizować na przykład labelkę, taki *text view*; jak to zaktualizować na podstawie danych tego bloczka logiki biznesowej; jak przechowywać dane; jak użyć jakiejś lokalnej bazy danych. I znowu: w aplikacjach mobilnych są zaawansowane rozwiązania jeśli chodzi o bazy danych, natomiast na samym początku skupiłbym się na jakimkolwiek zapisie do pliku, użyciu jakiejś biblioteki, która na to pozwala

Kolejny krok to komunikacja z API. Jest mnóstwo darmowych serwerów, które pozwolą nam coś pobrać do naszej aplikacji. Skupiłbym się więc na znalezieniu najpopularniejszej biblioteki, która pozwala na komunikację z serwerem, spróbował pobrać trochę tych danych i wyświetlić w interfejsie użytkownika.

Skupiłbym się na uniwersalnych rzeczach, czyli dobrych praktykach, czystym kodzie, w jaki sposób przygotować architekturę aplikacji. Na samym początku nie skupiałbym się na tym jakoś bardzo mocno, ale są pewne wzorce, z których warto korzystać. Warto poznać nawet same podstawy. Jeśli uznamy, że jednak to nie jest ta technologia, którą chcemy się zajmować, to na pewno te praktyki będziemy wykorzystywać gdzieś indziej. Nie skupiłbym się więc bardzo mocno na super wzorcach projektowych i tak dalej, ale na podstawach, które są przenoszalne pomiędzy technologiami.

Poczytałbym trochę o pracy w zespole, szczególnie kiedy taka osoba dopiero szuka swojej pracy. Poczytałbym o tym, jak praca w zespole wygląda, jakich technik się używa – szczególnie chodzi mi o techniki wytwarzania programowania, chociażby jakieś podstawy, żeby wiedzieć jakkolwiek, z czym to się je.

Kolejna rzecz to logiczne myślenie. Ja na przykład, szczególnie kiedy prowadzę rekrutację, bardzo zwracam na to uwagę, zwłaszcza jeżeli są to osoby początkujące. Mniej interesuje mnie, czy ta osoba zna dobrze język programowania. Wydaje mi się, że to jest coś, czego dość szybko można się nauczyć, i traktuję to trochę jako wtórne. Może trochę upraszczam, ale traktuję to raczej jako drugorzędną rzecz. Znacznie ważniejsze jest dla mnie to, czy osoba rozumie, o czym mówi, a to

można zrobić wyłącznie, moim zdaniem, przez praktykę. Więc projekty, projekty, projekty, logiczne myślenie i wyciąganie wniosków z błędów, które popełniłem we własnym projekcie. Raczej na tym bym się skupił.

Czyli dość podobnie, powiedziałabym, to każdej innej dziedziny.

Tak, wiesz, ja jestem fanem próbowania różnych rzeczy, ale nie skupiania się konkretnie na technologii jako takiej. Fajnie jest znać zawiłości technologii, natomiast wydaje mi się, że ponadczasowe są właśnie dobre praktyki, logiczne myślenie i tego typu rzeczy. Jak czegoś nie wiem z danej technologii, to bardzo szybko mogę to sobie znaleźć, ale logicznego myślenia się nie nauczę, jeżeli nie będę go codziennie ulepszał.

Jak już jesteśmy na etapie rekrutacji, to czy możesz powiedzieć nam, jak wygląda rynek programistów aplikacji mobilnych? Czy jesteśmy w stanie dostać pracę bez doświadczenia komercyjnego?

Na pewno rynek mobilny jest mniejszy od jakiegokolwiek innego, programistów aplikacji mobilnych jest mniej niż tych, którzy zajmują się JavaScriptem. Odpowiadając na drugą część pytania: tak, bo aplikacje mobilne to nie jest kompletnie coś innego. Zawsze są osoby, które nigdy wcześniej nie pracowały w IT, więc zawsze jest, że tak powiem, jakiś punkt zero, który muszą przejść.

Jeżeli chodzi na przykład o stawki, jak tak sobie patrzyłem po średnich i po raportach w ostatnim czasie, to nie są to górne widełki w porównaniu z innymi technologiami, ale też nie jest to słabo opłacalna branża.

Wydaje mi się, że to raczej jest taka średnia półka, może taka troszeczkę wyższa. Jeżeli więc chodzi o zarobki, też jest OK.

I teraz tak: kompletnie nie traktowałbym tego jak inny rynek, ale też trzeba pamiętać, że programistów aplikacji mobilnych jest mniej, bo nie każda firma decyduje się na wydanie własnej aplikacji na konkretną platformę. Zazwyczaj każda firma ma aplikację webową, ale z mobilką jest różnie. Nie każda chce w to inwestować, zakładam więc, że to raczej stąd wynika, a nie z braku popularności aplikacji mobilnych, bo smartfona ma każdy. To na pewno nie jest problemem.

