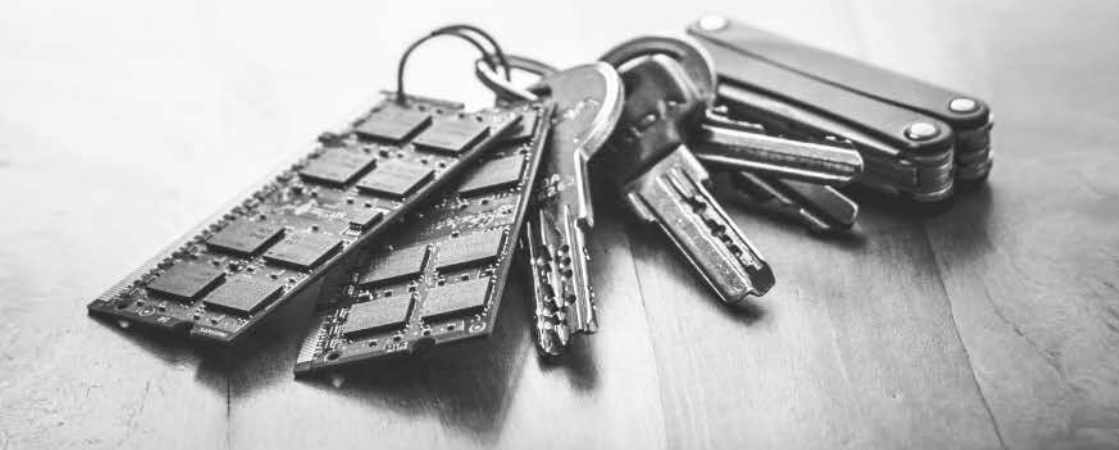
MACIEJ ANISEROWICZ

Zawód: Programista

Wszystko, czego potrzebuje świadomy developer



devstyle.pl

Żonce i Córeczce.

Cierpliwie przeszłyście ze mną tę drogę.

Dziękuję.

Projekt okładki i zdjęcie: Damian Baranowski

Zdjęcie autora: Maciej Korsan

Redakcja i korekta: Joanna Ginter

Redakcja wstępu: Joanna Wrycza-Bekier

Projekt makiety, skład, redakcja: Kaja Mikoszewska

Wydanie pierwsze, Białystok 2017

Spis treści

Od autora

Piszę te słowa na ławce w parku, pod kasztanowcem obsypanym kolorowymi liśćmi.

Piszę te słowa, delektując się jednym z najlepszych dni mojego życia.

Piszę te słowa po sprzedaniu 1300 egzemplarzy jeszcze nieukończonej książki, ze świadomością jej sukcesu.

I... piszę te słowa bez strachu.

Z czystym sumieniem oddaję w Twoje ręce efekt mojej dziesięcioletniej działalności na wielu polach IT w Polsce. Blog, podcast, scena, konkursy, konferencje, szkolenia: to wszystko za mną. Napisanie książki to jedyna poza-programistyczna aktywność, której się jeszcze nie podjąłem.

Do teraz.

Przez lata pracowałem dla klientów polskich i zagranicznych. W firmach małych i średnich oraz wielkich korporacjach. Kodowałem w biurze i zdalnie. Tworzyłem systemy na zamówienie i dla siebie. Utrzymywałem się zarówno z etatu jak i z freelancerki.

Moja droga przez programistyczny świat była i jest wspaniałą przygodą. Zapraszam Cię do przebycia jej razem ze mną. Być może unikniesz pułapek, w które ja wpadłem.

Nie chciałem stworzyć książki łatwej. Pisząc, zadawałem sobie trudne pytania. Ciebie również do tego zachęcam. Założę się, że odpowiedzi Cię zaskoczą. Ba, być może nawet zdecydujesz się na ważne zmiany w swoim życiu?

Przed Tobą siedem części. Każda z nich opisuje inny etap w karierze programisty. Będzie do bólu konkretnie, ale i lekko. Opowiem Ci wiele anegdotek z programistycznego życia. Wszystkie one – nawet te najdziwniejsze – są prawdziwe.

Na początek przyda się ustalić nić porozumienia. Zaproponuję zestaw refleksji o naszej branży. Przedstawię mój punkt widzenia na tematy, o jakich zwykle nie myślimy na co dzień. Te treści nadają ramy reszcie. Nie ze wszystkim się zgodzisz... Taką mam przynajmniej nadzieję!

Druga część pomoże postawić pierwsze kroki tym, którzy dopiero zaczynają. Chcesz zostać programistą lub programistką? A może już powoli startujesz? Tutaj znajdziesz bardzo dużo porad pozwalających ruszyć z miejsca. Natomiast bardziej doświadczeni wyjadacze będą mieli szansę, by z rozrzewnieniem wspomnieć dawne czasy, przypomnieć sobie tamte dylematy i być może zrozumieć nowe pokolenia...na nowo?

Następnie przejdziemy do szukania pracy. Tej najważniejszej, bo pierwszej. Nie wiesz, jak napisać CV? Masz je napisane, ale wyszło szyderczo puste? Nie wiesz, jak dostać się na rozmowę rekrutacyjną i jak ją przetrwać? Tutaj znajdziesz odpowiedzi. Nawet mając bogate doświadczenie możesz zastosować podane porady i odświeżyć swoje résumé! Czy przypadkiem zbytnio się ono nie zestarzało?

Jak często i z jakich powodów warto zmieniać pracę jako programista? Jak ją znaleźć, za czym dążyć, na co uważać? Z części „Nowa praca" dowiesz się, jak ułatwić ten proces. Każdy znajdzie tutaj coś ciekawego.

Bycie programistą wymaga ciągłej nauki, nieustannego rozwoju. Tak zwanego „ostrzenia piły". Tym aspektem naszej pracy zajmiemy się w części poświęconej programistycznemu rozwojowi. Jak, dlaczego i po co to robić? Czego się wystrzegać?

Co, jeśli programowanie przestaje wystarczać? Jak poradzić sobie w nowej roli „team leadera"? To bardzo trudne zadanie i moje doświadczenia zebrane w części „Zawód: Team Leader" mogą Ci znacznie pomóc.

Wreszcie: może by tak rzucić wszystko i pójść na swoje? Ostatnią część poświęciłem „freelancerce". Jak zostać wolnym strzelcem? Co daje ten tryb pracy, a na co warto uważać? W tej sytuacji szczególnie łatwo jest wpaść w pewne pułapki, przed którymi chcę Cię ostrzec.

Na koniec zostawiam rozdział zbierający wszystkie moje wnioski w jedną spójną całość. Drogowskaz na nadchodzące lata. Latarnię oświetlającą nieznaną – ale piękną – przyszłość.

Mam nadzieję, że podróż ta będzie dla Ciebie przynajmniej tak fascynująca, jak była dla mnie.

Zaczynamy! Nie mogę się doczekać.

Maciej Aniserowicz
devstyle.pl
maciej@devstyle.pl

1 Garść refleksji na start

Kto może zostać programistą?

Zostać programistą – w dzisiejszych czasach wydaje się to postanowieniem *trendy* i prostym, możliwym dla każdego. Wiele osób pyta: „czy JA mogę zostać programistą? czy to realne?". Odpowiedź zawsze brzmi: „TAK, jak najbardziej!". Czasami jednak odnoszę wrażenie, że w tym wszystkim brakuje refleksji: „dlaczego chcę nim być? PO CO mi to?

Między „możesz" a „zrobisz!" jest kolosalna przepaść, a droga – długa, trudna i kręta.

Planujących wejście do branży można podzielić na dwie grupy: „na edukacyjnym dorobku" i tych, którzy zajmowali się dotychczas czymś innym.

Licealista/student

W przypadku pierwszej grupy, czyli osób „na edukacyjnym dorobku", sprawa jest bardzo prosta. Oczywiście, że wrota do programistycznego raju stoją przed Tobą otworem!

Naokoło możesz spotkać głosy zniechęcenia. Szczególnie w niektórych zakątkach internetu łatwo o mądralińskich, krzyczących kompletnie bezpodstawnie: *Trzeba zacząć kodować już w podstawówce, żeby być dobrym programistą!!!*

ALE TO BZDURA!

Albo:

Masz w tym roku maturę, a jeszcze nie napisałeś pasjansa/notatnika/gry?

Jeszcze nie spędziłeś setek upojnych nocy, poznając tajniki C++?

Wszyscy programiści byli już w Twoim wieku wymiataczami!

Powtarzam: to bzdura! Owszem, niektórzy kodują od podstawówki. Są nawet tacy, którzy zarabiali już wtedy na tym jakieś grosze. Ale co to oznacza? Że programowanie w podstawówce nie eliminuje kandydatów i kandydatek do zawodu programisty. Czyli oznacza tyle co nic.

A student? Kończysz pierwszy czy drugi rok, a nie czujesz się jeszcze profesjonalistą? Nie ma co jęczeć; po prostu pora zacząć się nim stawać!

Na tym etapie już wiesz, czy to lubisz, czy nie. Już wiesz, czy chcesz dalej w to brnąć. Jeśli czujesz, że to nie jest droga dla Ciebie, po prostu daj sobie spokój. Ale jeśli Cię kusi, to... do roboty!

Ja właśnie wtedy zdecydowałem: „wchodzę w to, *all-in*!". W podstawówce komputer służył mi jedynie do odpalania kolejnych płytek z „CD-Action". W liceum było podobnie – z małymi wyjątkami. Przez połowę studiów też nie robiłem zbyt wiele poza czerpaniem z uciech życia studenckiego. Dopiero na trzecim roku coś mnie tknęło i zacząłem na poważne zastanawiać się, co dalej. Olśnienie. Tak, to dla mnie!

Od tamtej pory, od momentu podjęcia decyzji, uczyłem się praktycznie non stop. A to projekty własne, a to konkursy, praktyki, a to wreszcie próba pracy zawodowej... Pod koniec studiów byłem już całkiem zadowolony ze swojego poziomu. Ale z czystym sumieniem mogę powiedzieć, że:

Ja programistą stałem się dopiero w połowie studiów.

Student czy licealistka pyta: „czy nie jest dla mnie za późno?". Nie, nie jest dla Ciebie za późno! Tylko trzeba przestać się nad tym zastanawiać i zacząć coś robić.

W takim razie... czy później też można?

Przechodzący z innej branży

Druga grupa potencjalnych programistów to osoby mające już wykształcenie, zawód, prawdopodobnie rodzinę. Zauważyły one, że oprogramowanie jest WSZĘDZIE, że bez niego nie ma życia. Gdyby nagle wszystkich programistów na świecie zdjęła jakaś zaraza, ludzkość wkrótce wróciłaby do jaskiń i jeździła na rowerach z kwadratami zamiast kół.

Bez programistów współczesny świat nie istnieje.

Do tego jeśliby jeszcze zajrzeć jednemu czy drugiemu do portfela, to można im czasem pozazdrościć! Media podają, jakie to mają piękne życie za klepanie w klawisze, pochłanianie pizzy i kawy. Też tak chcę!

Proszę bardzo! Drzwi stoją otworem i są przykłady osób, które tego dokonały.

Teoretycznie każdy może zostać programistą! Nie trzeba być nikim specjalnym, by programować.

To praca dla kogokolwiek. Ale nie dla wszystkich!

Potrzeba tylko trochę zawziętości, trochę czasu, trochę chęci, trochę samozaparcia, trochę dyscypliny, trochę predyspozycji, trochę… No właśnie: trochę dużo tego „trochę".

Okazuje się, że to wcale nie jest takie łatwe.

Spróbuj wrócić z pracy i zabrać się do czegoś kompletnie obcego. Do drugiej pracy, która po kwadransie – czy nawet godzinie – niekoniecznie przyniesie rezultaty; przez którą nie będziesz mieć czasu na posiedzenie z rodzinką i obejrzenie nowego odcinka *Gry o tron*. Rób to regularnie, codziennie. Nie przez tydzień, ale miesiącami! Bez żadnego namacalnego efektu.

Skoro już mowa o efekcie, to… jaki miałby on być? Pewnie chcesz uprawiać programowanie zawodowo. Zanim jednak ktoś będzie mógł czerpać faktyczne korzyści z Twoich programistycznych umiejętności, to – nie czarujmy się – sporo wody w rzekach upłynie.

Z ręką na sercu: ja siebie nazwałbym przydatnym programistą dopiero po ponad dwóch latach ciężkiej harówy. To nie były dwie godzinki co kilka dni. To regularna nauka, która trwała po kilkanaście godzin dziennie. Codziennie. Siedem dni w tygodniu. Nauka programowania była wówczas prawie całym moim życiem. Dziś, mając dodatkowe zobowiązania (pracę, dziecko), nie byłbym w stanie tego osiągnąć.

Ale załóżmy, że dasz radę chociaż parę godzin dziennie posiedzieć dodatkowo przy komputerze. I że Ci się to spodoba. Przetrwasz początki, pokonasz frustrację, nadrobisz zaległości i opanujesz podstawy…

Nadchodzi ten moment, kiedy stawiasz pierwszy krok. Wysyłasz CV, idziesz na rozmowę. Aplikujesz oczywiście na najniższe stanowisko, bo nie masz doświadczenia i nadal stosunkowo niewiele umiesz.

I... konkurujesz z rzeszą studentów!

Studentów bez dzieci, kredytów, rodzinnych wczasów, ubezpieczeń... Czyli takich, którzy mogą pracować za kwotę znacznie poniżej Twojego absolutnego minimum. A dodatkowo oni mają o wiele więcej czasu na naukę i dalszy rozwój!

Starość nie radość, a programerka... komu?

Nie tego spodziewasz się po niniejszej książce? Interesuje Cię wyłącznie motywacja, *yes, we can* – i do przodu? Niestety, nie ma czarowania. Już na samym początku trzeba spojrzeć prawdzie w oczy: to nie jest bułka z masłem.

Da się nauczyć programowania i osiągnąć sukces, ale zmiana branży wymaga poświęceń. Potrzeba wiele czasu i dużych ilości odłożonych pieniędzy. Z czegoś trzeba żyć, pracując za studencką stawkę przez kto wie ile czasu.

Jak najbardziej jestem za próbowaniem swoich sił. Trzymam kciuki, kibicuję i biję brawo. Takie kroki jednak trzeba wykonywać świadomie, z otwartymi oczami, realistycznie oceniając swoje możliwości. Hurraoptymizm i podejście „jakoś to będzie" może bardzo zaszkodzić.

Nikomu niczego nie żałuję. Niech i siedemdziesięciolatek próbuje swoich sił. Tylko...

Czy znasz 40-, 50- albo 70-latka próbującego zostać prawnikiem albo lekarzem?

Pewnie nie. A dlaczego?

Bo to trudne i dużo się uczyć trzeba.

I tu jest pies pogrzebany. Ludzie naoglądają się w telewizji programistów siedzących przed kompem, uderzających w klawiaturę – i tyle. Poczytają w gazetach o aroganckich osiemnasto-, dwudziestopięciolatkach, mających mleko pod nosem i zarabiających krocie. I wszystko im przychodzi ot tak, tylko dlatego, że piszą na ekranie jakieś szlaczki.

I TO jest problem!

Chcesz zostać programistą? Proszę bardzo! Ale nie traktuj tego zawodu jako banalnego sposobu na zbicie kasy. To nie jest tylko kwestia postanowienia „od dziś będę programować" i inkasowania dużych wypłat.

Proces rozwoju programisty trwa długo i nie kończy się nigdy.

Szczególnie, jeśli nieprogramista staje się nim na własną rękę, bez pomocy.

Nie każdy musi być dobry, ale...

Owszem, można wyjść z założenia, że wyrobnicy też są potrzebni. I jest to prawda.

Jednak co z odpowiedzialnością? Uświadom sobie, że brak kompetencji to więcej błędów w tworzonym oprogramowaniu. A Twój błąd, wynikający ze słabego przygotowania, może mieć fatalne konsekwencje. Jasne, nawet najlepszym się zdarza. Najpierw jednak trzeba znaleźć się wśród tych najlepszych, by mieć czyste sumienie.

Ewentualne smutne rezultaty nigdy nie są winą tylko i wyłącznie programisty. Ale to...

programista pociąga za spust.

Warto się nad tym zastanowić.

Tak więc... nie pytaj, czy się opłaca. Nie pytaj: „za ile?" ani „czy to dla mnie?". Po prostu zakasaj rękawy – i do roboty!

Przygotowując się do zawodu, zobaczysz, czy jesteś w stanie temu podołać. Nie dowiesz się tego „na sucho". I nikt nie podejmie decyzji za Ciebie.

Dlaczego warto zostać programistą?

Każdy kiedyś jakoś zaczynał. Moja przygoda z programowaniem miała wiele początków.

Jeden z ważniejszych to wybór informatycznych studiów. Lubiłem grać na komputerze, nie kręciło mnie nic poza tym… no to kim mogę być, jak nie programistą?

Bakcyla połknąłem w połowie studiów. Wtedy wsiąkłem całkowicie, bez umiaru, bez opamiętania. I tak mi zostało na baaardzo długo.

Jak postrzegałem ten zawód, kiedy go wybierałem? Na co liczyłem, zarywając noce przez kilka trudnych lat, nieustannie szukając swojego miejsca i chłonąc nową wiedzę, mieszając życie prywatne z zawodowym, tak że nie da się ich już rozerwać? Czego oczekiwałem od codziennej pracy, gdy czyniłem z niej swoje hobby, swoją pasję i swój największy nałóg?

Założę się, że moje wyobrażenia – moje nadzieje! – w znacznej mierze pokrywały się z tymi, które teraz masz Ty. Zacznijmy od odrobiny euforii i ekscytacji!

Co jest takiego SUPER w zawodzie programisty?

Zaraźliwa pasja

Fajnie wierzyć, że IT = pasja! Być programistą to coś więcej, niż klepać kod. To można – i trzeba! – kochać.

Tak naprawdę bywa różnie, ale faktem jest, że programowanie to często coś więcej niż tylko praca. To także hobby, ulubiona rozrywka.

Na każdym kroku spotkasz pasjonatów i pasjonatki. Dla nich pokonywanie kolejnych barier jest esencją tego zawodu. Dyskusje o nowościach, wymiana doświadczeń, wyprawy na konferencje, być może nawet wspólna lektura blogów… Czyż nie po to buduje się zespoły?

Programista wstaje rano pełen energii, z przekonaniem, że dzisiaj będzie bardziej interesująco niż wczoraj… i mniej interesująco niż jutro. Piękne?

Tak, jeśli się sprawdzi. A dokładając do tego magiczne słowo „sztuka", uzyskujemy mieszankę godną pozazdroszczenia.

> **Choose a job you love, and you will never have to work a day in your life.**
> ***Konfucjusz***

'nuff said.

Pieniądze

Zarobki w IT to temat na osobne rozważania i do tego dojdziemy w dedykowanym rozdziale, lecz jedno nie podlega dyskusji: większość programistów nie może narzekać. Przynajmniej na „materialną" część swojej pracy. Zarabianie legendarnych już „piętnastu tysi na rękę" jest często nierealne, ale w porównaniu do reszty społeczeństwa mamy naprawdę dobrze. Nie aż tak rewelacyjnie, jak to się nieraz podaje w mediach, ale dobrze.

Marzyłem, że po latach dziadowania wreszcie będę miał mocny komputer – taki, jaki chcę, a nie z promocji w markecie! Że wreszcie kupię wymarzoną furę! Że wreszcie odwieczny problem, kino czy browar, odejdzie w niepamięć!

I... co z tego wyszło?

Ten temat rozwinę już za kilka stron. A tymczasem lecę popływać w moim skarbcu, jak ten kaczor z bajki.

Stosunkowo łatwo zdobyć pracę

Pewnie znasz ten gryps z rozmową dwóch programistów:

– *Wylali mnie z roboty. Szukałem nowej.*

– *I co?*

– *I to był najgorszy kwadrans w moim życiu!*

Czy to dobrze, czy źle – można na to patrzeć z różnych perspektyw, i zrobimy to w kolejnych rozdziałach. Oczywiście nie każdy i nie wszędzie znajduje pracę w mgnieniu oka. Niekoniecznie też tak popularne hasło: „w Polsce brakuje teraz pięćdziesięciu tysięcy programistów!" oznacza dobre wieści.

Raczej bezpiecznie można jednak założyć, że programiście znalezienie pracy zajmuje krócej niż przedstawicielom wielu innych zawodów.

Różnorodna wiedza

Nie każdy projekt informatyczny jest pasjonujący, ale każdy rozwiązuje jakiś problem. Programista ma okazję dość dokładnie się z tym problemem zapoznać.

Pracując przy systemach finansowych, możemy poznać, jak działają finanse. Tworząc *software* telekomunikacyjny, zdobywamy wiedzę na temat połączeń telefonicznych i sposobu działania centrali telefonicznej. Z kolei implementując reguły panujące w gospodarce magazynowej, łączymy wiedzę zarządcy z wiedzą magazyniera.

I jacy jesteśmy potem mądrzy!

Praca programisty daje szansę poznania bardzo wielu innych profesji. Wystarczy chcieć i wykazać inicjatywę.

DevSpotkania

W każdym większym mieście (i w wielu mniejszych) można znaleźć tzw. meetupy, czyli cykliczne spotkania mające na celu pogłębianie swojej wiedzy. W naszym przypadku chodzi oczywiście o temat programowania. W zależności od regionu w takich przedsięwzięciach może uczestniczyć od kilku do nawet stu osób! Bardzo często są to inspirujące inicjatywy, motywujące do działania.

Środowisko IT obfituje także w większe, bardziej zorganizowane konferencje. Co roku odbywa się ich cała masa. Pojechanie na takie wydarzenie często gwarantuje zastrzyk energii, popychający do działania na długie tygodnie.

Jeśli spróbujesz takich spotkań i wyjazdów, to w bardzo krótkim czasie zdobędziesz dziesiątki nowych znajomych. *Fornever alone*.

Kariera

Lubisz różnorodność? W IT da się ją odnaleźć, wystarczy się postarać. Nuda? Monotonia? Jest bardzo wiele dróg ucieczki przed tymi mordercami.