A czy myślisz, że to się zmieni? Wydaje mi się, że na razie wygląda to tak, że mamy stronę internetową, która jest aplikacją (tak to nazwijmy), a potem do tego robimy aplikację mobilną, czyli taką dostępną w sklepie. Czy myślisz, że trend się odwróci? Że będzie w drugą stronę: aplikacje dostępne w sklepach będą częściej się pojawiały? To jest jakby moja wizja, nie wiem, czy tak faktycznie jest.

Ja też nie wiem. Na pewno jest pewne to, że znacznie łatwiej i przyjemniej korzysta się z dedykowanej aplikacji instalowanej na urządzeniu niż na przykład z takich uruchomionych przeglądarce. Nie ulega wątpliwości, że taka aplikacja jest zdecydowanie łatwiejsza i przyjemniejsza w użyciu, więc to zależy od projektu i konkretnej aplikacji. Jeżeli mamy coś nastawionego na odczucia użytkownika, to zakładam, że na pewno lepsza będzie aplikacja, którą zainstalujemy i która będzie wyglądała jak taka fajnie połączona z tym systemem, z którym działamy. Nie będzie to aplikacja w aplikacji – mam na myśli przeglądarkę.

Nie wiem, czy to się odwróci. Wydaje mi się, że nie, bo stworzenie aplikacji webowej pozwala Ci uruchomić tę aplikację, mimo wszystko, od razu w przeglądarce na telefonie. Przy wielu biznesach jest to wystarczające, więc znowu wracamy do tego, że to zależy. Nie wiem, nie mam pojęcia, czy to się odwróci. Wydaje mi się, że raczej nie, raczej aplikacje mobilne będą czymś z boku, jako uzupełnienie biznesu.

To zróbmy kolejny krok. Powiedz nam, proszę, czy jesteś w stanie określić takie minimalne wymagania techniczne, które powinien spełniać kandydat, żeby móc otrzymać swoją pierwszą pracę jako programista aplikacji mobilnych.

Chwilę wcześniej powiedziałem o tym, na co zwracam uwagę, jeżeli chodzi o sam język programowania na przykład. Nie zwracam aż takiej uwagi. Chcę, żeby kandydat mniej więcej wiedział, o co chodzić – żeby rozumiał język programowania, znał podstawowe struktury, ale ja zbyt mocno nie magluję kandydatów z tej kwestii. Raczej zadaję pytania techniczne, czyli rzucam jakiś problem; widzę, jak się nad tym głowi; patrzę, jakie rozwiązania proponuje, i szukam tego, w jaki sposób on myśli o danym problemie. Wydaje mi się, że to jest sto razy ważniejsze niż to, czy on zna super najnowsze usprawnienia w danym języku, jakieś tam, że tak powiem, kolorowe dodatki do języka. Na pewno więc na to zwracam uwagę.

Nie jestem też specem od ChatGPT, lecz on rozwiązuje wiele podstawowych problemów i tych zaawansowanych również. Mimo wszystko jednak to, co on wypluje, trzeba potem sobie jeszcze przetrawić i przeanalizować. Trzeba mieć na uwadze, że wiele

problemów możemy rozwiązać za pomocą ChatGPT, więc tym bardziej sprawdzam sposób myślenia kandydatów, a nie to, czy oni użyją takiego czy innego typu danych – tak w uproszczeniu. To się bardzo łączy z tym, żeby tworzyć własne projekty i mieć doświadczenie praktyczne, bo to się zwyczajnie czuje.

Twoją wypowiedź rozumiem w taki sposób, że nie ma co uczyć się teorii. Często słyszę „OK, to skoro zaczynam szukać pracy, to uczę się teorii, czyli szukam pytań rekrutacyjnych i przygotowuję sobie odpowiedzi". Ty tego nie szukasz, tak? Chodzi bardziej o to, żeby praktykować, widzieć zmysł rozwiązywania problemów, a nie odpowiadać na regułki.

Wiesz, też nie chcę tak tego spłycać, bo wiadomo, że to, z czego my na co dzień korzystamy, z czegoś konkretnego wynika. Swój proces nauki trzeba więc zacząć od tego, powiedzmy, szukania odpowiedzi i rzeczy, o których na rekrutacjach się pyta. Chodzi mi bardziej o to, żeby je rozumieć. Ja nie chcę usłyszeć zdania z internetu, ja chcę usłyszeć, jak Ty to rozumiesz. Ja mam taki sposób myślenia.

U osób początkujących jest dla mnie jeszcze szczególnie ważne to, czy czuję u nich jakąś zajawkę, takie chcenie: „ja chcę programować, chcę rozwiązywać te rzeczy, chcę tworzyć oprogramowanie i chcę pracować w zespole". I taką ciekawość. Co więcej, ważne jest dla mnie, żeby znaleźć u kandydata nastawienie na zasadzie „nie wiem, ale się dowiem", a nie stwierdzenie: „Ja już wszystko wiem i teraz jestem programistą 15K. Zatrudnijcie mnie, bo jest bardzo dużo ofert pracy, więc jak nie tu, to sobie pójdę dalej".

Kończąc temat rekrutacji: czy Twoim zdaniem jest coś, co pomoże nam się wyróżnić na tle innych kandydatów? Coś, o czym mało osób pamięta? Wspominaliśmy o tej aplikacji w sklepie. Czy myślisz, że to może być to coś, czy jest coś innego albo dodatkowego?

Tak, na pewno to jest fajny dodatek, tym bardziej że jeśli ktoś robił coś praktycznie, to z taką osobą inaczej się rozmawia (jak już wcześniej powiedziałem). Jeżeli więc taka osoba ma jakąś aplikację w sklepie, to możemy sobie o tym pogadać, to jest fajny temat do rozmowy. Mogę zapytać, jak zrobiłeś to; dlaczego zrobiłeś tak; a jaki miałeś problem i jak go rozwiązałeś; dlaczego akurat taka aplikacja, a nie inna. Mamy fajny temat do pogadania. Jest to miejsce, w którym kandydat może trochę się popisać, bo gadamy o czymś, co on stworzył (mam nadzieję, że to on stworzył).

To jest też, myślę, fajna wskazówka. Wydaje mi się, że na żadnej z moich ostatnich rekrutacji nie miałem okazji porozmawiać o aplikacji, którą ktoś wykonał. Bardzo miło więc bym się zaskoczył, gdybym miał taką okazję.

OK, no dobrze. Czy możesz polecić jakąś książkę, która pozwoli osobom początkującym wystartować z aplikacjami mobilnymi?

Wrócę do tego, że można wziąć jakąkolwiek książkę o aplikacjach mobilnych, żeby wejść ten temat, ale jestem fanem tego, żeby inwestować w takie ponadczasowe tematy. Poleciłbym więc książkę *Czysty kod* Roberta Martina, żeby poznać dobre praktyki, to, jak

pracować z kodem, żeby sobie go układać tak, żeby to było zrozumiałe, gdy wrócę do tego kodu za pół roku.

To tylko dodam, że u nas w podcaście ta książka pojawiała się, nie wiem, już z pięć razy.

To będzie szósty raz. Ciekawe, czy była już ta druga, o której za chwilę powiem, czyli *Pragmatyczny programista* – to też jest pewnego rodzaju klasyka. Ona jest może dość trudna, szczególnie na samym początku, dla osób, które dopiero wchodzą do branży, ale bardzo fajnie porusza mnóstwo tematów takiej codzienności programisty i budowania świadomości bycia osobą odpowiedzialną w projekcie IT i tego, w jaki sposób pracować w zespole, jak podchodzić do rozwiązywania problemów i trochę jak walczyć z ego. Ją też bardzo polecam.

A masz może pod ręką autorów? Bo myślę, że ta pozycja też już była, tylko chciałbym się upewnić. Nie wiem, czy mówimy o tej samej, ale ostatnio było wznowienie chyba na trzydzieści lat…

Na dwudziestą rocznicę chyba nawet.

O, dwudziestą, tak jest.

Tak, to jest Pragmatyczny programista. Od czeladnika do mistrza, Andrew Hunt i David Thomas.

To też już było.

Świetna książka swoją drogą, więc mega polecam. I byłbym zapomniał. Jestem też właśnie na etapie kończenia swojej książki o Flutterze. I to też nie jest tak, że opowiadałem Ci o tym Flutterze i go tak wychwalałem, bo piszę o nim swoją książkę, tylko naprawdę uważam,

że Flutter jest świetny. Zachęcam więc do sprawdzenia mojej książki. Co prawda ona nie jest jeszcze teraz dostępna, bo premiera będzie prawdopodobnie jesienią tego roku, ale tak: wydawnictwo Helion, *Flutter. Podstawy*. To jest coś, co jest dopiero w zapowiedzi, ale to jest takie moje malutkie dzieciątko tego roku.

I żeby zwieńczyć temat książek: szukałbym czegoś o uniwersalnych zasadach programowania (*Czysty kod*), coś o architekturze, coś, co jest niezmienne pomiędzy technologiami.

No dobrze, to znowu mamy kilka pozycji do przeczytania.

To tak na sam koniec: gdzie możemy Cię spotkać sieci. Jeżeli ktoś chciałby się o coś dopytać, to gdzie ma Cię szukać?

Punktem startowym będzie mój blog programistabyc.pl. Dla osoby, która będzie chciała coś o mnie znaleźć, jakiś kontakt, jest to dobry punkt startowy. Na co dzień najczęściej piszę na LinkedInie, więc tam można znaleźć moje wpisy – takie, powiedzmy, z dnia codziennego i w ramach mojej inicjatywy budowania świadomości programistów, zwiększania swojej skuteczności i tego, w jaki sposób pracować w zespole lepiej. Mam też taką swoją inicjatywę Dziarski Dev – może kogoś zainteresuje. Ale tak, punktem startowy będzie blog programistabyc.pl.

No dobrze, czyli jeżeli ktoś chciałby zaczynać od programowania aplikacji mobilnych, to może uderzyć w ten sposób do Ciebie.

Ja Tobie, Krzysztofie, bardzo dziękuję za tę rozmowę i podzielenie się z nami swoimi doświadczeniami.

Świetnie, dzięki Mateusz.

Dzięki, cześć.