Ja przez całą swoją karierę staram się uświadomić jak największym rzeszom programistów, że na klawiaturze świat się nie kończy. Ba, on tam się dopiero zaczyna!

Osobiście doświadczyłem tego, jak IT może ukształtować człowieka pragnącego zmian, jakie szanse otwierają się przed aktywnymi programistami. Mnogość opcji powoduje, że klepanie kodu może być tylko przystankiem na cudownej drodze prowadzącej... dokądkolwiek zechcesz.

Wiem, że brzmi to cukierkowo. Ale to prawda. I jeszcze do tego wrócimy.

Elastyczność

Czego programista potrzebuje do pracy? Komputera i – zwykle – internetu. Swoboda, jaką to daje, jest ogromna. Przy odrobinie dyscypliny, dobrej woli i porozumienia z pracodawcą pracować można skądkolwiek i kiedykolwiek.

> **Cyfrowi nomadzi – programiści zwiedzający świat, bez brania urlopu.**

Raz w kawiarni, raz w hotelu, raz na plaży. Mało kto ma szansę na zasmakowanie takiego stylu życia. Nie każdemu też będzie on odpowiadać. Ale jest to opcja zdecydowanie włączająca programistów do wolnych zawodów.

Ta praca może być bardzo przyjazna nie tylko poznawaniu świata, ale też życiu rodzinnemu. Chore dziecko, zamknięte przedszkole, wakacje w szkole? Możliwość elastycznego wykonywania codziennych obowiązków bardzo pomaga we wszystkich tych sytuacjach.

Kreatywna nieskończoność

W dzisiejszych czasach każdy potrzebuje lekarza, prawie każdy prawnika, a niemal wszystkie osoby „kreatywne"... programisty! Kogoś, kto przekuje pomysł w działające rozwiązanie.

Jaką przewagę daje nam umiejętność programowania? Taką, że wszystkie – nawet te najbardziej szalone – pomysły możemy zrealizować samodzielnie! W czasie gdy „normalny" przedsiębiorca spędza tygodnie na poszukiwaniu

wykonawcy swojego pomysłu, my możemy po prostu usiąść i zaimplementować własny.

Możliwości, jakie to daje, są o-grom-ne. Wystarczy z nich skorzystać.

⏎

Dlaczego nie warto zostać programistą?

Czas na kubeł zimnej wody. Zbytni optymizm może być równie szkodliwy jak przesadny pesymizm, dlatego warto zachować po prostu… realizm.

Zobaczymy, że wcale nie musi być tak różowo. Do większości z poruszonych poniżej kwestii wrócimy jeszcze w tej książce. Na razie zerknijmy tylko: na co w zawodzie programisty trzeba uważać?

Nuda

Wcześniej pisałem, że przed nudą można uciekać? Owszem, można. Co nie oznacza, że jest to łatwe!

Programowanie jest cudowne… ale nie zawsze. Często jest to po prostu znój. Pot i łzy. Codzienna programerka może zabić wszelką radość tworzenia. Zdusić kreatywość i odebrać chęci do działania.

A dlaczego? Bo szara, zwykła praca niekoniecznie pokrywa się przez cały czas z wymarzonymi *dev*-wyzwaniami. Nawet w najciekawszych projektach przychodzi czas na monotonne poprawianie błędów. Na użeranie się z niekompletną specyfikacją. Na… klepanie w kółko tego samego.

Wiele lat temu zrozumiałem, że to do mnie należy dbanie o swoją kondycję developerską. To ja sam mam sobie zapewnić rozwój i różnorodność; jeśli nuda i monotonia powoli, lecz stanowczo, uśmiercają mój zapał to… jest to moja wina.

I to JA jestem odpowiedzialny za wprowadzenie zmian.

Nie pracodawca. Nie koledzy. Nikt inny. Ja.

Walka z maszynami

Od jakiegoś czasu nie programuję już zawodowo, etatowo, po osiem godzin dziennie. Gdybym miał podać jedną najważniejszą przyczynę takiego stanu rzeczy, prawdopodobnie postawiłbym na walkę z maszynami.

| **Ja przegrałem. One wygrały. Terminator chichocze.**

Co rozumiem przez taką walkę?

⇨ *W jednej pracy moim pierwszym zadaniem było zainstalowanie na serwerze Team Foundation Services, czyli oprogramowania pomagającego programistom w pracy. Na zainstalowanie jednego programu przewidziano w harmonogramie… trzy dni! I wiesz co? Ledwo się wyrobiłem. Choć zwykle „robię dużo, robię szybko, robię dobrze".*

Niedziałające narzędzia. Biblioteki pełne błędów. Powolne (tak bardzo megaszybkie, ale jednak niewystarczająco!) komputery. Durne problemy wymagające wielogodzinnych sesji naprawczych. Konieczność stąpania na paluszkach między wszystkimi naczyniami połączonymi, nazwanymi środowiskiem developerskim, bo delikatne zruszenie dowolnego elementu powoduje „pad" całego systemu…

Pamiętasz wgrywanie gier z kaset na Atari? Kiedy to na widok małego wykrzyknika w rogu telewizora cały pokój cichł z obawy przed koniecznością wgrywania wszystkiego od nowa? Witaj w naszym świecie!

⇨ *Mam przyjemność znać programistę, który w wieku sześćdziesięciu lat wrócił do naszego fachu po… dwudziestu latach przerwy! Jego perspektywa jest bezcenna: potrafi porównać problemy, z jakimi borykamy się teraz, w drugiej dekadzie XXI wieku, z problemami lat dziewięćdziesiątych. Wtedy problemy rozwiązywało się zaaplikowaniem odpowiedniej wiedzy technicznej. Teraz ogłaszamy sukces, gdy „się kompiluje", a projekt „odpala się" na developerskiej maszynie. Straszne!*

Ooo, tak! To zdecydowanie była przyczyna mojego odejścia od programowania na co dzień. Aż mi ciśnienie skoczyło na samo wspomnienie!

Zdrowie! Nikt się nie dowie...

Skulony mizerny człowieczek, wgapiony w ekran przez szesnaście godzin dziennie? Bez przerwy, bo „bug na produkcji" albo „taki fajny nowy lib wyszedł"? Tu krzesło za pięć tysi ani biurko z elektrycznie regulowaną wysokością nie pomogą.

Do tego dieta składająca się z pizzy, kawy i browara. Okazjonalne burger, bo więcej warzyw, więc pewnie zdrowszy...

Znam wielu programistów, którzy efekty takiego trybu życia musieli odczuć na własnej skórze, by zrozumieć: mój organizm nie jest niezniszczalny. Zresztą, byłem jednym z nich. Opamiętałem się, gdy schylenie w celu zawiązania sznurówek było wyczynem prawie niemożliwym do wykonania, a idąc spać nad ranem, zostawiłem biurko zachlapane krwią tryskającą z nosa z wyczerpania (*sic!*).

> **...zdrowie, ile cię trzeba cenić, ten tylko się dowie, kto cię stracił...**

Odpowiedzialność

Oprogramowanie jest wszędzie. Awaria naszego kodu może pozostać niezauważona latami, a może spowodować katastrofę.

Żeby daleko nie szukać: pamiętasz wybory samorządowe w 2014 roku? W sporym uproszczeniu: amatorsko napisany system do głosowania paraliżuje państwo (co niekoniecznie było winą programisty, o czym zresztą wówczas pisałem: http://zawodprogramista.pl/wybory-2014).

Rakieta warta miliony dolarów spada na ziemię z powodu błędów sterownika.

Samochód staje się podatny na atak, pozwalając na zablokowanie przedniego koła za pomocą telefonu komórkowego.

Wielka firma w ciągu minuty traci dziesiątki tysięcy skrzynek mailowych podczas wdrożenia nowej wersji systemu do korporacyjnej komunikacji.

Drukarka przestaje drukować na czarno po automatycznej aktualizacji oprogramowania.

To wszystko wydarzyło się naprawdę! Przykłady można mnożyć. A przecież ta nowa era dopiero się rozpoczyna.

Dopisz własny czarny scenariusz i podpisz się pod nim. Nieprzyjemne uczucie. Błędy się zdarzają. To normalne. Tylko że nasze błędy mogą powodować ogromne szkody.

Nie żeby większość z nas poświęciła choć sekundę na takie refleksje. Ale inaczej patrzy się na swój kod, jeśli od niego może zależeć ludzkie życie.

Uzależnienie

Kodowanie może przerodzić się z hobby w tragiczny nałóg. Bez granic, bez zahamowań. Bez umiaru. *Life-work balance*? A gdzie tam; to hasło dobre na okładkę pism dla gospodyń domowych. „Jestem programistą. Oddycham, bo muszę. A poza tym to tylko programuję".

To jest prawdziwa pułapka. Byłem tam ja, było tam wielu moich znajomych. W internecie codziennie spotykam kolejne osoby tkwiące w tych sidłach i szczycące się tym.

⇨ *W jednej z firm zapytałem kolegę spędzającego w biurze dwanaście godzin dziennie: „dlaczego tyle siedzisz w pracy?". Odpowiedział mi: „a co ja będę robił w domu?". Bardzo smutne i bardzo prawdziwe.*

Eksploatacja

Z jednej strony mamy rynek pracownika. Z drugiej: nie zawsze i nie wszędzie.

Tak jak wielu programistów nieczysto wykorzystuje swoją pozycję na rynku pracy, tak wiele firm wyciska ze swoich „klepaczy kodu" wszystko, co się da. Chore nadgodziny, powtarzalne zadania, brak urlopów, wyzwania godne małpy, ciśnienie czasu...

Bezosobowy zasób. Numerek stukający w klawiaturę tak długo, aż zadziała. Wymienny trybik, kręcący się od rana do nocy – albo nara! A nie wszędzie jedne zamknięte drzwi oznaczają automatyczne otwarcie się dziesięciorga kolejnych.

Serio. To nie jest straszenie. Tak naprawdę wygląda rzeczywistość wielu z nas.

⏎

Mity o zawodzie programisty

Niezależnie od tego, czy jesteś już w IT, czy dopiero się do nas wybierasz (a może ani jedno, ani drugie), na pewno dotarły do Ciebie pewne słuchy. Opinie i prawidłowości o nas – programistach. Kilka jest szczególnie godnych odczarowania. Każdy z tych tematów będzie w późniejszych częściach książki rozwinięty. Tutaj jedynie – pozwolę sobie na frywolną parabolę – liźniemy czubek góry lodowej.

Kasa

Zaczynamy od oczywistego. Wizja wypełnionego skarbca jest czasami czynnikiem przyciągającym nowe osoby do zawodu, a potem następuje, nomen omen, zawód właśnie.

To prawda, że programista może zarobić dużo. Nawet bardzo dużo. Ale… prawie każdy może zarobić dużo! Legendarne już piętnaście tysięcy programistycznej pensji jest możliwe, lecz zdarza się sporadycznie. Patrząc na całe środowisko IT i stosując „aproksymację na oko", prawie nikt tyle nie zarabia.

Żeby tyle zarobić, trzeba być niesamowicie dobrym. Tutaj nie płaci się za nic. Trzeba mieć wiele lat wartościowego, różnorodnego doświadczenia. Rozwijać się przez długi czas i czasami godzić się na nudne projekty, których po prostu nikt nie chce tykać. A niejednokrotnie także… mieszkać w odpowiednim mieście, znać odpowiednich ludzi. No i mieć nieco szczęścia.

Zatem: czy programista MOŻE zarabiać bardzo dużo? Tak! Czy większość tak zarabia? Nie! Osobiście znam programistów zarabiających ponad dwadzieścia tysięcy. I znam zarabiających minimalną krajową.

Łatwo o pracę

Firmy bardzo starają się przyciągać do siebie programistów. Czasami ich starania ocierają się o granice absurdu, a nawet ją przekraczają. Ale… co zrobisz, skoro „taki mamy rynek"?

W końcu w samej Polsce w 2017 roku brakuje pięćdziesięciu tysięcy programistów! KAŻDY znajdzie zatrudnienie w takiej sytuacji, prawda?

Niestety, nieprawda. Wszystkie te raporty i głośne „klikalne" tytuły pomijają jeden ważny szczegół: faktycznie na rynku brakuje ludzi. Ale:

> **Brakuje 50 tysięcy specjalistów, a nie 50 tysięcy obojętnie kogo.**

Potrzeba doświadczonych ekspertów i ekspertek, mających za sobą wiele nierównych bojów.

W internecie znajdziesz wiele historii opowiadających o sukcesie. I bardzo dobrze, każdy potrzebuje inspiracji. Jednak należy zauważyć, że może być różnie. Za każdym sukcesem kryje się ogrom pracy i na każdy sukces może przypadać wiele porażek. O nich jednak mówi się rzadko.

Junior (albo kandydat na juniora) może miesiącami szukać pracy. Nawet w Warszawie, nie wspominając o mniejszych miastach. Dlaczego? Bo w ostatnich latach na rynek weszło bardzo wielu juniorów! Do absolwentów i absolwentek uczelni dołączyła masa samouków i kandydatów po różnych kursach i szkołach programowania. To bardzo dobrze, w końcu

> **doświadczeni seniorzy wyrosną właśnie z osób nowych,**

a nie na drzewach. Oznacza to jednak, że na starcie wcale nie musi być różowo. I coraz częściej nie jest.

Natomiast senior… niby nie ma tematu, prawda? Rozchwytywany, nie będzie wiedział, którą ofertę wybrać z setek możliwych! I to prawda. Ale co to za oferty?

Większość seniorów nie chce pracy byle jakiej. Apetyt rośnie w miarę jedzenia. Chciałoby się i dobrze zarabiać, i rozwijać, i rozwiązywać ambitne problemy, i mieć realny wpływ na tworzone projekty. A do tego jeszcze przychodzić do

biura w elastycznych godzinach i pracować w fajnym zespole. Spełnienie takiej kombinacji wymagań już nie jest banalne. Szczególnie, jeśli definitywnie odrzucamy możliwość relokacji.

Czy na rynku jest dobra praca? Tak! Czy zawsze, dla każdego, od zaraz? Nie!

To łatwe!

Znasz takie historie? No pewnie!

Była przedszkolanką, teraz programuje!

Z budowy przed monitor!

Księgowa, front-end developer!

Są z nimi dwa problemy...

Problem pierwszy. Często przedstawiają programowanie jako jedyny słuszny kierunek rekomendowany osobom niezadowolonym ze swojej dotychczasowej kariery.

Problem drugi. Takie opowieści zbyt mocno opływają lukrem i miodem. Szybko, prosto i wspaniale; raz-dwa i jestem programistą! Nawet jeśli wspomina się tam o trudach, to „uśredniony komunikat" można zinterpretować jako: rób to, już, teraz, uda ci się!

Ja sam trzymam kciuki za każdego adepta i każdą adeptkę sztuki programistycznej. Kibicuję im bardziej niż drużynie narodowej na mundialu. Ale trzeba zrozumieć, że to nie jest łatwe. Zdobycie nawet najbardziej podstawowej wiedzy wiąże się z ogromnym wysiłkiem.

Nauka do poziomu przydatności na rynku pracy to jeszcze trudniejsze zadanie. Niejednokrotnie trzeba zdobywać doświadczenie na własną rękę. Nie bez powodu istnieją wymagające studia programistyczne i tysiące książek na ten temat!

Przyswojenie samodzielnie choćby niewielkiej części tego materiału to nie lada wyczyn.

Czy programowanie jest proste? Tak! Uczysz się i umiesz! Nic w tym skomplikowanego.

Ale czy jest łatwe? Ooo, to już co innego. Nie, to nie jest łatwe.

A... potem? Co, gdy już jesteś w zawodzie? To często ciężka orka. Niejeden projekt to po prostu żmudne klepanie w klawiaturę. I trzeba zagryźć zęby i to przetrzymać. A przez to wszystko, wypalenie zawodowe to realny problem w naszej branży.

I nie jest to problem jedyny. Sportowcy zmagają się z kontuzjami kolan. Programiści: nadgarstków i pleców. Nie ma zmiłuj; klepiąc kod, można zrobić sobie krzywdę.

Ja osobiście doświadczyłem wszystkich negatywnych skutków tej pracy. I fizycznych i... tych drugich. Tu nie zawsze świeci słońce.

Dla inspiracji, ale niepozbawionej kontaktu z rzeczywistością, polecam Ci serię moich wywiadów w podcaście DevTalk. To rozmowy kierowane do osób wchodzących do branży IT albo zastanawiających się nad tym ruchem. Znajdziesz je pod adresem: http://zawodprogramista.pl/devtalk-junior.

NERD ALERT!

IT Crowd, *Sillicon Valley*, *Big Bang Theory*, a nawet *Mr. Robot*... Popularne seriale pokazują, jak jest.

Osoba techniczna to dziwak w pomiętym T-shircie albo koszuli w kratę. Do tego skarpety, sandały, delikatny asperger, miłość do pociągów, strach przed kobietami. Wybuchowy mix. Zamykasz stado takich dzikusów w piwnicy, zalewasz kawą, dorzucasz pizzę, delikatnie mieszasz i wychodzi system informatyczny.

Hmm...

Tak, można spotkać takich programistów. Ale można też spotkać takich nauczycieli WF-u.

Kilkanaście lat temu takie historyjki faktycznie często znajdowały odbicie w rzeczywistości. Jednak ta profesja tak bardzo ewoluuje, że nie da się już wpakować wszystkich nas do jednego wora.

Obowiązki programistów i programistek szybko ulegają zmianie i rozszerzeniu. Teraz wymaga się od nas nie tylko stukania w klawisze. Dostajemy inne

bardzo odpowiedzialne zadania. Analiza wymagań, spotkania z klientem, szacowanie i prototypowanie... Programista często jest jednocześnie dodatkowo biznesmenem i doradcą.

I bardzo dobrze! Bo to niesamowicie interesujące zajęcie.

⇨ *Mam znajomego programistę, który występował jako model w kampanii reklamowej, użyczając swojego wizerunku na billboardach, plakatach i w prasie. Programista!*

Nie jesteśmy jednakowi.

Czy można spotkać stereotypowego programistę? Tak!

Czy znajdziesz go na każdym kroku? Nie!

Za starzy na... programowanie

Z obowiązku i w tym miejscu wspomnimy fałszywe przeświadczenie, że dobry programista musiał zacząć programować w wieku dziesięciu lat. A najlepiej jeśli kodował już na ZX Spectrum, Atari, Commodore i dopiero potem przesiadł się na peceta. Przepisywał kod z pisma „Bajtek" i przenosił go do piwnicy kolegi na taśmie magnetofonowej.

Pewnie fajnie mieć takie historie do opowiedzenia. Ja ich niestety nie mam. Co prawda z ZX Spectrum faktycznie romansowałem już w wieku trzech lat, ale do programowania dojrzałem dwadzieścia lat później.

To nie jest sport wyczynowy. Jedyna granica to ta w Twojej głowie. I te w Twoim życiu, wyszczególnione już w pierwszym rozdziale.

Czy programowanie od podstawówki pomaga? Tak, poszerza perspektywę.

Czy jest konieczne? Zdecydowanie nie.

Czy programista musi być pasjonatem?

Uwaga, uwaga! Oto ON! Wybraniec! Zbawca! Czysty, płonący, zapamiętały. W ognistym rydwanie, z seledynowymi, zero-jedynkowymi lejcami.

Programista – pasjonat. Poświęcony programowaniu. Prawdziwie.

Prawdziwy pasjonat

W internetach wizerunek programisty idealnego rysuje się bardzo wyraźnie. Celem jego życia jest programowanie. Tak długo jak to robi, jest spełniony. To jego główne zainteresowanie, jego hobby, pasja. Jest w tym oczywiście bardzo dobry i swoją działkę zna na wylot. Jednocześnie stara się nieustannie poszerzać horyzonty i czerpać z konkurencyjnych rozwiązań.

Nie dla niego praca od ósmej do szesnastej. Tak robią tylko lenie! Słabeusze! Przecież do poznania jest ciągle tyle nowych rzeczy. Taki ktoś to w pewnych okolicznościach skarb. Dobrze jest mieć go czy to w zespole, czy to jako wykonawcę projektu.

Jednak czasami wolę mieć z nimi jak najmniej do czynienia. Wiesz dlaczego? Jeśli ktoś uważa się za takiego właśnie programistę, to w bardzo wielu przypadkach na tym nie poprzestaje.

To, co robi, jest wyjątkowe, może nawet godne podziwu, ale jemu to nie wystarcza. Jeśli on jest taki, to tego samego wymaga od wszystkich innych. Gardzi pseudoprogramistami nadającymi wysoki priorytet czemuś tak banalnemu jak czas dla rodziny, wakacje bez komputera czy wreszcie… wysokość pensji. Najchętniej wyrwałby flaki licealiście pytającemu: „czy opłaca się pójść na informatykę?".

Wszyscy inni, kwestionujący świętość jego postawy i świętość czynności kodowania oraz jej „najważniejszość", to materialistyczne plugawce. A to przecież Wielkie Powołanie! Na pohybel im!

Przyznam się z własnej nieprzymuszonej woli: doskonale taką postawę znam. Sam taki byłem z jakiegoś powodu.

Cukier w cukrze

Ilu mamy pasjonatów? Niekoniecznie aż tak ekstremalnych, ale jednak pasjonatów? To zależy, jak na to spojrzeć.

Kiedyś myślałem, że prawdziwi programistyczni pasjonaci to norma. Że po prostu tak wygląda ta branża. To były czasy, gdy nowych ludzi poznawałem zwykle na konferencjach i meetupach. Czyli tam, gdzie bywają… pasjonaci właśnie!

Prawdziwy obraz polskiego IT ujrzałem, gdy zacząłem regularnie odwiedzać „normalne" firmy. Zobaczyłem tam, że prawdziwy zapaleniec to właściwie rzadkość. I mało tego – zrozumiałem wartość postawy prezentowanej przez tych „normalnych", rzemieślników! Takich, którzy pracują z kodem. Dostają zadania, wykonują je, a potem idą do domu. Trzeba się czegoś nauczyć? To się nauczą. Nie trzeba? To poczytają kryminał, pobawią się z dziećmi albo pójdą do kina.

I tyle. Robią swoje. Jak dziewięćdziesiąt pięć procent populacji.

Programowanie to przecież po prostu profesja. Praca. I nie ma niczego złego w tym, że „tylko" pracą pozostanie.

Żałuję, że tak późno otworzyły mi się oczy. Mam nadzieję, że między innymi dzięki tej książce Twoje otworzą się, zanim będzie za późno.

⇨ *Poznałem drugie oblicze pasji o godzinie trzeciej trzydzieści w nocy. Mój dev-maraton trwał w najlepsze. Drugi czy trzeci miesiąc programowania po szesnaście, osiemnaście godzin dziennie. Bo tak trzeba. Samo się nie zrobi. Pasja! Power! Kawa, energetyk, fajka, znowu kawa – i jazda! Nie jestem przecież jakiś mięczak. Mugol.*

Nagle: kap. Kap. Kap… Co to? A to wesoło na spację chlupocze krew. Prosto z mojego nosa, bez ostrzeżenia. Jak zmaltretowany musiał być organizm, by w taki sposób błagać mnie o litość? Cudem nie wylądowałem wtedy w szpitalu.

Bez umiaru, bez przystanku, bez chwili refleksji. Latami żyłem tylko i wyłącznie kodem. Jak rozmowy, to tylko o kodowaniu. Jak lektury, to tylko techniczne.

Do kina chodziłem i tam drzemałem. Oglądając serial, wyczekiwałem końca, by wskoczyć z powrotem przed klawiaturę.

Przecież wszystko inne to marnowanie czasu!

A jednak. Skończyło się. Po ponad dekadzie – nie mogłem patrzeć na kod. Potrzebowałem kilkumiesięcznej przerwy, by w ogóle być w stanie myśleć o programowaniu. Straszne uczucie, uwierz mi.

Owszem, daj się ponieść. Owszem, poszukuj pasji. Owszem, celebruj odkrywanie tego wspaniałego świata. W końcu:

> **Passion is something that likes to be conquered.**
> ***Julia Marcell***

Ale... zachowaj odrobinę umiaru. Nie zmarnuj tego.

Wnioski?

Najlepiej robić coś z pasją i mieć z tego dużą kasę.

Gorzej robić coś z pasją i nie mieć kasy.

Jeszcze gorzej robić coś bez pasji i mieć z tego kasę.

A najgorzej robić bez pasji i nie mieć kasy.

Czy lepiej być pasjonatem? Oczywiście, że tak! Jakie większe szczęście może spotkać człowieka niż realizowanie się w swoim hobby i możliwość ustawienia sobie dzięki temu życia? Na pewno stosunkowo niewielka część ludzi może się tym cieszyć. A programowanie jest zawodem, który może takie coś dać. Umiarkowany pasjonat to pozycja idealna. Świadome trzymanie dzikich zapędów w ryzach nie jest proste, ale to najlepsza droga.

Niepasjonat? Tak zwany rzemieślnik? Też jest potrzebny! Dopóki robota jest zrobiona i świat się kręci dalej, każdy wybiera swoją pasję. Można być programistą, a prawdziwe emocje zachować na coś innego.

Przede wszystkim – jako pasjonat czy pasjonatka nie wymagaj takiego samego podejścia od wszystkich wokół. Żyj i daj żyć innym. Programistyczny dżihad to zło.

I na koniec, ku refleksji:

Czy nie można być pasjonatem przez jedną trzecią doby?

Czy pasja faktycznie musi oznaczać harowanie po godzinach?

⏎

Czy to zawsze będzie rynek pracownika?

Książka ta powstaje w dziwnych czasach. Dziś zapotrzebowanie na programistycznych rzemieślników jest tak wielkie, że prawie każdy prędzej czy później znajdzie pracę. Nawet ci słabi.

Czy tak będzie zawsze?

Zanim jednoznacznie odpowiemy na to pytanie, zerknijmy nieco wstecz.

Mam znajomych, którzy siedzą w branży o dekadę dłużej niż ja. Pod koniec lat dziewięćdziesiątych pracodawcy wcale ich sobie nie wyrywali. Bezrobotny programista nie był zjawiskiem dziwnym. Trzeba się było postarać, naszukać, naczekać, aby znaleźć pracę w zawodzie. Nawet w dużych miastach.

Dziesięć lat później ja sam zdecydowałem się na przeprowadzkę z mojego kochanego Białegostoku do Warszawy. Dlaczego? Za pracą! Firm jak na lekarstwo, a ambitnych wyzwań praktycznie brak.

Teraz wygląda to zupełnie inaczej i nawet w tak stosunkowo niewielkim mieście jak Białystok zapotrzebowanie na programistów jest ogromne. Jednak czy mądra byłaby wiara, że taki trend się utrzyma na zawsze?

Oj, nie. Świat się zmienia w ekspresowym tempie. Programowanie teraz wygląda nieco inaczej niż nawet dekadę temu. Ba, niż pięć lat temu!

I nawet nie chodzi o tak popularny dylemat, jak to, czy programistów zastąpią maszyny. Czy sztuczna inteligencja będzie sama pisała kod? To źle zadane pytanie.

A jeśli kod w ogóle nie będzie potrzebny?

Jednak nie wybiegajmy w przyszłość tak daleko.

Zamiast wróżyć z fusów, możemy przyjrzeć się faktom. Dwadzieścia lat temu nawet doświadczony programista mógł długo szukać pracy, a wcale nie dostawał w zamian kokosów. Dziesięć lat temu również praca nie wszędzie sama poszukiwała programisty, o czym miałem okazję przekonać się na własnej skórze.

W niedawnej przeszłości mieliśmy do czynienia ze zjawiskiem absolutnie wyjątkowym, z kilkoma szalonymi latami. Bez ograniczeń, bez umiaru, bez limitów. Ale dziś? Da się zauważyć, że rynek zaczyna się nasycać. Juniorami. Wejście do branży IT? Zostanie programistą? Proszę bardzo, ale trzeba zapłacić frycowe. Bo ten wielki boom jakby przygasa. Zaczepienie się na stażu albo w pierwszej pracy to czasami wyczyn. I w tym pomogą kolejne rozdziały.

Programista sprawdzony w bojach na pewno znajdzie zajęcie w dającej się przewidzieć przyszłości. Jeśli ma cenne i wartościowe doświadczenie, dostosuje się do zmiennej rzeczywistości, zdobędzie nowe umiejętności. Nie zardzewieje, nie osiądzie na laurach. Ale wykonawca najprostszych zadań, niewymagających specjalizacji, dopiero szukający swoich pierwszych szlifów? Co zrobi, jeśli tych zadań w ogóle nie trzeba będzie wykonywać? Chętnych do ich wykonywania już teraz jest często więcej niż wolnych miejsc.

Więc… czy zawsze będzie tak „dobrze" jak dzisiaj – w drugiej dekadzie XXI wieku? Nie! Naiwnością byłoby wierzyć inaczej.

I musisz się na to przygotować. Wszyscy musimy.

Prawdziwa odpowiedzialność świadomego programisty

Jako programiści dostajemy szansę budowania świata. Nie zawsze w pasjonujący sposób, ale jednak tworzymy coś z niczego. Coś, co zmienia życie milionów.

Ja sam budowałem rzeczy bzdurne, ale fajne i śmieszne. Innym razem: po prostu niepotrzebne. Jeszcze kiedy indziej: nudne, ale konieczne do działania świata. A niektórzy szczęśliwie tworzą soft ratujący życie. Dla każdego coś miłego.

Wszystkie te grupy zadań mają jedną cechę wspólną: wymagają klepania w klawiaturę. I nader często to właśnie do tego sprowadza się praca programisty. A czy... nie stać nas na więcej?

Programista siedzi w swojej piwnicy i czeka, aż spadnie na niego *task*. Dłubie, testuje, kompiluje, wdraża. I siedzi dalej. W oderwaniu od rzeczywistości.

Powoli zaczyna się to zmieniać. Z biegiem lat zaczyna się od nas wymagać czegoś więcej. I bardzo dobrze!

Świadomy programista sam chce czegoś więcej!

Potrzebuje wpływu na finalny produkt. Cieszy się zrozumieniem rozwiązywanego problemu. Dochodzi do przyczyn takiej a nie innej specyfikacji.

Dlaczego? Bo kto jak nie programista zna system od podszewki? To programista jest ekspertem w zakresie technologii i możliwości ujarzmienia mocy drzemiących we wszechobecnych maszynach. To świadomy programista, rozumiejąc sedno problemu, potrafi zaproponować optymalne rozwiązanie.

Niezależnie od tego, czy piszę grę, CRM, dialer, portal, czy „inteligentny słoik": moją pracą nie jest walenie w klawisze. Walić w klawisze może każdy. Walić w klawisze może małpa. Moją pracą, moją odpowiedzialnością, jest myślenie. *Problem solver*, a nie... *keystroker*.

O ile przyjemniej wchodzi się do biura, gdy na swoje zajęcie spojrzy się z takiej perspektywy!

Owszem, nie wszędzie takie podejście się sprawdzi. Niektóre środowiska widzą w programistach jedynie bezmyślne konieczne maszynki zmieniające kawę w kod. I, przyznajmy to otwarcie: latami zaniedbań zasłużyliśmy sobie na takie postrzeganie. Co w takiej sytuacji? Pokażmy, że może być inaczej. Kluczem jest zrozumienie, że taka percepcja jest uzasadniona. Że naprawdę sami chcieliśmy być odizolowani od biznesu, wymagań, analityki i... klientów. Więc się od nas odczepiono.

> **Zmusiliśmy nawet przedstawicieli biznesu**
> **do porozumiewania się w naszym języku!**

„Skasuj usera", „zrób *update* wiersza" – i tak dalej. Wygraliśmy. I czas na odwrót, czas to zmienić.

Tak naprawdę kodowanie nie jest ani najbardziej skomplikowanym, ani najbardziej wymagającym elementem pracy programisty. Tego można się nauczyć, i to w dość krótkim czasie.

Komunikacja, głębokie zrozumienie problemu, szeroka wiedza – to stanowi eksperta. Domaganie się takiej pozycji wymaga odwagi i przeciwstawienia się *status quo*. Pokaż, że interesuje Cię problem! Wykaż się inicjatywą! Szukaj znaczenia, ważności w tym, jak spędzasz ogromną część swojego życia. Zakwestionuj bzdurne pomysły i zaproponuj lepsze!

Bądźmy częścią całego procesu, a nie wyłącznie podwykonawcami cudzych idei. Gdzieniegdzie już tak jest. Gdzie indziej jest to nierealne. Ale między tymi dwoma miejscami świadomy programista bierze sprawy we własne ręce. I działa.

Zadaniem chirurga nie jest pokroić. Misją chirurga jest naprawić. Zadaniem nauczyciela nie jest przeczytać na głos podręcznik. On ma nauczyć.

Zadaniem programisty nie jest kodować. Misją programisty jest...

Dokończ to zdanie na http://zawodprogramista.pl/misja!

Programistyczna kasa i uzależnienie

Dopiero zaczynamy, a już pojawia się po raz trzeci czy czwarty. Magiczny temat rzeka. No ale cóż, taka rzeczywistość...

Raczej nie jest Ci obce pojęcie „programista 15k", prawda? Programista zarabiający piętnaście tysięcy miesięcznie. Nie jest dookreślone, czy netto czy brutto, w jakiej walucie i za co. Ale wiadomo, że to poziom, na którym programista zaczyna być szanowany... w kręgach wyznających kult tej magicznej piętnastki.

Zarabiasz piętnaście tysięcy? Gratuluję! Nie zarabiasz? To nic! Marzysz o tym? Są lepsze obiekty westchnień.

Myślałem o pieniądzach bardzo wiele i mam sporo spostrzeżeń do przekazania. Podczas tej lektury chociaż kilka razy wydaj z siebie „hmm" albo podrap się po głowie, dokonując pewnych refleksji całkowicie od nowa, a będę wielce zadowolony.

Jak to jest z tymi 15k?

Od wielu lat jestem bardzo aktywny i udzielam się w ciekawych inicjatywach, niektóre nawet sam organizuję. Prowadzi to do postrzegania mnie jako jednego z tych bardzo dobrze zarabiających. Ktoś kiedyś nawet powiedział o mnie, że „on to wypłatę pewnie taczkami do banku wozi".

I wiesz co? Tak jest. Sam uważam, że zarabiam i zarabiałem bardzo dobrze. Ale niezmiernie istotne są oczekiwania i perspektywa. Bo przyznam Ci się do czegoś:

> **Nigdy nie zarabiałem tych magicznych 15 tysięcy miesięcznie jako programista.**

Wstyd!

Owszem, zdarzały się miesiące znacznie przekraczające tę kwotę, ale nigdy jako etatowy developer. Jedni powiedzą, że jestem za głupi. Inni, że nie osiągnąłem swojego potencjału.

Mało który programista jest w stanie tyle zarobić. Pogódź się z tym, bo często nie ma nawet o co walczyć. Pieniądze to nie jedyna miara sukcesu programisty. Zawsze bardziej ceniłem sobie swobodę, rozwój i ogólną satysfakcję z życia nad absolutny, bezkontekstowy stan konta.

Pisałem już o tym, że w drugiej dekadzie XXI wieku żaden programista nie zarabia w przybliżeniu piętnastu tysięcy miesięcznie. Ta grupa jest tak niewielka, że można ją pominąć w rozważaniach. A ci osiągający wyższe wyniki finansowe rzadko kiedy są „po prostu programistami" i mają wiele innych obowiązków.

Dodatkowo wysokość pensji jest zależna od wielu różnych czynników i tylko jeden z nich to faktyczne umiejętności. Wcale też nie jest powiedziane, że tak duża pensja wynagradza to, z czym muszą się mierzyć w swojej codziennej pracy. Naprawdę. Ja świadomie zarabiałem poniżej swoich możliwości, żeby móc skupić się na poszukiwaniu… Czego?

Szczęście?

Już dawno temu pewien mądry człowiek powiedział:

> **Money doesn't make you happy. I now have $50 million but I was just as happy when I had $48 million.**
> ***Arnold Schwarzenegger***

Powyżej pewnego poziomu, na którym mamy zapewnione podstawowe życiowe potrzeby, tak to po prostu działa. Dodatkowe dziesięć procent podwyżki okupione jakimkolwiek dyskomfortem może nie być tego warte.

Zanim wpadniesz w dziką pogoń za kasą, zadaj sobie pytanie: dlaczego? Nie „po co?", ale właśnie „dlaczego?". Nie pytaj o cel, lecz o przyczynę. Dlaczego bierzesz udział w tym wyścigu? Z jakiego powodu sprzedajesz kolejne godziny

swojego życia za premię? Albo zostawiasz ukochany rodzinny kąt i przeprowadzasz się do większego miasta w poszukiwaniu eldorado?

Autentycznie Ci brakuje? Czy może po prostu kolega ma więcej i wreszcie mu pokażesz? Albo na Wykopie napisali, że jak nie masz piętnastu tysi, to jesteś leszcz?

Wiem, że presja w niektórych środowiskach jest ogromna. Ilość kasy wpadającej co miesiąc na konto liczy się bardziej niż to, kim naprawdę jesteś. Możesz albo się temu poddać, albo... zmienić środowisko?

Jeśli szukasz „szczęścia", to w tym dzikim tańcu go nie znajdziesz. W ten sposób nie da się wygrać. Podniesiony poziom życia szybko Ci spowszednieje.

> **Nigdy nie będzie dość, bo zawsze można więcej.**

To błędne koło, w które wpada zbyt wielu młodych programistów i programistek. Nie wszystko złoto, co się świeci.

Koszt

W imię czego i jakim kosztem?

Lanos i Lambo jadą tak samo szybko, jeśli ich jedyna trasa to dom–biuro w warszawskim korku. Willą za kilka baniek się nie nacieszysz, spędzając w niej pięć godzin dziennie, z czego cztery poświęcając na niespokojny sen. Pyszne whisky kosztujące tysiaka za butelkę nie będzie smakowało tak dobrze, jeśli jego jedyną rolą będzie szpan albo ukojenie nerwów.

Nawet jeśli ciągle świat wpaja Ci, że potrzebujesz więcej i więcej, to rozejrzyj się i wykonaj analizę. Widzisz, co można mieć, co można dostać. Ale powierzchowne spojrzenie nie odpowie na pytanie: co trzeba za to sprzedać?

Nie potępiam wysokich zarobków, bynajmniej! Sam dążę do ich maksymalizacji, bo lepiej mieć niż nie mieć. Nie twierdzę też, że poświęcenie się temu jednemu celowi musi się źle skończyć! Chciałbym jedynie, by takie poświęcenie odbywało się... świadomie.

Nie ma nic za darmo.

Sufit

Choć ten rozdział kieruję do każdej osoby związanej z branżą, tutaj szczególnie chciałbym przemówić do „młodych wilków". Szczęściarzy, którzy prawie od razu wskoczyli na naprawdę poważny pułap. Nie ma ich wielu, ale kilka takich watah się znajdzie.

Pamiętaj, że zawsze u góry jest jakiś sufit i że nie będzie się on podnosił w nieskończoność. Często startując z wysokiego pułapu, po prostu szybciej go dotkniesz. Czasami nawet z impetem.

To nie jest tak, że jeśli jako absolwent czy absolwentka zarabiasz dychę, to za pięć lat będziesz zarabiać stówę. Może się zdarzyć, że po takim czasie będziesz zarabiać tyle samo. Albo i mniej!

Widełki na jakimkolwiek stanowisku nie są z gumy. Im bardziej zbliżasz się do górnej granicy, tym mniejsze masz pole manewru. A co po osiągnięciu maksimum, i to prawdopodobnie niemałym kosztem?

Niewola

Patrząc z boku, może się wydawać, że „młody krezus" złapał Pana Boga za nogi. Każdy by tak chciał! Ale niestety, bardzo często wraz z kolejnymi podwyżkami mocno rosną też codzienne koszty. Ten proces to

inflacja stylu życia – uwaga!

Poziom minimum, konieczny do osiągnięcia stanu zadowolenia, jest nieustannie podnoszony. Fajna fura, mieszkanie, drogie ciuchy, restauracje... W tym nie ma niczego złego. Lecz co będzie, jeśli tego zabraknie? Wydaje się, że nie może zabraknąć, prawda? Ale jednak. Życie nie będzie już takie fajne.

To nie jest podróż w jedną stronę. Jak się dochrapiesz do celu i nieodpowiedzialnie rozdysponujesz efekt swoich starań, znajdziesz się w potrzasku.

W niewoli.

Nie zmienisz pracy na fajniejszą, a projektu na bardziej rozwijający. Bo nie możesz obniżyć pensji.

Nie postawisz się szefowi. Bo nie zaryzykujesz zwolnienia.

Nie przeprowadzisz się w spokojniejsze okolice. Bo nie dorównasz tam swoim dzisiejszym zarobkom.

Nie rzucisz wszystkiego w cholerę, próbując zrealizować własny świetny pomysł na biznes. Bo masz tak dużo! Ale to wszystko już jest z góry wydane!

W pewnym momencie

przestajesz mieć kasę, a kasa zaczyna mieć Ciebie.

Wiem, jak to brzmi, ale naprawdę – chodzi tylko o świadomość. Odpowiedz sobie na pytanie: czy dzięki wysokim zarobkom Twój poziom niezależności zwiększy się czy zmniejszy? Wbrew pozorom to nie jest pytanie absurdalne!

Można sprowadzić to do symulacji: co się stanie, jeśli z dnia na dzień stracisz pracę? To możliwy scenariusz; firmy bankrutują, oddziały są przenoszone między miastami. Nie można brać żadnej jednej, konkretnej pracy za pewnik. I co wtedy? Ile ofert pracy będzie dla Ciebie z góry skreślonych, bo za mało płacą?

Pragmatyzm

Oczywiste jest, że lepiej zarabiać więcej niż mniej. Ale warto każdy sukces na tym polu mądrze wykorzystać. Nie zakładać, że zawsze będzie tylko lepiej. To bardzo naiwne i nieodpowiedzialne.

Nigdy nie będę namawiał, aby żyć z wizją czarnego scenariusza przed oczami i ciągle wynajdywać potencjalne katastrofy, przeżywając je we własnej wyobraźni. To psuje całą radość z pięknej codzienności. Ale przyklejone na stałe różowe okulary są po prostu niebezpieczne.

Sam zawsze staram się oceniać, czy potencjalna nagroda pieniężna – nawet jeśli pokaźna – na pewno jest warta mojego poświęcenia. Nieraz okazuje się, że nie! I zdarzało mi się rezygnować z przedsięwzięć, które w krótkim czasie przyniosłyby nawet po kilkanaście tysięcy złotych. Staram się jednak, aby pieniądze nigdy nie były celem samym w sobie.

Jestem czasami pytany, skąd ciągle czerpię energię do działania. Gdzie znajduje się źródełko mojej nieskończonej motywacji? Otóż odpowiedź jest zadziwiająco prosta: zwykle staram się robić to, na co aktualnie mam ochotę. Ma to ogromną zaletę: prawie nigdy nie jestem do niczego zmuszany. Ale jest też pewien koszt: nieraz życie według tej zasady oznacza, że zarabiam mniej, niżbym mógł.

Jednak prawdziwa wartość życia nie kręci się tylko wokół stanu konta.

Zapomniał wół, jak cielęciem był

Nie lubię moralizowania, ale... czasem trzeba. Przez moment, przez chwilę.

Jeżeli czytając ten rozdział, cieszysz się, to być może nie jest on kierowany do Ciebie. Ale zapamiętaj go, wróć tutaj za pięć czy dziesięć lat. Wtedy może się okazać potrzebny. Naprawdę.

Pokolenia

Sporo słyszy się o „jakości kolejnych pokoleń" polskiego IT: że to lenie, idioci, gimbaza, że nie ogarniają, są roszczeniowi, bezczelni, agresywni.

Wierzyłem. Zakładałem, że to musi być prawda. I trwało to latami. Jednak wystąpiły dwie okoliczności.

Po pierwsze: zastanowiłem się, czy inwektywy rzucane pod adresem „młodych" są charakterystyczne tylko dla nich? Gdzie następuje ta magiczna granica? Jaki jest wiek – czy też rok urodzenia – po którym człowiek staje się „tępą gimbazą"?

Okazało się, że nie ma takiej granicy. Słabi kandydaci i kandydatki zdarzają się niezależnie od tego, czy szukamy seniora, czy juniora. Bucowate indywidua znajdziemy zarówno na stażu, jak i w „wieży z kości słoniowej", legowisku

architektów. Po prostu odpychający ludzie znajdą się wszędzie i wiek nie ma tu nic do rzeczy.

Po drugie natomiast: zdecydowałem się poznać tych strasznych młodziaków. Przekonać się na własnej skórze, z czym mamy do czynienia, a nie tylko wierzyć innym na słowo.

W dniu pisania tych słów jestem aktywnym użytkownikiem Snapchata od półtora roku. Pewnie jednym ze starszych. Przez ten czas nawiązałem kontakt z piętnasto- i dwudziestolatkami, z gimnazjalistami, licealistkami, studentami i stażystkami. I co się okazało? Uwaga: to nie są debile.

> **To ambitni, młodzi ludzie. Tacy, jakim sam byłem.**

Niezależnie od tego, jak byśmy „kroili" gatunek ludzki, w każdej grupie znajdzie się „lepsza" i „gorsza" część. Do każdego można się przyczepić.

Skleroza

Ludzie siedzący w zawodzie od dziesięciu czy dwudziestu lat zapomnieli, jak to jest. Nie potrafią postawić się w sytuacji dzisiejszych adeptów sztuki programowania. To naturalne: zdobywając jakąś umiejętność, posiadając kawałek wiedzy, tracimy wspomnienia sprzed tego momentu. Zjawisko to ma nawet swoją nazwę:

> **curse of knowledge, czyli klątwa wiedzy.**

Niekoniecznie trzeba z tym walczyć, ale warto to sobie uświadomić.

Tak, ci młodzi, ci nowi, nie pamiętają, jak to było iść do biblioteki i uczyć się z prawdziwej papierowej książki. Mają wszystko podane w internecie. I co z tego? Czy to źle? Mnie czasami wypominano, że nie pamiętam czołgów jeżdżących po ulicach polskich miast. Czy to źle? Nie! Bardzo dobrze, że kolejne pokolenia mają łatwiej, a nie trudniej. To się nazywa postęp. Rozwój. Ewolucja?

Po drodze, jako „seniorzy", gubimy pewną perspektywę. Kilkanaście czy kilkadziesiąt lat temu do nauki, do tego, by zostać dobrym programistą czy dobrą programistką, trzeba było posiąść o wiele mniej wiedzy. Próg wejścia – jeśli

chodzi o zakres materiału – był zdecydowanie niższy. Teraz nawet samo bycie na bieżąco z technologiami to nie lada wyzwanie, a co dopiero wejście w ten świat od kompletnego zera! Mimo rozwoju internetu i ogromu wiedzy dostępnej „na zawołanie" – teraz jest trudniej.

Paradoks? Właśnie zaprzeczyłem sam sobie? Jak może być i łatwiej, i trudniej zarazem? Widocznie może. Bo jest po prostu inaczej!

Nie obawiam się o przyszłość. Będzie miał kto przejąć pałeczkę. Część z tych osób okaże się rewelacyjna, a część słaba. Tak samo dzieje się w moim pokoleniu i tak działo się w pokoleniach wcześniejszych.

Zamiast się wywyższać, pomagajmy. Zamiast szydzić, edukujmy. Zamiast podawać sygnet do ucałowania, bądźmy przykładem, a wszystko się ułoży.

> **Szacunek należy się temu, kto na niego zasługuje,**
> **a nie temu, kto się go domaga.**

Można też niekiedy zaobserwować dziwną obawę, że ci młodzi mieliby zabrać starszym pracę. I tutaj niestety zaserwuję szach-mat: jeśli osoba świeża w branży zagraża Twojej pozycji, to co się z Tobą działo przez ostatnie lata? To tylko i wyłącznie Twoja wina! I naprawdę: wstyd się do tego przyznać.

2 Początki – dla młodszych i starszych

Jak zostać programistą? Wiele dróg do celu

Klamka zapadła: chcesz do IT! To zastanówmy się: jak się tu dostać? Jakie są furtki i możliwe drogi?

Studia

Studia informatyczne to świetny sposób na wejście do branży. Sposób, z którego ja skorzystałem. Sprawdzona, standardowa droga. Niestety, bardzo czasochłonna, bo zajmuje kawał życia. Jednak mimo konieczności spędzenia długich lat na przygotowaniu do zawodu programisty – i średniego poziomu tego przygotowania – zdecydowanie polecam wybranie tej ścieżki. To pierwsza i najsensowniejsza opcja. Oczywiście jeśli życie pozwala Ci na taki wybór.

A dlaczego? Bo nigdzie indziej i nigdy więcej nie dostaniesz tak łatwej do wykorzystania szansy rozwoju w wielu kierunkach. Eksperymentowania, poznawania informatycznego świata i... popełniania błędów bez konsekwencji. To możliwość spróbowania chleba z dziesiątków pieców, nawiązania znajomości na lata, wrośnięcia w programistyczną społeczność i wybicia się w niej.

Trzeba natomiast uważać, by ten czas odpowiednio wykorzystać! To niezmiernie istotne. Tym tematem zajmiemy się w kolejnym rozdziale.

Nauka programowania na uczelniach

Większość studentów i studentek ma nierealne oczekiwania względem uczelni i prowadzących zajęcia: „nauczcie mnie programować!". Są one jak najbardziej uzasadnione, lecz zwykle niemożliwe do spełnienia w dzisiejszej rzeczywistości. Im szybciej to zrozumiesz, tym lepiej.

> **Programowania nie nauczą Cię na studiach.**
> **Nauczyć się go – to Twój obowiązek.**

Na studiach studiujesz: uczysz się Ty, a nie uczą Ciebie. Jeśli o tym zapomnisz, lata nauki będą skutkować jedynie frustracją, a nikt Ci ich przecież nie zwróci!

Dlaczego studia nie nauczą programowania? Powód jest prosty: większość programistów woli... programować, niż uczyć. Poza tym typowy programista nie może sobie przyjść, ot tak, na uczelnię i zacząć wykładać. Skoro zatem programiści nie uczą, lecz programują, to

> **kto uczy programowania na uczelniach?**
> **Często nieprogramiści!**

Po drugiej stronie katedry spotkasz pracowników uczelni. Nie ma niczego złego w byciu pracownikiem uczelni, ale akurat w naszym zawodzie najważniejsza jest praktyka. A jak podzieli się z Tobą swoim doświadczeniem praktycznym ktoś, kto tego doświadczenia – komercyjnego – nie ma? Niestety – nie podzieli się. I tu jest pies pogrzebany.

Raczej nie poznasz historii z życia. Nie skubniesz prawdziwego bojowego doświadczenia od osoby, która zjadła zęby na prawdziwych, dużych projektach. Prawdopodobnie będą Ci wtłaczać do głowy definicje: co to jest klasa, co to jest metoda, co to jest procedura? I każą podpisywać te dziwne kształty na znienawidzonych diagramach UML.

Nie jest to niczyja wina: ani programistów, ani pracowników naukowych. Po prostu dopóki nauczaniem przedmiotów programistycznych nie będą zajmować się osoby z praktyką programistyczną, a najlepiej także z kwalifikacjami trenerskimi, tak będzie to wyglądało.

Co prawda to się zaczyna zmieniać. Coraz częściej uczelnie nawiązują współpracę z firmami, pozwalając „prawdziwym programistom" na prowadzenie praktycznych przedmiotów. To bardzo dobry kierunek rozwoju i mam nadzieję, że za kilka lat stanie się normą na wszystkich uczelniach. Póki co jest to jeszcze rzadkość.

Jeśli nie studia, to co?

Jest pewien problem: studia nie są rozwiązaniem dla każdego. To kilka długich lat codziennej pracy z wizją nagrody w przyszłości. Jest to satysfakcjonująca droga, ale nie wszyscy mogą sobie na nią pozwolić. Pogodzenie dziennych studiów z inną pracą i rodziną to często wyzwanie ponad siły i możliwości.

Jak w takim razie mogą poradzić sobie ci „starzy", dla których na studia jest po prostu za późno? Czy po przekroczeniu trzydziestki drzwi do IT bezpowrotnie się zamykają?

Oczywiście, że nie!

Każdego roku tysiące osób udowadniają, że można z powodzeniem zmienić branżę i przejść na stanowisko programisty z zupełnie innego. Z kilkoma takimi osobami nagrałem obszerne wywiady. Znajdziesz je pod adresem: http://zawodprogramista.pl/devtalk-junior.

Jaka jest zatem alternatywa dla studiów?

Studiowanie, ale samodzielne

Nauka na wyższej uczelni nie jest konieczna, by studiować. W końcu słowo to ma jeszcze inną definicję:

> **studiować – gruntownie poznawać, badać coś.**
> ***Słownik języka polskiego (sjp.pwn.pl)***

Co to oznacza? Że wystarczy zakasać rękawy, odpalić internet i po prostu: nauczyć się programowania we własnym zakresie!

Skąd? Listę przykładowych stron z kursami do nauki programowania – w języku polskim i angielskim, zarówno płatnych, jak i darmowych – znajdziesz na: http://zawodprogramista.pl/kursy.

Mamy w branży bardzo dużo samouków. W wielu przypadkach są to wspaniali specjaliści z najwyższej półki. Jak to możliwe? Tak bez uczelni? Oczywiście!

Pamiętaj: długie godziny nauki we własnym zakresie są konieczne, niezależnie od obranej drogi. Nie da się zlecić komuś zadania: „naucz mnie kodować"

i z założonymi rękami czekać na efekty. W każdej sytuacji, absolutnie zawsze, trzeba się uczyć samodzielnie. Uczyć bardzo dużo. Realizować projekty, wyciągać naukę z błędów i nieustannie eksplorować ten obszerny programistyczny świat.

Tego etapu nie da się pominąć.

Już wiemy, że teoretycznie każdy może zostać programistą. Jest to kwestia determinacji, zaangażowania, przygotowania na hektolitry potu i łez. A przede wszystkim zrozumienia: to nie jest łatwa droga.

W tym scenariuszu dochodzi jeszcze jeden, dość istotny, problem: efekt samouwielbienia. Wśród wielu samouków szczególnie można zaobserwować – przynajmniej na początku drogi – zachwyt nad własną wspaniałością. I fakt, to może być uzasadnione. Bo dokonują czegoś wspaniałego, godnego pochwały i oklasków. Ale… nie do przesady!

Nie można zignorować etapu weryfikacji swojej wiedzy, konsultacji z ekspertami i poszerzania horyzontów.

> **Nie skupiaj się na tym, co już wiesz.**
> **Skup się na tym, czego jeszcze nie wiesz.**

Warto też zrozumieć, że praca w pojedynkę nie przyniesie wystarczających efektów w czasie kilku tygodni. Skok wiedzy i umiejętności „od zera do niezera" będzie oczywiście przerażająco ogromny już od samego początku, lecz to rzadko wystarczy do osiągnięcia najważniejszego programistycznego celu: otrzymania pracy w zawodzie programisty.

Na studiach potrzeba do tego lat. W przypadku samodzielnej nauki może uda się osiągnąć ten efekt szybciej, ale na tej drodze czyha wiele pułapek. Takie podejście da więcej elastyczności niż program na uczelni, jednak warto traktować to jako zabawę. Tak, żeby ciągle mieć frajdę i iść do przodu. Bardzo łatwo bowiem jest się zagrzebać w nieistotnych szczegółach, wybrać złą ścieżkę i zabrnąć w ślepy zaułek. Zginąć pod natłokiem nowych terminów, pojęć, zadań i całych oceanów wiedzy, tylko czekającej na pochłonięcie.

> **Ciągła nauka może stać się wymówką usprawiedliwiającą ucieczkę od działania!**

Co zatem może pomóc? Znalezienie sobie mentora, mentorki, guru. Nauczyciela lub nauczycielki! Gdzie? A proszę bardzo...

Szkoły programowania (bootcampy)

Można cały proces nieco uprościć. Tak, aby zminimalizować ilość czasu potrzebnego na naukę, a zmaksymalizować jej efekty.

Jest dla Ciebie za późno na studia trwające kilka lat? W szkole programowania będziesz się uczyć bardzo ciężko, ale owocnie i produktywnie. Większość dostępnych kursów trwa kilka miesięcy albo nawet kilka tygodni. Dlaczego tak krótko? Bo skupiają się one dokładnie na tych umiejętnościach, które są wymagane dzisiaj na stanowisku junior developera. Tak, by przygotować Cię do poszukiwania pierwszej pracy. Już, teraz.

Gdyby z całego programu studiów wycisnąć esencję – wszystko, co jest absolutnie niezbędne do startu w IT – to również byłyby to tygodnie, a nie lata. Praktyka, praktyka, praktyka.

Lubisz samodzielną naukę (to i tak jest zawsze wymagane w zawodzie programisty), ale – z różnych względów, na przykład braku czasu albo natłoku informacji – wolisz podążyć wyznaczoną, sprawdzoną ścieżką, zamiast w pojedynkę odkrywać tę właściwą drogę? Nauczyciel/mentor dokładnie wie, którędy iść. I Cię poprowadzi.

Żebyśmy się dobrze zrozumieli: pracę nadal wykonujesz Ty, bez tego ani rusz! Ale dokładnie wiesz, jaką pracę wykonać. I po co.

Oczywiście jest i druga strona medalu, a jakże! Taka przyjemność może kosztować nawet kilkanaście tysięcy złotych. Jednak moim zdaniem... to dobrze! Dlaczego? Ponieważ ten fakt podkreśla, że to nie jest droga dla każdego. Ta droga to ostateczność, będąca jednocześnie wiążącą deklaracją: ja naprawdę tego chcę! Dzięki niej docenisz to, co otrzymasz. To inwestycja. W siebie.

Rzeczy tanich, przecenionych, otrzymanych w gratisie się nie szanuje.

No i oczywiście tak krótki czas przygotowań jest spowodowany maksymalnym okrojeniem materiału. Skupieniem się tylko i wyłącznie na tym, co ułatwi jeden konkretny cel: jak najszybsze znalezienie pierwszej pracy jako junior developer. Aż tyle, ale i tylko tyle. Po ukończonym kursie nadal będą Cię czekały całe lata dalszej nauki!

Dużo zależy od osób w Twoim otoczeniu. Sporadycznie może się też trafić się słaby mentor – ale wtedy nie miej litości. Skargi, reklamacje, zwroty: trzeba walczyć o swoje!

Zwróć też uwagę, że nawet po ukończeniu kursu może wcale nie być różowo ze znalezieniem pracy. Naprawdę. O tym już pisałem wcześniej. Jeśli zależy Ci głównie na kasie, to spoko, nic mi do tego. Ale możesz się srogo rozczarować! Zorientuj się, ile w Twojej okolicy zarabia młodszy programista i policz, ile czasu będzie Ci się zwracał taki kurs… Oczywiście zakładając, że taką pracę zdobędziesz.

Niezależnie od wszystkiego…

Praca programisty jest trudna. Droga do tej pracy – wyboista i pokręcona. Im więcej masz zobowiązań, im mniej czasu do dyspozycji, tym będzie Ci ciężej.

Przygotuj się na walkę.

Jednak na pocieszenie: jaka jest satysfakcja z osiągnięcia łatwego celu? Niewielka. Czasem warto powalczyć, pomęczyć się. Zmagać się z przeciwnościami losu i… ze sobą. To ma być trudne!

Mamy w polskiej branży IT wiele przykładów osób, które dopięły swego. Po ciężkiej harówie, pokonaniu wielu przeszkód, sprostaniu wyzwaniom – pracują jako programiści i programistki. Jest to dla nich powód do dużego zadowolenia.

Jeszcze raz zapraszam Cię do wysłuchania kilku wywiadów dostępnych pod adresem: http://zawodprogramista.pl/devtalk-junior.

Natomiast rekomendacje książek pomocnych w pogłębianiu programistycznej wiedzy znajdziesz na stronie: http://zawodprogramista.pl/ksiazki.

⏎

Jak nie zmarnować lat na studiach?

Wracamy do podstaw. Do „standardowego" sposobu zostawania programistą. Na studia.

Ustaliliśmy już, że uczelnia nie nauczy Cię programować. Więc po co tam chodzić? Wbrew pozorom powodów jest sporo!

Dyplom

Co daje Ci uczelnia na koniec, tak namacalnie? Daje Ci dyplom. A do czego jest Ci on potrzebny? Do tego, by dostać pracę w firmie wymagającej wyższego wykształcenia.

W ofertach pracy dość regularnie można znaleźć taki zapis. Jednak często to nie jest prawdziwe wymaganie, absolutnie konieczne do dostania pracy! Absolutna konieczność występuje tam, gdzie jest to uregulowane prawnie. Więc... w zdecydowanej mniejszości firm.

Zatem po co zawracać sobie głowę dyplomem? Jedyna realna korzyść z dyplomu – na dzień dzisiejszy – jest taka, że dwa lata po zakończeniu studiów zwiększy się przysługująca Ci liczba dni urlopowych. Dobre i to! Chociaż...

Żyjemy w dziwnych czasach!

Teraz większość firm puszcza do Ciebie uwodzicielskie oczko i jest w stanie tłuc się z innymi w kisielu tylko po to, żeby Cię do siebie zwabić. I na dyplom tak realnie mało kto patrzy. Ale to nie będzie trwało wiecznie! Sytuacja na rynku pracy może się zmienić. I się zmieni.

Ta bańka pęknie.

Dzisiaj żyjemy na programistycznym Dzikim Zachodzie.

Nie ma zasad, nie ma reguł. Każdy robi, co chce. Taki stan nie będzie trwał wiecznie.

I co wtedy? Wtedy – nie do wiary! – nawet papier może zacząć się liczyć. Wyśmiewany dyplom może być wymagany do podjęcia pracy na stanowisku programisty. I te czasy mogą nadejść szybciej, niż się ktokolwiek spodziewa.

Zdecydowanie lepiej dyplom mieć, niż go nie mieć. Niekoniecznie na dzisiaj, ale na przyszłość.

Wiedza

Na wspomnianym dyplomie masz wypisane przedmioty. Logika, statystyka, elektronika, analiza matematyczna, fizyka, algorytmy, struktury danych, asembler, Turbo Pascal... Hej! Czy przyda Ci się to w pracy zawodowej?

Prawdopodobnie: nie.

Ale wiesz co? Szczerze życzę Ci, żeby ta wiedza się jednak przydała. Żeby z czasem okoliczności zmusiły Cię do odkurzenia podręczników z tych tematów. To może być błogosławieństwo.

Teraz może Ci się wydawać, że do końca życia będziesz klepać stronki. *Born to code*, do grobowej deski. I że zawsze to będzie takie fajne, jak jest teraz.

Uwierz mi: nie będzie.

To, co robisz teraz, prawdopodobnie Ci się znudzi.

Mało tego: niektóre z wykonywanych dzisiaj obowiązków mogą po prostu zniknąć z rynku.

W takiej sytuacji dobrze mieć w głowie chociaż plączący się zbiór pojęć pozwalających na łatwiejsze doedukowanie się później.

Nie chodzi o uczenie się wszystkiego na zapas (bo może się kiedyś przyda), ale o rozeznanie, poślizganie się po powierzchni, poznanie, „z czym co się je". We własnym zakresie tego nie zrobisz, bo aby eksplorować zakamarki, najpierw trzeba wiedzieć o ich istnieniu.

Studia dają Ci tę możliwość. Pozwalają zatrzymać się na chwilę i zastanowić:

Jak wielu rzeczy jeszcze nie umiem!

Te „zbędne" przedmioty zapewniają cenną perspektywę. Uczą tak niesamowicie potrzebnej pokory.

Nigdy nie napisałem ani jednej linijki produkcyjnego kodu w asemblerze, ale poznałem go na studiach. Do CV go nie wpiszę, ale mogę o nim porozmawiać. Bo wiem chociaż mniej więcej, o co chodzi.

⇨ *Mój znajomy pracował nad systemem, w którym instrukcje do wykonania przez aplikację były zaszyte w XML-u. Inne zespoły implementujące odczyt tych instrukcji traktowały XML jako tekst, a wyłuskiwanie wykonywalnych instrukcji odbywało się przez cięcie i krojenie tego tekstu za pomocą operacji na znakach. Bardzo, bardzo szybko kod przestał być ładny i utrzymywalny. Wspomniany znajomy napisał do tego własny kompilator! Bo pisał wcześniej jeden na studiach i zauważył, że to dla niego dobre zastosowanie.*

Też zresztą pisałem na studiach własny kompilator. To był mój ulubiony przedmiot z całych pięciu lat. Dzięki temu wiem, jak to wszystko działa pod spodem. Owszem, bez tego również umiałbym generować kolejne strony kompilowalnego tekstu, ale byłbym mniej świadomym programistą!

Taką wiedzę mieć warto. Bez studiów jej nie zdobędziesz. Nie dlatego, że w Ciebie nie wierzę! Po prostu nie znajdziesz na to chęci i czasu.

Na studiach dowiesz się o tematach, których nie nauczysz się na własną rękę, bo ich dziś nie potrzebujesz. A raczej: nawet nie wiesz, że ich potrzebujesz. Bo nie wiesz, że są.

Poczytałem konstrukcje języka Ada. Napisałem coś w Lispie. Zobaczyłem OCaml. Pokazano mi Fortrana i COBOLa. Nie mogę powiedzieć, że je znam. Nigdy ich nie użyłem i najprawdopodobniej nigdy nie użyję, ale je widziałem. Mogę

o nich porozmawiać na bardzo podstawowym poziomie – na takim samym jak osoby, z którymi być może będę mieć do czynienia.

To bardzo ważne: wiedzieć, o czym mówią. I również potrafić wypowiedzieć się na wiele tematów. Tak się wchodzi w środowisko.

▫⇨ *Pewnego razu wziąłem po godzinach poboczny projekcik. Jedna z procedur bardzo wolno działała i moim zadaniem była jej optymalizacja. Przez pierwsze kilka wieczorów kombinowałem na czuja. Zmieniałem to, co mi się wydawało wolne.*

W końcu podszedłem metodycznie: przypomniałem sobie, że był taki przedmiot jak ASD: „algorytmy i struktury danych", a na nim jakieś drzewa, czy coś. Można wykorzystać do optymalizacji.

W ciągu kilkugodzinnego researchu przypomniałem sobie wszystko. Dzięki zastosowaniu red-black tree*, o którego istnieniu poprzedni autor programu nie miał pewnie pojęcia, czas wykonania zszedł z pół godziny do kilku sekund.*

Wtedy pomyślałem sobie: „aaaaa, to ja po to chodziłem na studia!".

Aktywności

Zabawa to piękna sprawa, co nie? Wszystko jest dla ludzi. Dwa pierwsze lata studiów koncertowo zmarnowałem, pławiąc się w tanich szampanach. Na szczęście po drugim roku naszły mnie bardzo bogate w skutki refleksje i się opamiętałem.

Pójście na uczelnię i poprzestanie na zaliczaniu przedmiotów do niczego nie prowadzi. Nawet jeśli z jakichś dziwnych powodów zdobędziesz same piątki, to jest to bardzo, bardzo zła droga.

Nie potrzebujesz piątek. Z większości przedmiotów wystarczą Ci trójе.

A dlaczego? Dlatego, że dzięki temu zostanie Ci więcej czasu na Rzecz Najważniejszą. Czyli naukę programowania.

Jak już wiemy, studia nauczą Cię wielu rzeczy. Poza kodowaniem. W tym zakresie działasz w pojedynkę. Musisz mieć czas na naukę programowania, bo musisz choć trochę umieć kodować. Z tym nie da się dyskutować.

Ale to nie wszystko. Studia otwierają Ci bardzo wiele drzwi. To tam dowiesz się o ciekawych inicjatywach: studenckich zlotach, konkursach, konferencjach. Tanie szampany warto wymienić również na to. Zgłaszaj się wszędzie, jedź i poznawaj ludzi! To jest czas, w którym możesz przygotować sobie naprawdę wygodne i interesujące dalsze życie.

Wkręcaj się w środowisko. Rób wiele, jak najwięcej. Nie bój się profesjonalnych spotkań dla ludzi już pracujących. IT to otoczenie bardzo otwarte i pomocne. Zapytasz, poprosisz – odpowiemy, pomożemy. Dzięki zaangażowaniu możesz pchnąć swoje doświadczenie o całe lata świetlne naprzód. Wykorzystaj tę szansę.

W IT różnica wieku nie ma znaczenia.

To do tego są studia.

Moja rzeczywistość na piątym roku to studiowanie, praca w zawodzie programisty na część etatu, start w konkursie Imagine Cup, udział w dużym informatycznym projekcie skupiającym kilkudziesięciu studentów z całej Polski, no i pisanie magisterki. Oczywiście ani snu, ani imprez zbyt wiele wówczas nie doświadczyłem.

Było bardzo ciężko, ale ten jeden rok zaprocentował na bardzo, bardzo długi czas. Procentuje nadal. Zarówno umiejętnościami technicznymi, jak i nauczkami wyciągniętymi z popełnionych wówczas błędów. I oczywiście znajomościami.

Czwarty rok z kolei spędziłem w Danii na wymianie studenckiej (Socrates/Erasmus). Również bardzo polecam. Kiedy wyprowadzić się na rok do innego kraju, jeśli nie na studiach właśnie?

Tak naprawdę wszystko sprowadza się do uzmysłowienia sobie jednego faktu: w studiach nie chodzi o naukę. Ale nie chodzi też o imprezy. Świadome przejście przez studia da Ci więcej doświadczenia, znajomości i perspektyw niż

kolejne pięć lat spędzone na kodowaniu. Nie bój się; programowanie nie ucieknie. Pieniądze – też nie. Natomiast to wszystko, co mogą dać Ci studia, ucieknie jak najbardziej. Z rodziną i kredytem raczej nie zrobisz tych rzeczy. Zapomnij.

> **Of all the words of mice and men, the saddest are, „It might have been".**
> ***Kurt Vonnegut***

Na studiach możesz pracować na pół etatu, na luzie. Jak najbardziej realny jest zatem pakiet *all inclusive*. Oczywiście nie będzie łatwo, ale nie ma tak być!

Jaka uczelnia?

To spory dylemat. Ale nie ma nad czym się specjalnie rozpisywać: idź tam, dokąd chcesz. Tutaj raczej każdy wybór jest dobry. Bo popatrz:

Jeśli trafisz na „dobrą" uczelnię, to się pewnie więcej nauczysz. I poznasz „śmietankę" – wzajemnie się motywującą i napędzającą. Tysiące ludzi rocznie dają radę, więc i Ty dasz. Będzie lans.

Jeśli natomiast wybierzesz, tak jak ja, uczelnię teoretycznie „słabszą", to wszystkie obowiązki wypełnisz raz-dwa i zostanie Ci masa czasu na wszystko inne. Na rozwój we własnym zakresie.

Skończyłem Politechnikę Białostocką. Przez przynajmniej połowę studiów nie wymagała ona ode mnie prawie niczego. Bywały tygodnie, że nawet się tam nie pojawiałem. Skupiałem się na wszystkim innym – tym, co wówczas uznawałem za najważniejsze. Skupiałem się bardzo mocno i cieszę się, że nie wylądowałem na jakiejś lepszej uczelni. Kto wie, może teraz musiałbym, o zgrozo, pracować w NASA zamiast siedzieć przy kominku i pisać tę książkę?

Największa wada słabszej uczelni jest taka, że nie zmusza ona do wykorzystania czasu w dobry sposób. Niczego nie musisz. Wszystko zależy od Ciebie. O wiele łatwiej jest te lata zmarnować, przebimbać. Ale już wiesz, co należy zrobić, prawda?

Więc...?

Wiele osób poświęca czas na studia, ignorując szanse, jakie one dają. Marnuje pięć lat swojego życia i twierdzi, że studia są do bani.

Ale to ich wina. Bo nie wykorzystują odpowiednio tych lat.

Studia nie są niezbędne, ale bardzo się przydają. Ja po dziennej, pięcioletniej informatyce na PB radzę sobie nieźle. Nie żałuję. Co prawda, zdobyta tam wiedza nie przygotowała mnie do zawodu, ale dzięki uczelni wiedziałem przynajmniej, czego się uczyć.

Jedyne, czego trochę żałowałem, to niepodjęcie w większym zakresie pracy zarobkowej. Przez ponad połowę studiów robiłem bardzo dużo. Fora, projekty, konkursy, ciągle coś, non stop. Po ukończeniu uczelni byłem naprawdę dobry i całkiem doświadczony. A do tego – biedny jak mysz kościelna i nieco tym faktem sfrustrowany.

Pamiętam, że kupiłem wtedy wieżę za sześćset złotych, bo na lepszą nie było mnie stać. Grała jak słuchawki pchełki z kiosku Ruchu, takie za pięć złotych. Ależ mi było wówczas smutno! O samochodzie czy wkładzie własnym do potencjalnego przyszłego mieszkania nie mogło być mowy.

Z drugiej strony: wszystko, co wtedy robiłem, zwraca mi się do dziś, i to nie tylko finansowo. Z nawiązką.

Moja rekomendacja: warto iść na studia dzienne.

Zawsze można zrezygnować. Albo po trzech latach odebrać papierek i spasować.

Natomiast w drugą stronę nie jest tak łatwo. Im więcej lat na karku, tym ciężej odwrócić decyzję o pominięciu uczelni.

Czy programista musi znać matematykę?

Trzeba zdawać matematykę na maturze, by dostać się na informatykę. Może stąd bierze się przeświadczenie o konieczności znajomości matematyki w zawodzie programisty? Dodatkowo pierwsze lata takich studiów wypełnione są matematyką i fizyką po brzegi.

Więc... tak, programista musi znać matematykę. Bo jak inaczej?

Ble!

Przyznam Ci się do czegoś: na studiach miałem poprawkę z algebry. Całek nie rozumiałem aż do trzydziestego trzeciego roku życia. Prawie wszystkie matematyczne przedmioty były dla mnie czarną magią. Na wspomnienie równań różniczkowych nadal dostaję gęsiej skórki. Może to wina moja, może materiału, a może prowadzących (na pewno!), ale takie są fakty.

I co? I nic! Z powodzeniem zdobywałem kolejne szlify w zawodzie programisty. Sam siebie uważam za dobrego programistę, a opinie innych to potwierdzają.

Więc czy matematyka jest programiście zbędna? Nie, to też nie tak!

Kto potrzebuje matmy?

Matematyka jest potrzebna osobie, która wykonuje pracę wymagającą znajomości matematyki! Większość dzisiejszych zadań programistycznych nie należy jednak do tej grupy.

Tak, matematyka jest „królową nauk". Ale nie zawsze jest niezbędna. Mnie osobiście przez kilkanaście lat pracy nie przydała się ani razu. Kto więc jej potrzebuje?

Bez niej na pewno nie dostaniesz pracy jako programista gier. Nie zostaniesz też Mistrzynią Algorytmów ani wzbudzającym szacunek Lordem Optymalizacji. Na stronie http://zawodprogramista.pl/matematyka znajdziesz linki do materiałów traktujących o matematyce w programowaniu.

Matematyka nie jest konieczną bronią w arsenale programisty, ponieważ została już poskromiona i zaimplementowana przez naszych poprzedników! W bibliotekach i algorytmach kryje się matematyczna wiedza. My musimy ją tylko wykorzystać. Nie wymyślamy tego na nowo. Lata matematycznych doświadczeń sprowadzamy teraz do wywołania jednej metody. Używamy matematycznej potęgi bez dokładnego jej zrozumienia. I to jest w porządku.

W rozwoju i planowaniu ścieżki swojej kariery wszystko jest sztuką wyboru. Mój brak matematycznej edukacji to świadoma decyzja. Zamiast na matematykę, postawiłem na zagadnienia związane z programerką. Skupiłem się na poznawaniu świata programowania obiektowego, testów jednostkowych, wzorców projektowych czy mechanizmów kontroli wersji.

Nie da się umieć wszystkiego.

Nauka czegokolwiek wymaga czasu, jest dużą inwestycją. Moja inwestycja okazała się trafna: skupiłem się na tym, co mnie pociągało. Nie próbowałem na siłę stać się matematykiem. Zamiast tego zostałem pragmatycznym programistą.

Nie twierdzę oczywiście, że należy matematykę olać! Na pewno lepiej ją znać, niż jej nie znać. Po prostu nie jest ona niezbędna w naszej pracy, czego sam jestem dowodem. Niemniej jednak każdy, kto matematykę zna na poziomie ponadpodstawowym, ma mój wielki szacunek.

Najważniejszy język każdego programisty

Każdy adept i każda adeptka sztuki programowania staje przed pierwszym, najtrudniejszym dylematem. Uczyć się języka proceduralnego czy obiektowego? A może funkcyjnego? Niższego czy wyższego poziomu? Uczyć się takiego, z którym łatwo znaleźć pracę, czy takiego, który da dużo frajdy?

Spójrzmy prawdzie w oczy: jest tylko jeden język, którego naukę należy traktować jako absolutny priorytet. Jeden język niezastępowalny. Jeden język służący jako globalny środek komunikacji programistów na całym świecie.

To język angielski.

Prawie zawsze dokumentacja biblioteki jest napisana po angielsku. Najprężniej działające fora programistyczne – również. Poradniki, tutoriale, kursy: angielski. Książki, blogi, artykuły: angielski. Podcasty, konferencje, przykładowe aplikacje: angielski!

Wreszcie: klienci! Do kontaktu z nimi także często wymagany jest angielski.

Język programowania można zmienić w trakcie kariery, nawet wielokrotnie. Ba, każdy większy projekt wymaga znajomości więcej niż jednego języka programowania. Ich uczymy się na bieżąco, a gdy trzeba – zapominamy i robimy w głowie miejsce na nowy.

Ale angielski? Jest wymaganiem uniwersalnym.

Czy brak znajomości angielskiego to wielka tragedia? Absolutnie nie! To po prostu jeden z obszarów do nadrobienia. Dobra wiadomość jest taka, że liczy się głównie zdolność komunikacji. Niekoniecznie trzeba od razu zaprzątać sobie głowę kilkoma czasami, gramatyką i składnią. Gdy wejdziesz w ten świat, zobaczysz, że spora część środowiska zna angielski na poziomie „Kali lubić krowa jeść". I to wystarcza – na początek.

W Polsce w zawodzie programisty czy programistki niejednokrotnie wystarczy podstawowa umiejętność konsumowania informacji. Bycie bierną stroną, rozumiejącą komunikaty. Czytanie dokumentacji, specyfikacji, wymagań; przeglądanie forów, blogów; wyszukiwanie informacji po angielsku. Choć oczywiście warto pójść o krok dalej i nauczyć się więcej.

Jak uczyć się angielskiego?

Z nauką angielskiego jest jak z nauką programowania: mając określony cel, nie skupiamy się na przyswojeniu jak największej ilości materiału, na przerobieniu

setek podręczników i kursów. Celem jest poznanie angielskiego przydatnego w pracy programisty, a ten charakteryzuje się dwiema cechami. Po pierwsze, jest dość prosty; po drugie – zawiera wiele słów z żargonu IT, których musisz się nauczyć.

Cel ten można osiągnąć, podążając dwiema drogami. Najlepiej jest podążać nimi jednocześnie.

Biernie

Wbrew pozorom, nauka angielskiego nie musi być trudna; szczególnie, że bardzo dużo możemy nauczyć się bez poświęcania dodatkowego czasu! Naprawdę.

Zacznij od najprostszego. Nie będzie to wymagało od Ciebie żadnych dodatkowych aktywności w codziennym życiu: wyeliminuj polskie napisy z konsumowanych treści wideo. Prawie każdy raczy się co jakiś czas filmem lub serialem. Pełnię komfortu daje – poza popcornem – lektor lub polskie tłumaczenie tekstowe. Pozbądź się ich – najlepiej jednego i drugiego. Pierwszy obejrzany w ten sposób odcinek będzie wydawał się trudniejszy i mniej przyjemny. Ale z każdym kolejnym coraz bardziej się przyzwyczaisz – zaufaj mi. A jeżeli totalnie niczego nie rozumiesz, to włącz napisy, ale po angielsku. To bardzo pomaga, a nadal uczy.

Tę samą praktykę stosuj wszędzie, na przykład w grach komputerowych. Kupuj wersje angielskie. Na grach też można się dużo nauczyć!

Na początku może pojawić się frustracja: „ja niczego nie rozumiem!!!". Spokojnie, to normalne. Tak ma być. Nie masz prawa rozumieć, skoro nie umiesz. Trzeba się po prostu nauczyć!

Wypróbuj również praktykę nauki języka przez radio. Dowiedziałem się o niej od Piotra na Snapchacie. On w ten sposób uczył się greckiego! Ta metoda polega na puszczaniu sobie audycji radiowych po angielsku, w których rozmowy przeważają nad muzyką. Wiadomości, wywiady, felietony – cokolwiek. Nie skupiaj się na tym, rób swoje. Radio niech brzęczy w tle. Te treści będą się powoli przetwarzać w głowie same, podświadomie.

To na pewno nie jest najbardziej efektywny sposób nauki, ale nie zawsze o efektywność chodzi. Szczególnie w przypadku metody niewymagającej absolutnie żadnego dodatkowego czasu ani wysiłku. Piękne, prawda?

Oczywistym uzupełnieniem powyższych metod jest czytanie książek po angielsku. Przyznam, że sam nigdy nie przeczytałem ani jednej pozycji z literatury pięknej w języku innym niż polski, bo czas na taką lekturę traktuję jako sto procent relaksu.

Ale zamierzam to zmienić. Po kilku pierwszych książkach pochłoniętych w ten sposób podobno nie ma już różnicy. Trzeba się tylko przyzwyczaić.

A teraz czas na coś bardziej związanego z branżą.

Odwiedzaj i regularnie przeglądaj wybrane programistyczne blogi tworzone przez zagranicznych twórców. Zapisz się do anglojęzycznych technicznych newsletterów. Nie musi być tego dużo, ale ważne, aby codziennie podtrzymywać kontakt z językiem – także w jego technicznym aspekcie. Oswajać się z żargonem, poznawać nowe słówka; rozumieć specyficzne terminy.

Zobacz rekomendacje na: http://zawodprogramista.pl/blogi.

Nie do przecenienia są także nagrania z zagranicznych konferencji programistycznych. Oglądając wystąpienia ze światowych scen, uczysz się dwóch obszarów jednocześnie: angielskiego oraz programowania. Brawo Ty!

Na YouTube znajdziesz tysiące nagranych prezentacji – do wyboru, do koloru. Moje typy przedstawiam na: http://zawodprogramista.pl/konferencje.

I wreszcie: podcasty. Prywatne audycje radiowe niezależnych twórców; medium w Polsce coraz popularniejsze. Na rynku znajdziesz dużo wartościowych audycji, oferujących za darmo masę wiedzy.

W samochodzie, komunikacji miejskiej, na siłowni, na rowerze, podczas odkurzania... Wszędzie tam, gdzie masz wolne ucho, wykonując inną prostą czynność. Telefon, słuchawki – i gotowe! Podobnie jak w przypadku nagrań z konferencji masz okazję uczyć się angielskiego i programowania jednocześnie.

Wpadaj na http://zawodprogramista.pl/podcasty i wybieraj.

Aktywnie

Bierne konsumowanie treści to ta prostsza część nauki języka. Tak samo jak z programowaniem: łatwiej jest przeczytać książkę albo zaliczyć kurs, niż stworzyć coś swojego.

Czas więc na kolejny poziom. W tym przypadku ponownie mam dla Ciebie kilka propozycji.

Zawsze zachęcam osoby programujące – także początkujące! – do założenia bloga. Własny blog po angielsku to idealne miejsce na naukę języka.

„Ale ja nie chcę pisać bloga!" – wiem, wiem, znam to, słyszałem setki razy. Ale w tym przypadku to słaba wymówka, bo w procesie nauki języka istotne jest pisanie, a nie publikowanie treści. Możesz tworzyć artykuły choćby do szuflady. Uczyć się będziesz i tak, a jednocześnie wszystkie wymówki z kategorii „nie chcę bloga" pozostaną zaspokojone.

Pisanie to jedno. A co z mówieniem?

Jeśli nauczysz się mówić po angielsku, to automatycznie wskoczysz na kolejny poziom, niestety niedostępny dla większości polskich programistów. Dlaczego tak wiele osób zajmujących się programowaniem nie mówi po angielsku? Może się wstydzą. Może nie mają okazji poćwiczyć. Może praca tego od nich nie wymaga, więc zadowalają się minimum...

Mówienie dzieli się na dwa obszary: jednostronne przekazywanie informacji oraz rozmowę.

Jak ćwiczyć łatwiejszy aspekt: przekazywanie informacji? Nie zaskoczę Cię; tutaj trzeba po prostu mówić! O czymkolwiek. O swoim projekcie, swoim dniu, wyzwaniach i problemach. Im więcej mówisz, tym lepiej Ci to idzie. I – podobnie jak w przypadku „bloga do szuflady" – nikt nie musi Cię słuchać! Bardzo polecam Ci spróbowanie składania codziennych raportów na Snapchacie albo Instagramie w formie krótkiego znikającego wideo. Bierzesz telefon, wciskasz guzik i zaczynasz mówić. Wysyłasz w internet, gdzie – przynajmniej na początku – nikt oprócz Ciebie tego nie ogląda. I nie ma się kompletnie czego wstydzić, bo akurat tam Twoje próby znikają po dwudziestu czterech godzinach!

W środowisku firmowym można ciekawie rozwijać swój angielski zespołowo. Jak wiele dałoby zarezerwowanie nawet jednej godziny w tygodniu na krótką prezentację jednego członka zespołu na dowolny techniczny temat? Oczywiście po angielsku!

A co z najtrudniejszym sposobem wykorzystania języka, czyli rozmową? Dialogiem? W pracy polecam eksperyment, który nazwałem *English Friday*. W piątki cały zespół komunikuje się tylko i wyłącznie po angielsku, włączając w to przerwy na kawę czy wyjście na lunch. Wspaniała sprawa!

Nie każda firma wspiera jednak takie inicjatywy. Często wręcz nie ma na to żadnych szans. Wtedy trzeba na własną rękę poszukać partnera czy partnerki do nauki. Wystarczy nawet trzydzieści minut tygodniowo na telefonie, Skypie czy Hangoucie, by w krótkim czasie zaobserwować wyśmienite rezultaty. Ale gdzie znaleźć chętnych do rozmowy? Polecam poszukanie kogoś o podobnych potrzebach na mojej grupie na Facebooku. Dołączyło tam znaczne grono polskich devów; na pewno znajdzie się ktoś chętny na takie wzajemne kształcenie. Dołącz na http://zawodprogramista.pl/facebook.

Na tym kończymy rozważania o języku angielskim. Teraz piłeczka jest po Twojej stronie: do dzieła! Bez wymówek.

Nic nie rozumiem! Jak nauczyć się terminologii?

Świat IT ma dużo pojęć. Nawet po polsku nasz żargon czasami ciężko zrozumieć. Na początku wszystko to może wydać się przytłaczające. To nowa rzeczywistość, nowy język.

Ale nie ma powodu do strachu. Oswojenie się z tymi wszystkimi słowami jest stosunkowo proste. Jedyne, czego Ci brakuje, to odpowiedni kontekst, w którym

te terminy są używane z sensem. Tak jak przy nauce nowego języka – musisz się osłuchać. Zidentyfikować sytuacje charakterystyczne dla każdej z tych fraz.

Co najlepsze: spokojnie możesz to zrobić w samotności. Ba, nawet „w tle", nie spędzając nad tym dodatkowego czasu!

Nie rozumiesz co to „frontend", „backend", „serwer", „test", „framework", „bug"...? Słuchaj rozmów programistów! A gdzie ich podsłuchać? Podobnie jak w przypadku nauki angielskiego – polecam podcasty!

Podcasty mają zarówno swoich zwolenników, jak i przeciwników. Faktem jest, że dla Ciebie będą świetnym źródłem wiedzy programistycznej. Nawet jeżeli wyda Ci się, że niczego z nich nie rozumiesz, osłuchasz się z terminami, przyzwyczaisz do pewnych sformułowań.

A skąd wiem, że podcasty działają? Bo pisało do mnie wiele początkujących osób, słuchających mojego podcasta DevTalk. Był dla nich bramą do świata IT. Nawet jeżeli temat danego odcinka wydawał się zbyt skomplikowany i niezrozumiały, zostawało w głowie kilka nowych pojedynczych słów, kształtujących nowy język w głowie słuchacza czy słuchaczki.

Aktualną listę polecanych przeze mnie podcastów znajdziesz na stronie: http://zawodprogramista.pl/podcasty. Dużo dobrego!

Ale to nie jedyne źródło takich informacji. Oczywiście również miałem ten sam problem na początku swojej drogi, za moich czasów jednak nie było podcastów. Wpadłem na inny – choć bardziej angażujący – sposób: oglądałem nagrania występów z konferencji. Przez pierwsze dni większość z nich była dla mnie czarną magią, nawet mimo studiów! Jednak po miesiącu czy kwartale różnica w rozumieniu przeze mnie konsumowanych treści zaskoczyła mnie samego. Okazało się, że nieświadomie nauczyłem się bardzo wiele. Zacząłem rozumieć słowa, które wcześniej nic nie znaczyły. Coś zaskoczyło.

Aktualny spis wartych obejrzenia konferencji znajdziesz na stronie: http://zawodprogramista.pl/konferencje.

Nie łam się. Od tego momentu do chwili, w której poznasz cały nowy słownik, dzieli Cię raptem parę tygodni.

Warto zauważyć analogie między nauką nowego języka (angielskiego) a nowego żargonu po polsku. Ciekawe, prawda?

⏎

Jak uczyć się programowania?

Odpowiednie podejście do procesu nauki programowania może albo znacząco uprościć, albo znacząco utrudnić całą drogę. Przez lata dosłownie setki początkujących zadawało mi pytania pozwalające na obserwację standardowego sposobu rozpoczęcia edukacji. Niestety, ten sposób nie jest optymalny. Ba, jest nawet niewłaściwy.

Podzielę się z Tobą sposobem alternatywnym, gwarantującym o wiele lepsze efekty.

Sposób standardowy – błędny

Pierwsze pytanie, jakie prawie zawsze pada z ust adeptów i adeptek dev-sztuki, brzmi: „jakie materiały do nauki polecasz na początek?".

I nic dziwnego! Ucząc się języka, chwytamy za podręcznik. W szkole uczą nas historii, każąc wkuwać daty z książek. Nauka na kursie prawa jazdy zaczyna się od omówienia znaków drogowych. Na muzyce dostajemy opis nut i jakichś innych dziwnych szlaczków. Tak działa edukacja.

Naturalne wydaje się więc nakupienie książek o programowaniu. Obłożenie się dokumentacją do języków programowania, ściągnięcie gigabajtów programistycznych kursów wideo, zasubskrybowanie dziesiątków dev-blogów i katowanie mózgu wszelkimi *how to*, „dla opornych", „na start" – i tak dalej.

Oglądamy, słuchamy, czytamy. Chłoniemy, bez umiaru. Przecież dzięki temu osiągniemy poziom ekspercki, co nie? Obejrzę trzy kursy, przeczytam pięć książek i jestem programistą!

Niestety nie. Ta wiedza zacznie się po prostu… ulewać. Jedyne, czego się dorobisz, to poczucie zagubienia. I ewentualnie migrena.

Wcale nie dziwię się takiemu postępowaniu. Znam to doskonale. Sam przebyłem taką drogę – i dotarłem do ściany. Dlatego wiem, że takie podejście nie działa.

Co okaże się po pewnym czasie? Że doskonale znasz literaturę. Wiesz, jakie są tytuły, jacy są autorzy. Kanały na YouTube nie będą miały przed Tobą tajemnic. Otworzysz tamę i zaleje Cię fala informacji. Przydatnych, ciekawych, fascynujących – owszem. Ale na tym etapie, na samym początku, Twoim zadaniem nie jest przyjmowanie ton wiedzy. Twoim zadaniem jest nauka programowania!

> **Continuous learning is the new procrastination.**

Po całym tym trudzie zainstalujesz programistyczne narzędzia, z pasją siądziesz do rozwiązywania problemu. Klikniesz *new project*, a następnie zagapisz się na migający kursor. Bo nie wiesz, co dalej. Przecież nadal nie umiesz programować. Cały czas jesteś w tym samym miejscu. Na początku drogi. Na starcie.

Bierna nauka faktycznie sprawdzi się może w przypadku historii. Ale do programowania lepiej podejść inaczej.

Sposób lepszy: *just do it!*

To proste:

> **Jak nauczyć się programowania? Programując!**

Zapomnij o książkach technicznych. Zapomnij o kursach. Na samym początku prawdopodobnie nic Ci nie dadzą, tylko namieszają w głowie. A dlaczego? Bo nie zrozumiesz, co tam jest napisane!

Wymyśl sobie projekt do realizacji i go zrealizuj. Od samiuśkiego początku. Do końca. Czy będzie trudno? Oczywiście! To ma być trudne i jest trudne. Ale to będzie tak samo trudne – a może nawet trudniejsze! – z głową wypełnioną masą nieprzydatnej na tym etapie teorii.

Pamiętaj, że najwięcej nauczysz się na własnych błędach. Popełnisz ich dziesiątki, codziennie. Problemy napotkasz na każdym kroku. Na tym polega praca programisty: rozwiązujemy problemy!

W czym jednak takie podejście jest lepsze niż nauka „na zapas"? Otóż dzięki rzuceniu się od razu w wir praktyki, na głęboką wodę, będziesz rozwiązywać jeden problem po drugim, po kolei. Nie przeładujesz głowy rozwiązaniami tysięcy problemów, których jeszcze nie masz.

Sugeruję od samego początku, od pierwszego dnia, zapisywać swoje spostrzeżenia i dokumentować proces nauki. Popełniasz błąd? Znajdź rozwiązanie! Ale na tym nie koniec: zanotuj nie tylko rozwiązanie, ale również swoją pomyłkę. Dzięki temu w pełni zrozumiesz, co się stało. I uwierz mi: niejednokrotnie do tych notatek wrócisz. Zawsze rekomenduję, aby dodatkowo upubliczniać takie notatki – założyć programistycznego bloga opisującego Twoją drogę. Jeżeli to zrobisz już teraz, podziękujesz mi za jakiś czas. Jeśli jednak nie dasz się namówić na bloga, to chociaż notuj sobie to wszystko prywatnie. Koniecznie!

Skup się też na jednej technologii. Nie staraj się liznąć wszystkiego dookoła, bo Cię to po prostu przerośnie. Nie daj się również rozproszyć, wciągnąć w niekończący się łańcuch „ciekawych" pojęć. Całe dni możesz spędzić, przeglądając opisy kolejnych definicji i obszarów informatyki. Doprowadzi to do tego, że finalnie nie osiągniesz swojego celu, który prawdopodobnie był inny niż „przeczytaj pół internetu", prawda?

Napotykasz na swojej drodze nowe, ciekawe pojęcie? Coś, co Cię interesuje, pociąga, czego chcesz się nauczyć? Wspaniale! Dodaj to do swojej listy „rzeczy do nauki w przyszłości" i wróć do aktualnego zadania.

Książki, blogi, podcasty, kursy, konferencje... to wszystko, owszem, jest warte uwagi, ale tylko jako uzupełnienie procesu. Po to, żeby osłuchać się z terminami, otrzaskać się z definicjami, zdobyć nieco technicznej ogłady. Ale nie powinno to być głównym celem samym w sobie.

Nie masz czasu na „wszystko" i musisz z czegoś zrezygnować? Zdecydowanie zrezygnuj z biernego przyswajania informacji. Praktyka przede wszystkim!

Dlaczego ten sposób nauki działa?

Układanie planu nauki to trudne zadanie. Wymaga poznania bardzo rozległego obszaru wiedzy, spojrzenia z góry na cały materiał, być może nawet doświadczenia pedagogicznego.

Kilkanaście lat temu zdecydowanie nie posiadałem kompetencji, aby (nawet sam sobie) ułożyć skuteczny plan edukacji programistycznej. Spójrzmy prawdzie w oczy: najprawdopodobniej Ty również takich kompetencji nie masz.

Programowanie to bardzo szeroka dziedzina. Pamiętam, że na początku swojej drogi próbowałem poznać teorię. Po prostu posiąść wiedzę z książek i specjalistycznych artykułów, bez natychmiastowego aplikowania jej w praktyce. Próbowałem podążyć ścieżką, której nie polecam. Nie polecam, bo wiem, że nie działa.

Wówczas mój proces wyglądał tak:

1. Weź kartkę z listą „pojęć, których się jeszcze muszę nauczyć".
2. Wybierz jedno pojęcie.
3. Wyszukaj artykuł lub książkę na jego temat.
4. W miarę lektury dopisuj do listy pojęć kolejne terminy.

Stosunkowo mało skomplikowane pojęcie – „operacje na plikach", dostępne w większości technologii jako prosta biblioteka – może skutkować dodaniem do listy pojęć między innymi: „mechanizmu działania dysków", „serializacji (binarnej i tekstowej)", „kompresji", „komunikacji sieciowej". Te wszystkie tematy wypada znać. Ale czy są niezbędne już teraz, od pierwszego dnia? Nie!

Widzisz, na czym polega problem skupienia się na teorii? Uczymy się na zapas i nie widać końca tej nauki. A końca faktycznie nie ma. Nigdy nie dotrzesz do momentu, w którym z czystym sumieniem powiesz: no dobra, teraz wiem już wszystko, czas zabrać się do roboty! Zagłębienie się w jedno zagadnienie zawsze – absolutnie zawsze! – spowoduje pojawienie się kilku kolejnych kwestii. To niekończąca się lista – jak popularna zabawa *whack-a-mole*, w której polujemy młotem na wyskakujące z dziur krety. Tylko w tym przypadku za każdym razem uderzenie (niekoniecznie trafne) zaprosi na planszę kilka następnych kretów.

Tej nierównej walki nie da się wygrać.

Oczywiście, bardzo przyjemnie jest po prostu zagłębić się w kolejne tomy teorii. Bierne przeglądanie informacji daje nam uczucie satysfakcji, pożytecznego wykorzystania czasu i bycia produktywnym. No bo przecież się uczę, co nie? Tylko co z tego, skoro jutro (albo za tydzień czy miesiąc) większości materiału nie pamiętam?

Uwaga: to pułapka!

Dlaczego skupienie się tylko i wyłącznie na praktyce daje wspaniałe rezultaty? Dlatego, że – chcąc nie chcąc – i tak przyswoimy wtedy baaaardzo dużo teorii! Programowanie to ciągła nauka. Z tą różnicą, że to nie my ułożymy sobie plan zajęć. Przecież – jako uczniowie, nowicjusze – nie mamy do tego kompetencji. To już uzgodniliśmy kilka akapitów wyżej.

Just-in-time learning is da bomb!

Dopiero wiedza zastosowana w praktyce ma szansę na dłużej zagościć w głowie. Teoria zaaplikowana do rozwiązania faktycznego problemu staje się przydatna. Ją być może zapamiętamy.

Po początkowym etapie zdecydowanie warto sięgnąć po książki. Ale dopiero wtedy! Zerknij na rekomendacje na stronie http://zawodprogramista.pl/ksiazki.

Programista a specjalizacja

Programista ma do wyboru tyle fajnych języków, tyle niesamowitych technologii, że często ciężko jest się oprzeć. I jak z Pokemonami: „muszę znać je wszystkie!". Bez umiaru, bez wytchnienia, bez końca.

To bardzo zdradliwa i niebezpieczna ścieżka. Dlaczego?

> **Know something about everything and everything about something.**
> ***Thomas Henry Huxley***

Wszystko pięknie. Ale programista – szczególnie początkujący – skupia się za bardzo na pierwszej części, ignorując drugą, podczas gdy to ona jest ważniejsza.

W zdobyciu pierwszej pracy, pokazaniu się na rynku, nawiązywaniu znajomości pomaga specjalizacja. Klasyfikacja samego siebie. „Robię front-end", „programuję w Javie", „jestem dotnetowcem", „interesuję się Pythonem". Z góry wszystko wiadomo.

Łapiąc wiele srok za ogon jednocześnie, nie wyspecjalizujesz się. Liźniesz każdego tematu po trosze, ale nie zostaniesz ekspertem. A to właśnie eksperci są najbardziej cenieni na rynku, to ich się poszukuje. Tak jak w innych aspektach życia; prędzej oddam samochód do warsztatu specjalizującego się w mojej marce niż do ogólnego garażu „mechaniki pojazdowej". Bo

> **nie da się umieć wszystkiego.**

Pierwsze lata swojej świadomej edukacji poświęciłem na zgłębianie tajemnic świata .NET. Znałem wszystkie sztuczki w składni C#. Czytałem o zarządzaniu pamięcią, garbage collectorze, kompilatorze. Napisałem nawet kompilator własnego prostego języka, który uruchamiał się na platformie .NET Framework 2.0.

Robiłem to wszystko, żeby być bardzo dobrym w jednej dziedzinie. Nie rozdrabniałem się na poznawanie innych języków i technologii, bo na to być może przyjdzie czas później.

Znajdź odpowiedź na pytanie:

> **Jeżeli ktoś zaprosiłby Cię do programistycznej audycji,**
> **o czym byście rozmawiali?**

I staraj się, by ta odpowiedź odzwierciedlała rzeczywistość. U mnie lata temu odpowiedź brzmiałaby: .NET. Z czasem zmieniłem swój cel i moją specjalizacją stały się testy jednostkowe oraz systemy kontroli wersji.

Wąski zakres. Głęboka wiedza.

Takie podejście pozwoli Ci nie tylko na zwiększenie poczucia własnej wartości i pozycji na rynku pracy, lecz także na zauważenie znacznych postępów w nauce.

Skupienie na jednym temacie i ignorowanie milionów technologicznych rozpraszaczy zdecydowanie zaprowadzi Cię dalej. I szybciej.

Z czasem da się zaobserwować ciekawy efekt kuli śnieżnej. Wąska specjalizacja oraz umiejętne pokierowanie karierą oraz własnym wizerunkiem (tak!) spowoduje, że otrzymasz dodatkowe możliwości rozwoju. Mogą zacząć się pojawiać propozycje wystąpień na meetupach, konferencjach, w podcastach w formie wywiadu. Z czasem może nawet okazje do przeprowadzenia szkoleń? To zmotywuje i zachęci do jeszcze intensywniejszej nauki.

I, co bardzo ważne: decyzja o wyborze specjalizacji nie jest wiążąca na całe życie. W każdej chwili można ją zmienić!

Programista – poliglota?

Lata temu modny stał się trend programisty poligloty, władającego więcej niż jednym językiem.

Czy warto znać więcej języków? Na pewno! Ale jak to się ma do specjalizacji? Otóż: warto, ale nie od razu!

Nie łudźmy się: znajomości języka ani biegłości w technologii nie zdobywa się w kilka wieczorów klepania gry w kółko i krzyżyk albo kolejnej to-do-listy. Do tego potrzeba czegoś więcej: poważniejszego projektu, często realizowanego w modelu pracy zespołowej. Projektu, który nie tylko rozpoczniemy, ale również utrzymamy; wdrożonego, sprzedanego i, co najważniejsze, używanego. To wtedy powstają prawdziwe problemy. Wtedy widzimy, gdzie dana technologia się sprawdza, a gdzie klęka i błaga o litość.

A co to oznacza? Że zdobywając biegłość przy jednym projekcie w technologii X, nie zdobywamy jednocześnie biegłości przy innym projekcie w technologii Y. Pomijając rzadkie wyjątki, tak się nie da.

Czy zatem nie ma sensu poszerzać horyzontów i bawić się poza własną piaskownicą? Sens jest, ale z pewnymi założeniami. Jak się bawić, to się bawić! Zabawa Javą dla programisty .NET nie zmieni jego perspektywy, bo środowiska te są dość zbliżone.

Skupmy się na porządnym poznaniu jednego, by dopiero potem zabierać się za drugie. I nie łudźmy się, że wyspecjalizujemy się w wielu obszarach jednocześnie.

Warto także zauważyć, że pewne warunki pracy wymagają, by zostać programistycznym poliglotą, czy tego chcemy, czy nie. Przez całą swoją karierę byłem tak zwanym full-stack developerem – musiałem programować w SQL (na różnych silnikach baz danych: MS SQL Server, MySQL, Oracle, Postgres), C# i JavaScript (z wszystkimi jego frameworkami). Nie uciekałem także od HTML i CSS.

Często doświadczony programista nie musi specjalnie uczyć się nowych technologii. Doświadczenie zdobyte w jednym środowisku jest niejednokrotnie aplikowane w zupełnie innym projekcie.

⇨ *Przez całe zawodowe życie byłem związany głównie z technologiami Microsoftu. Mimo to bez większego problemu poprawiłem bug w Redmine, czyli dużym systemie do zarządzania projektami, napisanym w Ruby. Bez trudu również upraszczam swoje życie blogera, dodając własne pluginy do WordPressa (PHP).*

Zadaniem zawodowego programisty jest myślenie i… programowanie. Ekspert czy ekspertka wykona pracę niezależnie od technologii. Bo doświadczenie i specjalizacja zdobyte na jednym polu prawie zawsze przydadzą się również w innych obszarach.

⏎

Jaki język wybrać na początek?

Początkujący programiści stoją przed kilkoma pytaniami, które mają wpływ na start edukacji i pracy w zawodzie. Niektóre z tych pytań są trudne, ponieważ wymagają szybkiej nauki wielu nowych pojęć. Dlatego też odpowiedzi na nie często będą – w dłuższej perspektywie – błędne. Na szczęście

> **każda decyzja podjęta na początku programistycznej drogi jest całkowicie odwracalna.**

Pierwsze pytanie, jakie zadaje osoba początkująca, to: „no dobra, chcę programować, ale w jakim języku?". O najważniejszym języku z perspektywy programisty już rozmawialiśmy, ale rozumiem, że taką decyzję trzeba prędzej czy później podjąć. Przecież uczymy się czegoś jednego, konkretnego, wybranego.

Zamiast wbijać do głowy sztywne rekomendacje, podejdziemy do tego problemu nieco sprytniej. Okazuje się bowiem, że jedno pytanie pomocnicze przygotuje grunt pod decyzję. Brzmi ono:

> **Jakiego typu aplikacje chcesz pisać?**

To nie wymaga zrozumienia meandrów branży od samego początku. A może wytyczyć kierunek rozwoju od pierwszego dnia.

Web – front

„Strony internetowe" – brzmi bardzo średniowiecznie, ale jednak... Od portali społecznościowych, przez portale aukcyjne, strony z wiadomościami, fora internetowe, do czegokolwiek innego, co odpalasz codziennie w swojej przeglądarce.

Na to, co widzisz, składają się przynajmniej dwa elementy: front-end i back-end.

Jeżeli chcesz pisać „strony WWW", zacznij od kombo: JavaScript (czyli JS) + CSS + HTML. Jest to zestaw narzędzi front-end developera.

Z JavaScript wiąże się wiele innych pojęć, nazw – frameworków i bibliotek. Każdy język ma takie pomocnicze rozszerzenia, ale to w JavaScript najpewniej od pierwszej minuty się z nimi spotkasz. Mogą to być na przykład: Angular, React, jQuery, Knockout, Vue, Ember, Meteor... W tym przypadku lista ciągnie się i ciągnie, nawet gdybym ją uzupełnił kwadrans przed zakończeniem książki, to w tym czasie powstałoby w światku JS coś nowego. Wybierz którąś z bibliotek – i do dzieła!

Web – back

Ogromna większość stron internetowych musi mieć zaplecze na jakimś serwerze. Popularnymi językami pozwalającymi realizować takie zadania są PHP i Ruby. Obie te technologie wspomagane są frameworkami, czyli rozszerzeniami ułatwiającymi pisanie aplikacji.

W świecie Ruby prym wiedzie Ruby on Rails. W świecie PHP wybór jest bardziej zróżnicowany: Symfony, Laravel, Kohana czy Zend.

Aplikacje internetowe można napisać także, korzystając z innych narzędzi, ale nimi zajmiemy się niżej.

Duże systemy / korpoświat

Świat bankowości, ubezpieczeń i innych korporacji jest w dużej mierze oparty na dwóch rodzinach narzędzi: Java oraz .NET. Java to nazwa zarówno całego środowiska, jak i języka, a .NET – technologii, której najpopularniejszym językiem jest C#.

Oczywiście zarówno w Javie, jak i w .NET powstają także mniejsze rozwiązania. Sam takie pisałem. W tym rozdziale jednak zajmujemy się tym, co najpopularniejsze i „standardowe".

Oba te środowiska mogą okazać się dobrym wyborem, jeśli chodzi o jak najszybsze znalezienie pracy.

Aplikacje mobilne

Znamy je wszyscy, korzystamy z nich na co dzień. To, co znajduje się na naszych telefonach. Raz cieszy, raz bawi, raz zaskakuje, a nieraz irytuje.

Na Androida pisze się w Javie (lub innych językach na platformę Java, na przykład Kotlin); na iOS – oczywiście w technologiach od Apple. Językiem standardowym i znanym od dawna jest Objective-C. Kilka lat temu Apple wydało nowy język programowania – Swift. Najlepiej nauczyć się obu, zaczynając od Swift, który jest nowszy, bardziej elegancki i łatwiejszy do przyswojenia.

Mamy także możliwość pisania aplikacji, których duża część jest przenaszalna między urządzeniami. Jedną z technologii próbujących rozwiązać problem wielu producentów/platform/środowisk jest Xamarin. To narzędzie umożliwia pisanie w technologiach Microsoftu (C#) i uruchamianie aplikacji na (prawie) wszystkich urządzeniach mobilnych.

Gry

Ile osób zajęło się programowaniem, bo kiedyś grały w gry i chciały samodzielnie jakąś stworzyć?

Gra grze nierówna, lecz myśląc o grach „pełnych", na komputery czy konsole, zerkamy w stronę C++. To nie jest prosta zabawa i może wymagać bardzo dobrej znajomości matematyki, której omówienie mamy już za sobą.

Analiza danych / nauka

W ostatnich latach bardzo popularne stają się tematy typu: *data science*, *machine learning*, *artificial intelligence*. Trend ten zdecydowanie się utrzyma.

Najbardziej popularnym językiem wydaje się w tym zakresie Python. To bardzo uniwersalny język o wielu zastosowaniach i pisze się w nim także innego typu aplikacje, lecz pierwsze skojarzenie pojawiające się przy terminie „analiza danych" to właśnie Python.

Ekstrawagancje

Języków i technologii jest cała masa. Bardzo, bardzo dużo. Bez sensu jednak rzucać się na samym początku na język niszowy. Bo i po co?

Eksplorację mniej popularnych obszarów zostawmy na później. Jako sposób na poszerzenie horyzontów, gimnastykę dla programistycznego umysłu.

A jaki język ja bym dzisiaj wybrał?

Żeby być *fair*, postawię się na moment w sytuacji *software developer wannabe*. Niczego nie umiem, nie wiem, gdzie zacząć, ale wierzę, że będzie dobrze. Sakramentalne pytanie: od jakiego języka zacząć? Jestem już szczęśliwie świadomy tego, że nie jest to decyzja na zawsze i że w dosłownie każdej chwili wybór ten mogę zmienić.

Programowanie jako zawód

Do tej pory powinno być już oczywiste, że najwięcej można się nauczyć w pracy. Moim celem jest zatem jak najszybsze jej zdobycie, na dowolnym stanowisku związanym z programowaniem.

Najłatwiej będzie mi zdobyć pracę w języku, którego programiści i programistki są poszukiwani w moim regionie. Proste, prawda? Zatem wchodzę na wszystkie znane mi portale z ofertami pracy i wyszukuję stanowiska programisty lub stażysty w moim mieście. Zliczam ogłoszenia według języka i wybieram ten pojawiający się najczęściej.

Uczę się (postępując zgodnie z poradami w tej książce) i po kilku miesiącach wskakuję na stanowisko praktykanta lub programisty w dowolnej firmie. I moja edukacja rusza z kopyta.

Hobby

Istnieje jeszcze druga droga: nie zamierzam pracować jako programista! Chcę po prostu zobaczyć, o co w tym wszystkim chodzi – hobbystycznie. Sprawdzić: a nuż mi się spodoba?

Wtedy zaczynam od Pythona. Nie bez przyczyny język ten jest używany w nauce dzieci, w szkołach, na maturze. Mało tego, kiedyś przez wiele miesięcy próbowałem uczyć C# dorosłą osobę, która zupełnie nie umiała programować. Efekty nie były zadowalające. Punktem przełomowym okazało się przejście na Pythona.

Jest to po prostu dobry język na start dla hobbysty.

Jaki framework wybrać na początek?

Przed początkującymi programistami i programistkami wiele trudnych decyzji. Wybrać język, a potem jeszcze jakiś framework!

Gwoli wyjaśnienia: framework to kod „zewnętrzny", cudzy. W zależności od konkretnego przypadku może zawierać bardzo wiele przydatnych funkcji. Framework uruchamia Twój kod i dostarcza gotowe komponenty, klocki, części systemu, wspólne dla wielu aplikacji – nie tylko Twoich.

Wybór

Technologie mają wiele różnych frameworków. Do budowania aplikacji internetowych w .NET mamy ASP .NET MVC, WebForms czy Nancy. W PHP będzie to Symfony, Laravel albo Kohana. W Ruby znajdziemy Ruby on Rails, Sinatrę. W Pythonie – Django, web2py, Pyramid. Słowem: jest tego dużo.

Nie będę sztucznie budował napięcia i od razu odpowiem na dręczące Cię być może pytanie: który framework wybrać? Uwaga:

> **Nie ma to żadnego znaczenia!**

Wybierz pierwszy lepszy i napisz w nim coś. Nie marnuj czasu na zastanawianie się.

Ale dlaczego? Oto prawdziwe pytanie!

Każdy choć trochę doświadczony programista wie, że nawet wybór technologii nie jest związkiem na całe życie. Tak samo język: zaczynasz od jednego i w miarę rozwoju uczysz się kolejnych. Poruszyliśmy już zresztą ten temat w rozdziale o programistycznych poliglotach.

A framework? Tym bardziej!

Framework-driven development

Pamiętaj: uczysz się programowania, a nie frameworka! Można czasami zaobserwować bardzo niebezpieczne zjawisko: *framework driven development*. Oznacza to skupianie się na jak najszybszym wykonaniu zadania w danym frameworku, stosując się jedynie do jego wytycznych. Programistyczne *best practices* są wówczas mniej istotne. Liczy się znajomość frameworka – i już.

To bardzo niebezpieczne podejście.

Zmiana technologii czy wykorzystywanego języka nie jest niczym nadzwyczajnym, a programowanie z pomocą kilku frameworków w kilku różnych projektach jest na porządku dziennym. Wiedza podporządkowana jednemu wcale nie musi znaleźć zastosowania w innym! Dlatego też skupiajmy się na usprawnianiu umiejętności programowania. Nie jest to jednoznaczne ze znajomością (często ograniczoną) jak największej liczby frameworków.

Dokumentacja

Kolejna istotna kwestia to dokumentacje frameworków. Może to być fascynująca lektura, bardzo dokładnie tłumacząca założenia i decyzje podjęte podczas jego projektowania. Jednak jest pewne „ale".

Dokumentacja frameworka nie służy propagowaniu „dobrych praktyk" programistycznych. Jej cel to pokazanie programiście, co może osiągnąć za pomocą danego kodu. Dlatego też bardzo często podane w takiej dokumentacji przykłady wołają o pomstę do nieba, jeśli chodzi o „uniwersalną poprawność" oraz zgodność z ogólnymi regułami panującymi w naszym światku.

To normalne. Frameworków mamy baaaardzo dużo i nierealne jest, by każdy twórca dokumentacji pisał własne wersje rekomendacji najlepszych programistycznych praktyk. Oni, tak jak ja i Ty, skupiają się na swoim zadaniu. Ich zadaniem jest przedstawienie nam możliwości danego frameworka, a nie dbanie o naszą programistyczną edukację.

Pamiętaj, proszę, o tym.

Framework-less?

Osobiście lubię nawet iść nieco dalej i rozwiązywać zadane problemy bez użycia jakiegokolwiek frameworka. Mój kod – w oderwaniu od całej otoczki wymaganej do uruchomienia – stanowi efekt mojej pracy.

To jest prawdziwa sztuka. Zrozumieć problem i napisać kod niezależny od frameworka. *Framework-less development FTW!*

Wpięcie go w jakiś framework, uruchomienie i sprawienie, by zadziałał, to wtedy najczęściej banalna zabawa.

Czy frameworki są zbędne?

W takim razie może rzućmy to wszystko i zapomnijmy o frameworkach? O, nie! One powstały w słusznym celu: mają skrócić czas wytwarzania oprogramowania przez zdjęcie z naszych barków powtarzalnych zadań. Dzięki temu w każdym projekcie możemy skupić się na tym, co istotne w tym konkretnym przypadku. Na problemie klienta, dziedzinie systemu, modelowaniu projektowej rzeczywistości, a nie klepaniu w kółko kawałków kodu odpowiadających za te same, wspólne funkcje każdej aplikacji na świecie.

Zrozumienie powodów powstania frameworków oraz przyczyn wykorzystywania ich w projektach może być Twoim kolejnym krokiem w kierunku programistycznej dojrzałości. One są dla nas, a nie my dla nich.

Świadomy programista i świadoma programistka będą skakać między frameworkami i nie będzie to miało wpływu na serce systemu. Nie bądźmy frameworkowymi niewolnikami.

Praca...

W pracy zawodowej – w prawdziwym, dużym projekcie – niejednokrotnie nawet nie zobaczysz frameworka wykorzystanego do jego stworzenia. Systemy budowane latami opierają się na własnych abstrakcjach, integrujących kod z wybranym rozwiązaniem „uruchamiającym".

Służy to uniezależnieniu się od jednej konkretnej implementacji. Dzięki temu przejście na inny (albo na nowszą wersję aktualnego) jest nie tylko możliwe, ale także niejednokrotnie stosunkowo proste.

Kolejny krok w edukacji

Wypisz wszystkie frameworki, które rozważasz, i rzucaj monetą tak długo, odrzucając po jednym z nich, aż zostanie ten ostatni.

Alternatywnie możesz w wyszukiwarce wpisać „framework" i wybrać pierwszy, który wpadnie Ci w oko.

A co dalej? W swojej programistycznej karierze i tak prawdopodobnie napiszesz wiele projektów hobbystycznych. Postaraj się, aby każdy z nich używał innego frameworka. Poznaj wiele, poszerzaj horyzonty, nie przywiązuj się do jednego.

Naprawdę. Nie marnuj czasu i energii na podejmowanie decyzji niemających żadnego znaczenia.

⏎

Jaki projekt napisać na początek?

Kolejne obowiązkowe pytanie początkujących.

Poniżej znajdziesz wiele pomysłów na dev-wyzwania. Po lekturze tego rozdziału zapomnisz o pytaniu: „jaki projekt wymyślić?". Zostaniesz z problemem: „którą z propozycji wybrać?".

Podzielę się też skróconą historią moich projektów służących mi za kolejne etapy zdobywania programistycznej wiedzy.

Niezależnie od dalszych decyzji – jest jedna bardzo ważna wskazówka, którą w tym miejscu tylko zasygnalizuję, a omówimy ją w kolejnej części książki. Każdy zrealizowany projekt koniecznie dodaj do swojego CV wraz z linkiem do kodu źródłowego na GitHubie (nie wiesz co to? spokojnie, jeszcze do tego dojdziemy!). Link do działającej, uruchomionej wersji demonstracyjnej na pewno także zadziała na plus.

Pamiętaj też, że rozpoczęta aplikacja nie musi zostać zrealizowana do końca. Nie zostawiaj rozgrzebanej, niedziałającej, nienadającej się do niczego pracy, ale też nie trać czasu na dopieszczanie każdej swojej inicjatywy w 100%. To Ty definiujesz, kiedy dane przedsięwzięcie dobiegło końca, zostało zaimplementowane w stopniu zadowalającym. Osiągnij swój cel i ruszaj dalej.

Zaczynamy!

Pomysły na projekty

Niektóre osoby mają programistyczne marzenia. Zanim jeszcze napiszą linijkę kodu, już wiedzą, jakie będzie ich pierwsze zadanie. Szczęściarze! Szczęściary! Ja niestety do nich nie należałem.

Twoje otoczenie jest pełne pomysłów i inspiracji. Przeanalizuj swój jeden typowy dzień. Z ilu aplikacji korzystasz na telefonie? Na ile portali logujesz się podczas codziennych zadań? Jakie programy wywołują u Ciebie uśmiech, a które – dreszcze zgrozy? Stwórz listę swojego „cyfrowego ja".

Z tej listy aplikacji wybierz jedną dowolną pozycję i postaw sobie za cel zbudowanie jej klona. Od zera. W ten sposób nie będziesz tracić czasu na zastanawianie się, „a jak to ma wyglądać?" ani „co to ma dokładnie robić?". To już to wiesz! W końcu tej aplikacji używasz na co dzień.

Dalej Ci mało? Proszę bardzo!

Napisz sieciową grę w statki! Nauczysz się rysowania (plansza), podstawowych mechanizmów sterujących grami oraz komunikacji między komputerami.

Klon Facebooka czy GoldenLine! Nie stworzysz oczywiście tak zaawansowanego rozwiązania, ale i tak poznasz wyzwania, przed jakimi stają programiści i programistki aplikacji internetowych.

Odtwarzacz muzyki albo filmów. Poznasz operacje na plikach, biblioteki do obsługi audio-wideo oraz specyfikę tworzenia aplikacji desktopowych – uruchamianych lokalnie na komputerze.

Program do ewidencji przeczytanych książek. Zobaczysz, na ile sposobów można przechowywać dane, i nauczysz się na nich operować.

Strona porównująca dwa konta na Twitterze. Taka, która zaprezentuje wspólnych znajomych i interakcje między wybranymi użytkownikami. Zobaczysz, co znaczy „wykorzystanie zasobów zewnętrznej aplikacji".

Klient e-mail. Poznasz dwa z najstarszych i najczęściej używanych protokołów w internecie: SMTP i POP3.

Gra planszowa na telefon albo przygodówka na konsolę. Dowiesz się, czym charakteryzuje się dana platforma i zaszpanujesz przed znajomymi.

Aplikacja do zarządzania czasem. Stoper, timer, budzik. Nauczysz się postrzegać upływ czasu tak, jak robią to komputery.

Możesz też pokusić się o coś „sprzedawalnego", wymagającego wiedzy nie tylko programistycznej. Na przykład system do wystawiania faktur, zgodny z obowiązującym prawem. Z generowaniem PDF-ów, wyszukiwaniem danych firm po numerze NIP... To trudniejsze, niż może się wydawać!

Mało Ci? Wpadnij na http://dajsiepoznac.pl. Daj Się Poznać to organizowany przeze mnie konkurs, w którym w sumie wzięło udział ponad tysiąc polskich programistów i programistek. Przejrzyj ich pomysły na własne projekty i... zainspiruj się!

Najlepsze, co możesz zrobić, stawiając swoje pierwsze kroki w programistycznym świecie, to założenie dev-bloga i dokumentowanie tam swojej drogi. Podsuwam Ci tę myśl już któryś raz. I nie po raz ostatni. Coś jest na rzeczy!

Moja droga

Ja swoje pierwsze programistyczne kroki stawiałem na początku XXI wieku – dość dawno temu. Niektóre z moich poczynań dzisiaj nie miałyby sensu, ale to były czasy, w których nie każdy miał dostęp do internetu, nie było Facebooka, a telefon służył tylko do dzwonienia i nie można go było nazwać *smart*. Stare dzieje.

Zacząłem od totalnych podstaw i na pierwszy ogień poszła znajomość HTML i CSS. Stworzyłem banalną stronkę o mojej pasji: muzyce. Na stronie można było znaleźć teksty piosenek, okładki płyt i moje recenzje ulubionych albumów. A do tego wielki pływający (*marquee* – kto pamięta?) napis: „Maciej Aniserowicz WITA!!!". Nic wielkiego: goły tekst i obrazki. Całość napisana w Notatniku. I wrzucona na jakiś darmowy hosting za pomocą połączenia z numerem 0202122 przez modem *dial-up* z oszałamiającą prędkością 56 Kb/s.

Później zacząłem odkrywać podstawy JavaScript w Netscape Navigator i Internet Explorer 4, 5 albo 6. Zmierzyłem się z wyzwaniem strony z animowanymi wampirami. Poległem. Ale animacje zrealizowałem później w tak zwanym zegarze binarnym. Polecam jako łamigłówkę.

Następnym etapem był C++ i wygaszacze ekranu. Nauczyłem się krojenia grafiki i poruszania elementów obrazka. I konfiguracji aplikacji tak, by pełniła funkcję wygaszacza, czyli zajmowała cały ekran, wskakiwała na wierzch wszystkich pozostałych okien oraz zamykała się przy dotknięciu myszki lub klawiatury.

Później nadszedł czas na poważniejsze zmagania – komunikację sieciową. Napisałem pierwszy program, którego faktycznie używałem na co dzień. Moja aplikacja łączyła się ze stroną komunikacji miejskiej, pobierała HTML i kroiła otrzymany tekst, prezentując rozkład jazdy wybranych autobusów w bardziej przyswajalnej formie. Wtedy istotne były funkcje zapisu tych danych lokalnie i trybu pracy *offline*, bo – jak wspominałem – nie do każdego domu sięgał internet.

Komunikacja sieciowa bardzo mi się spodobała. Zbudowałem więc własny komunikator internetowy: klon pamiętnego Gadu-Gadu. Oczywiście komunikator

ten miał tylko jednego użytkownika: mnie. Komunikować mogłem się zatem wyłącznie ze sobą. Dlatego też dopisałem do niego kompatybilność z GG. Od tej pory mogłem używać swojego komunikatora, rozmawiając ze znajomymi korzystającymi z oficjalnego Gadu-Gadu.

W połowie studiów w trzyosobowym zespole napisaliśmy program do produkcji filmów animowanych. To oczywiście brzmi bardzo dumnie, o wiele poważniej, niż wyglądało w praktyce. Ale faktem jest, że za pomocą naszej aplikacji można było wyprodukować fruwające, kręcące się obiekty, zmieniające swoją pozycję w czasie.

Jednym z ambitniejszych projektów studenckich był wspomniany już wcześniej kompilator własnego języka programowania. To wyzwanie bardzo wiele mnie nauczyło. Stworzyłem własny język programowania, który – po skompilowaniu moim kompilatorem – uruchamiał się i działał na platformie .NET. Ciekawe doświadczenie.

Próbowałem też swego czasu zaimplementować własny komunikator audio-wideo do rozmów przez kamerkę i mikrofon. To jedna z moich największych porażek, bo co prawda dźwięk i obraz były przesyłane z jednego komputera na drugi, ale nie udało mi się doprowadzić tego do stanu „używalności". Od tamtej pory minęło kilkanaście lat, a ja nadal boję się obróbki audio-wideo.

Jak widzisz, pomysłów są dziesiątki. Setki. Pomysłów jest nieskończona liczba.

I nie ma większego znaczenia, od czego zaczniesz.

Ważne, by w końcu to zrobić.

Co wybierasz?

3 Pierwsza praca

Czy staż powinien być płatny?

Pytanie podchwytliwe? Z haczykiem? Bo przecież wiadomo, że za pracę się płaci, prawda? Ano nie zawsze. To zależy od kontekstu i okoliczności. Zależy od celu oraz kierunku dostarczania większej wartości, niekoniecznie w postaci pieniędzy.

Wiele razy nazywano mnie naiwniakiem, bo przez pierwsze lata swojej „aktywności zawodowej" pracowałem za darmo albo za symboliczne wynagrodzenie. Jednak takie podejście pozwalało mi przebierać w bardzo interesujących projektach. Poznać dziesiątki fascynujących osób. Nawiązać kontakty, które przetrwały kilkanaście lat i utrzymują się do dziś! Zdobyłem doświadczenie dokładnie w tych obszarach, na których mi zależało… oraz wielu innych. W ówczesnych okolicznościach nie byłbym w stanie pogodzić tych wymagań z godziwą pensją.

Kto bardziej skorzysta na udziale w projekcie niedoświadczonego, początkującego, nieopierzonego programisty? Ten zdobywający tak potrzebne doświadczenie, czy firma, zmieniająca się de facto w przedszkole i nianię?

Nawet senior przez pierwsze tygodnie jest w zespole kosztem!

A co dopiero junior! Ktoś musi się Tobą opiekować. Ktoś musi nauczyć, wprowadzić w system. Pokazać co i jak. Tobie zostanie w głowie wiedza; opiekunom czasu nikt nie zwróci.

Poza tym, jeśli się sprawdzisz, to – kto wie – może po takim eksperymencie zostaniesz w firmie na dłużej, już na normalnym stanowisku programistycznym?

Przez ostatnią dekadę przeprowadziłem wiele ciekawych inicjatyw. Animowałem kipiące pasją społeczności. Odnosiłem sukcesy na różnych polach, nawet teoretycznie niezwiązanych z zawodem programisty. To wszystko nie wydarzyłoby się, gdyby nie te pierwsze lata. Lata zainwestowane w całe moje przyszłe – wspaniałe – życie.

Oczywiście nie każdy może sobie pozwolić na niezarabianie. Jeśli pieniądze są potrzebne już teraz, przez cały czas, to mówimy nie o stażu, praktykach czy

zdobywaniu doświadczenia, tylko po prostu o pracy. A co jest najważniejsze przy poszukiwaniu pracy? Do tego tematu jeszcze dojdziemy, ale możemy wybiec lekko w przyszłość: oczywiście doświadczenie!

To nie jest pozwolenie złowrogim firmom na bezwzględne wykorzystywanie biednych krzywdzonych developerków. To wkład w przyszłość. Twoja najpewniejsza lokata. Jeszcze się nazarabiasz.

Nie zachęcam bynajmniej do pracowania za darmo latami – oczywiście, że nie! Ale można spotkać się z przegięciem w drugą stronę: wymogiem płacenia sobie nawet za udział w rozmowach kwalifikacyjnych.

Która perspektywa wydaje Ci się bardziej oderwana od rzeczywistości?

Kim jest junior developer?

Nazwy stanowisk w IT nieco się w naszym języku zdewaluowały. Zastanówmy się, na jakie kategorie można podzielić programistów. Na początku jest „ktoś, kto chce być programistą" i kogo nazywam software developer wannabe. Po spełnieniu marzenia taka osoba zostanie junior developerem, czyli młodszym programistą. Następnym szczebelkiem jest mid albo regular – zwał jak zwał. Później mamy senior developera – starszego programistę. I to koniec.

Junior... czyli kto?

Senior developerem zostałem przed ukończeniem trzydziestego roku życia. W tym wieku lekarze czy prawnicy jeszcze nawet na dobre nie rozpoczynają swojej kariery zawodowej. A w IT? Już wtedy nie za bardzo wiedziałem, co dalej. Muchos señor developer? Zombie developer? Czy po prostu: dziad?

Granice między poszczególnymi etapami są dość płynne. Dodatkowo w różnych firmach stanowiska o takich samych nazwach charakteryzują się różnym

zakresem obowiązków. Nie ma jednej uniwersalnej definicji. I nawet nie będę się starał takiej stworzyć. Junior to po prostu ktoś na początku drogi. Pod względem doświadczenia, nie wieku.

Juniorem jest osobą dopiero zaczynającą swoją przygodę z programowaniem. Jest się nim też w pierwszej pracy, w której zdobywa się początkowe doświadczenia. I tu, nie mogąc się powstrzymać, wbiję szpilę w niektóre wygodne środowiska.

> **Juniorem jest także programista**
> **robiący dokładnie te same zadania przez dekadę;**
> **taki programista nie ma dziesięciu lat doświadczenia,**
> **tylko jeden rok powtórzony dziesięciokrotnie!**

Pozwól tej głębokiej myśli na zagnieżdżenie się w umyśle...
I kontynuujmy.

Co powinien umieć junior developer?

Junior developer to osoba, od której nie można wymagać wiele. Bo i jakim prawem? To ktoś, kto chce się uczyć. Kto chce zdobywać doświadczenie i się rozwijać. Wreszcie: to ktoś świadomy własnych niedoskonałości... i pełen zapału, by jak najprędzej je wyeliminować.

Głównym zadaniem młodszego programisty, celem numer jeden w ciągu zawodowego dnia, jest nauka. Nieustanne starania, by wskoczyć na wyższy poziom. Można oczekiwać od takiej osoby konkretnych umiejętności oraz profesjonalizmu, jednak

> **po juniorze nie należy spodziewać się eksperta!**

Oferty pracy na stanowisko „junior developera z dwuletnim doświadczeniem w X i biegłą obsługą Y" są po prostu niezbyt wysublimowanym zabiegiem mającym na celu wypłacenie pensji juniora programiście zasługującemu na nieco więcej. To proste prawa rynku.

Wszyscy jesteśmy juniorami?

Skoro od junior developera nie można zbyt wiele wymagać, to może wszyscy jesteśmy junior developerami już od pierwszej chwili, gdy tylko w głowie pojawi się myśl: „będę programować zawodowo"? Nie.

To prawda, junior się uczy i dopiero zdobywa doświadczenie. Ale jeśli junior pracuje, to za tę pracę otrzymuje wynagrodzenie. Musi więc dostarczać pracodawcy pewną wartość.

Do stanowiska junior developera trzeba się przygotować. Jeżeli wiemy już, że interesuje nas konkretna technologia, firma czy dany projekt, to po prostu uczymy się tej jednej rzeczy. Czasami jednak nie do końca wiadomo, gdzie i w jakiej roli wylądujemy za kilka miesięcy. Wówczas warto skupić się na umiejętnościach uniwersalnych. Wymagana na stanowisku junior developera są tak różne, że nie da się ich wszystkich nauczyć na zapas.

Juniorrekomendacje

Mam dla Ciebie kilka rekomendacji (oczywiście jako dodatek do pozostałych, które znajdziesz w tej książce).

Po pierwsze, zapoznaj się z pojęciem „kontrola wersji". Poczytaj, co to jest, pobaw się narzędziami. Zrozum, czym różni się „scentralizowany" system kontroli wersji od „zdecentralizowanego". Może się to wydawać zbyt zaawansowane dla osoby początkującej, ale wcale takie nie jest. W każdym projekcie, w każdej firmie i każdym zespole – niezależnie od tego, gdzie trafisz – spotkasz się z narzędziem tego typu. Rozeznanie w temacie zajmie Ci maksymalnie kilka wieczorów.

A gdy już zrozumiesz ideę stojącą za kontrolą wersji oraz poznasz różnice między różnymi dostępnymi typami, naucz się podstaw Gita (chociaż mam nadzieję, że już na wcześniejszym etapie Git wpadł Ci w ręce). Zacznij tego Gita używać nawet w swoich własnych, indywidualnych, jednoosobowych projektach. Ba, zacznij go używać nawet poza zadaniami programistycznymi! (Ja jestem uzależniony od Gita; trzymam w nim nawet slajdy do prezentacji).

Po drugie, zdobądź minimum wiedzy o programowaniu obiektowym. Nie od razu zrozumiesz definicje takich pojęć, jak „polimorfizm", „dziedziczenie" czy „enkapsulacja". Ale nie musisz. Ważne, żeby słowa te wryły się w pamięć, by programowanie obiektowe tkwiło cały czas gdzieś z tyłu głowy. W końcu wszystkie elementy dopasują się do siebie, zaskoczą. Ale trzeba stworzyć im odpowiednie warunki. Po prostu: ucz się.

Po trzecie, zbadaj temat testów jednostkowych. Nie osiągniesz mistrzostwa w kilka dni; zajmuje to długie lata. Ale idea, koncepcje i praktyki testowania, nawet jeśli nie do końca zrozumiałe, niech zapuszczają korzenie. Sam fakt próby zrozumienia tego sposobu pracy z kodem – znajomość terminów, umiejętność porozmawiania na ten temat podczas rekrutacji – znaczy bardzo wiele. Wierz lub nie, ale całe rzesze doświadczonych zawodowych programistów i programistek są w tym zakresie na Twoim poziomie. Niestety.

Po czwarte: zapoznaj się z tematem relacyjnych baz danych. Wybierz którąkolwiek i po prostu dowiedz się, „co to jest". Zrób sobie bazkę! Możesz wybrać PostreSQL, SQL Server, Oracle, MySQL... Nie ma znaczenia. Na podstawowym poziomie niczym się one nie różnią, a prawie każda aplikacja na świecie przechowuje dane właśnie w bazie danych!

I wreszcie: kontrowersyjny temat wzorców projektowych. Nie ma się co oszukiwać, to JEST wyższa szkoła jazdy. To jest pojęcie zaawansowane, z którym nawet doświadczeni developerzy sobie nie radzą. Ale może nawet nie próbowali? Spróbuj Ty! To będzie gimnastyka dla Twojego mózgu. Sprawdzian programistycznego rozumowania. Sięgnij po legendarną książkę „Design Patterns" autorstwa czterech autorytetów okrzykniętych mianem *Gang of Four*. Została ona napisana ponad dwie dekady temu i... zagotuje Twoją czachę. Zdobyta z niej wiedza nie przyda Ci się bezpośrednio w żadnym projekcie (a przynajmniej nie w najbliższej przyszłości), ale dzięki tej lekturze nauczysz się pokory. Wskoczysz na wyższy poziom wtajemniczenia. Pamiętam swoją pierwszą przygodę z tą książką; tony irytacji oraz udawanie, że rozumiem. I wiem, jak wiele znaczyła ona dla mojego rozwoju oraz wzrostu świadomości.

Junior developer potrafiący ze zrozumieniem porozmawiać o Gicie, obiektówce, testach, bazach i wzorcach dostaje u mnie pracę z marszu. Reszty douczy się w robocie.

Jak zdobyć doświadczenie programistyczne?

Na rynku pracy w IT kluczową rolę odgrywa doświadczenie. Często same „umiejętności", nawet poparte papierami, nie wystarczą. I taka sytuacja ma bardzo sensowne uzasadnienie. Czysta „nauka" to tylko teoria. A w pracy masz umieć ją zaaplikować, co potwierdza jakże mądry cytat:

> **In theory, there is no difference between theory and practice.**
> **But, in practice, there is.**
> ***Jan L.A. van de Snepscheut***

Bez pracy nad produktem, bez dbania o ten żywy organizm, bez poprawiania błędów, kontaktu z użytkownikami, bez przejścia przez cały cykl życia oprogramowania – nasza wiedza jest tylko cząstkowa. W najlepszym wypadku: połowiczna. Bo nie jesteśmy przygotowani na sytuacje awaryjne.

Prawdziwe doświadczenie to nie tylko proces tworzenia nowego systemu. To także wszystko, co dzieje się po jego uruchomieniu! Wdrożenie, przekazanie użytkownikom i reagowanie na błędy. Przygotowywanie poprawek, ich instalowanie, kolejne wdrożenia, monitorowanie. Wszystko to „na już", „na teraz", często „na wczoraj". Nie działa, kasa ucieka, ludzie się wkurzają, szef krzyczy („jak mogliście do tego dopuścić?!, za kwadrans ma działać!!"), stres, prawdziwy test zespołu i procesów.

Przywracanie utraconych danych, łatanie dziur, walka z czasem, skakanie po środowiskach, porównywanie baz danych, omijanie błędów w wykorzystanych

bibliotekach, analizowanie logów... a to wszystko z brzytwą na gardle, z pistoletem przystawionym do skroni. Ze świadomością, że każda sekunda zwłoki to straty liczone w workach złota.

Uśmiechasz się ze zrozumieniem, rumieńce wyskoczyły na policzki? Piona!

Szepczesz: „o czym on gada; nikt by nie chciał pracować w takim miejscu". Poczekaj, poczekaj, przyjdzie kryska na Matyska.

Tak właśnie – w boju, w programistycznych okopach – zdobywa się prawdziwe szlify. Cenne doświadczenie.

Developer Ryan

Skąd wziąć doświadczenie?

Im wcześniej przejdziesz chrzest bojowy, tym lepiej. Spadną różowe okulary, zrzucisz klapki z oczu. Twoje wyobrażenie o pracy w IT ulegnie zmianie na zawsze.

Nie ma lepszego sposobu niż normalna praca przy prawdziwym, komercyjnym projekcie. Takim, który nie jest zabawą w prototypowanie, ale faktycznie trafia do użytkowników. Dlatego tak bardzo podkreślam, że zaczepienie się jako normalny (pół)etatowy programista jest kluczowe dla dalszego rozwoju. Pierwszą pracę znaleźć najtrudniej, a może być ona najważniejsza, bo to w niej wyobrażenia zderzają się z rzeczywistością.

Sam nie miałem szczęścia i w pierwszej pracy nie spotkałem się z żadnymi wyzwaniami. Straciłem kilka miesięcy życia i wyniosłem z niej jedynie nieprzychylne – i niezgodne ze stanem faktycznym – zdanie o pracy w korporacji.

Jak podnieść swoją wartość na rynku? Jak ułatwić znalezienie pracy marzeń? Pójść do firmy – jakiejkolwiek. Nie ma lepszego sposobu. Zawsze się dużo nauczysz; nawet jeśli nie w temacie programowania, to życiowo.

⇨ *Pierwsza praca, o której wspomniałem wyżej, nie dała mi technicznych umiejętności. Zapewniła za to tony frustracji, nudności o poranku i niechęć do życia. Naprawdę! To były chyba najgorsze miesiące mojego życia.*

Ale nic nie idzie na marne! W tym okropnym środowisku poznałem świetnych ludzi. Z niektórymi z nich kontakt utrzymuję do dziś! Z dwoma nagrałem odcinki podcastu. W przyszłości może zrobimy razem jakiś biznes? Od tamtej pory minęło kilkanaście lat i okropne doświadczenia skończyły się na zawsze. Zostały natomiast korzyści, być może też na zawsze.

We wszystkim da się znaleźć dobre strony. Jeśli masz pół szklanki wody, nie musisz marnować czasu na decydowanie, czy jest ona pełna, czy pusta. Możesz tej wody po prostu dolać.

To też powód, dla którego zachęcam do odłożenia wygórowanych oczekiwań finansowych na później. Przed wysłaniem pierwszego CV miałem już na koncie kilka „prawdziwych" projektów. Niektóre były zakończone sukcesem, inne to zdecydowane niewypały. Nieważne. W sumie nie zarobiłem na nich prawie nic, ale to też nieważne.

Ważne jest to, że prawie od początku swojej drogi gromadziłem nie tylko teorię, ale również najcenniejsze doświadczenie praktyczne. I jakże istotną umiejętność pracy w zespole.

Solo

Naukę jednak często zaczyna się w pojedynkę. Co wtedy? Nie trzeba załamywać rąk; zdobycie „prawdziwego" doświadczenia również na tym etapie jest jak najbardziej możliwe!

Wspólnym elementem wszystkich opisanych wyżej sytuacji jest pójście o krok (albo nawet o kilka kroków) dalej niż samo programowanie. Kluczowe jest uruchomienie stworzonego kodu. Udostępnienie go do wykorzystania przez prawdziwych użytkowników. Samodzielnie także można osiągnąć ten etap!

To w zasadzie proste. Wystarczy swój hobbystyczny projekt budowany w trakcie nauki (bo tworzysz taki, prawda?) doprowadzić do stanu „używalności". Upublicznić. Uruchomić w internecie lub udostępnić do ściągnięcia. Wrzucić na serwer, wypuścić do sklepu. I pochwalić się tym na Fejsiku.

Jest jeszcze inna droga: znaleźć interesujący projekt *open source* i dołączyć do niego! Zacząć współtworzyć jakąś bibliotekę z grupą zapaleńców z całego świata. Poprawić błąd, dodać funkcję... cokolwiek! A gdzie ich szukać? Oczywiście na GitHubie! GitHub to wyjątkowe miejsce spotkań programistów. Część internetu zarezerwowana dla nas. Tam żyje nasz kod, tam jesteśmy w domu.

Wtedy zobaczysz, co oznacza „zawód: programista". Zrozumiesz, w co się pakujesz. Pochwalisz się: „mam doświadczenie" i nikt Ci nie powie, że to nieprawda.

Jak szukać pracy jako początkujący programista?

Znowu ta sama mantra: najlepszym sposobem na prawdziwą naukę programowania jest znalezienie pierwszej pracy. Jak najszybciej. To jest już jasne, prawda? Powtarzam się, ale to dlatego, że nie wiem, od którego rozdziału zaczniesz czytać.

Masz gotowość, chęci. Masz piękne CV. Wiesz, na czym polega rekrutacja. I co dalej?

Nie wyślesz – nie znajdziesz

Praca nie znajdzie Cię sama; trzeba się trochę natrudzić. Na tym etapie nie możesz zgadywać, czego oczekuje firma. Jeśli zabunkrujesz się w postawie: „nie wyślę, bo i tak mnie odrzucą", nigdy nie znajdziesz pracy. Bo nigdy nie wyślesz. Koło się zamyka. Inicjatywa musi być po Twojej stronie! Nie wystarczy założyć konta na LinkedIn i czekać na grad ofert.

Kluczem do sukcesu jest wysyłanie, wysyłanie i jeszcze raz wysyłanie. A następnie rozmowy, rozmowy i jeszcze raz rozmowy.

Nie zrażaj się brakiem odpowiedzi. Nie zrażaj się odmowami. To normalne!

Po prostu próbuj dalej, systematycznie, do skutku. A jednocześnie nieustannie podnoś swoje kwalifikacje.

Wysyłaj CV do firm, które szukają kogoś takiego jak Ty. Wysyłaj do firm, które szukają kogoś innego. Wysyłaj nawet, jeśli akurat nie rekrutują. Niech Twój region dowie się, że szukasz pracy! Może akurat się spodobasz? Może mieli otwierać rekrutację w przyszłym tygodniu i z nieba im spadasz? A może przyda im się praktykant czy praktykantka do najbliższego projektu?

Co najgorszego może się stać?

Brak odpowiedzi. Ojej! Absolutnie nic!

Nikt mi nie odpowiada!

To częsty problem: wysyłasz, wysyłasz – i nic. Zero odzewu. Co wtedy? Po pierwsze: nie poddawaj się. A po drugie:

Dowiedz się, co jest nie tak.

Ty także na pewno nie odpisujesz na wszystkie maile, jakie do Ciebie przychodzą. Hrabia z Abu Dhabi proponuje Ci mailowo czterdzieści dziewięć milionów dolarów, a Ty ich nie chcesz i hrabia nie wie, dlaczego. Tylko że jego życie nie zależy od tego, czy przyjmiesz tę ofertę, podczas gdy przyjęcie przez firmę oferty złożonej przez Ciebie zmieni Twoją przyszłość. Musisz się dowiedzieć, dlaczego firmy Ci nie odpowiadają! Jak? Zapytaj!

Ktoś nie odpisał? To napisz ponownie z pytaniem: dlaczego? Mija kolejny tydzień i dalej nic? Napisz znowu, ale tym razem w inny dzień tygodnia. Serio! Masz dość zabawy w kotka i myszkę? Chwyć za telefon i zadzwoń!

Jeżeli problem braku odzewu występuje u Ciebie często, musisz interweniować. Nikt tego za Ciebie nie zrobi, a Ty się nie poprawisz, dopóki nie dowiesz się, co wymaga poprawy.

> **Insanity: doing the same thing over and over again and expecting different results.**
> ***Albert Einstein***

To Twój obowiązek, bo to Twoja przyszłość. Jeśli Twoje działanie nie przynosi rezultatów, czas je zmienić.

Stwórz sobie plik z informacjami o firmach, do których aplikujesz. Zamieść tam nazwę każdej z nich, e-mail, telefon, link do oferty, datę wysłania zgłoszenia, uzyskaną odpowiedź oraz – najważniejsza kolumna, która nigdy nie powinna zostać pusta! – powód odmowy.

A osobiście?

Firmy mogą dostawać dziesiątki CV dziennie, szczególnie na stanowiska juniorskie. I oczywiście, trzeba wysyłać – tak jak wszyscy, zgodnie z procedurą. Ale możesz zrobić coś więcej: wydrukuj kilkadziesiąt egzemplarzy swojego CV i ruszaj w miasto!

Zajdź do jednej, drugiej, dziesiątej firmy. Poproś o spotkanie z kimś od HR. Nie mają czasu? To zostaw swoje CV i poproś o kontakt. A po jakimś czasie koniecznie zrób follow-up. Przypomnij się.

Zapamiętają Cię. Nawet jeśli w danej chwili nikogo nie szukają, to zostaniesz „na radarze", bo prawie nikt tak nie robi. A to – pokazanie zaangażowania i niestandardowe działanie w sensowny sposób – bardzo wiele znaczy.

Społeczność na ratunek

W niniejszej książce wiele razy przeczytasz o społecznościach. Lubimy się spotykać, być razem. Integrować się, gadać, opowiadać o pracy. Znajdź taką społeczność w swoim mieście. Zacznij od odwiedzenia strony http://zawodprogramista.pl/spolecznosci.

Pójdź na spotkanie, przedstaw się. Poznaj ludzi. Daj znać, że szukasz pracy. Efekty Cię zaskoczą. To może być skuteczniejsze niż rozesłanie kilkunastu CV,

bo teraz działasz od środka. Pomijasz całą filtrację i trafiasz bezpośrednio do członków zespołu, z którymi być może będziesz pracować.

A jeśli zgłosisz się z własnym wystąpieniem i na scenie poopowiadasz o sobie i swoich poszukiwaniach, to ofertę pracy możesz dostać nawet tego samego wieczora!

⇨ *Jestem współorganizatorem spotkań Białostockiej Grupy .NET: #bstoknet. Po prawie każdym spotkaniu na miejscu zostaje mniejsza grupka osób, wymieniająca się doświadczeniami z pracy i opowieściami z biurowej codzienności. Takie „afterparty" trwa czasami nawet kilka godzin. To normalne, i inne meetupy wyglądają podobnie.*

Może w tak sprzyjających okolicznościach natkniesz się na seniora, osobę odpowiedzialną za prowadzenie rekrutacji i poszukującą nowych twarzy do swojego zespołu? Nieoficjalna, przyjazna atmosfera sprzyja zawieraniu nowych znajomości. O wiele łatwiej będzie Ci przebrnąć przez kolejne etapy rekrutacji, kojarząc się jako „ta znajoma twarz z meetupu".

⏎

Kiedy junior developer jest gotów do pracy?

Wielu prawie-junior-developerów siedzi smutno w domu i zastanawia się: „czy już jest mój czas, czy mogę podjąć pracę?".

Mam dla Ciebie olśniewającą wiadomość:

Decyzja nie należy do Ciebie!

Nie Ty musisz odpowiedzieć na to pytanie. Na to pytanie odpowiada rynek. Pracodawcy. Firmy poszukujące junior developerów.

Nie możesz wiedzieć, że jeszcze dla Ciebie za wcześnie, dopóki jakaś firma nie odrzuci Twojej kandydatury. Jednocześnie nie dowiesz się, że umiesz już wystarczająco dużo, dopóki jakaś firma nie przyjmie Cię do pracy. To przecież logiczne, prawda? A mimo to tak wiele osób spędza całe miesiące na dywagacjach nad tą kwestią.

Starasz się o pracę, ale jej nie dostajesz? Poucz się jeszcze i spróbuj ponownie! Starasz się o pracę i znajdujesz zatrudnienie? Brawo, *mission accomplished!*

Jedyną alternatywą jest niestaranie się o pracę.

Nie bój się spróbować. Jeżeli w głowie pojawiają się takie dylematy, to oznacza, że już czas. Musisz zaryzykować i dać sobie szansę. Tym bardziej, że nie ryzykujesz niczym! W najgorszym wypadku zaaplikujesz i nie dostaniesz pracy. Będziesz więc dokładnie w tym samym miejscu, w którym jesteś teraz, ale z dodatkowym doświadczeniem.

Nieprzyjęcie do pracy to normalna sprawa, szczególnie na początku kariery. To nie powód do wstydu, lamentów i załamywania rąk. To motywacja do dalszej nauki. Każde odrzucone CV jest krokiem w odpowiednim kierunku.

Nie skreślaj siebie z rynku za wcześnie. Może akurat firma poszukuje kogoś dokładnie z Twoim zestawem umiejętności? Może to właśnie ta rekrutacja, której tak się obawiasz, prowadzi do wymarzonego projektu w świetnym zespole? Nie pozbawiaj się szansy na rozpoczęcie kariery we właściwym momencie. I nie pozbawiaj firmy szansy na pozyskanie wartościowego członka zespołu – Ciebie.

Przyszło Ci kiedyś do głowy, żeby spojrzeć na to w ten sposób?

Odpowiedzialność

Nie obawiaj się też, że nie podołasz stawianym przed Tobą zadaniom. W momencie podpisywania umowy to firma bierze na siebie odpowiedzialność: „zweryfikowaliśmy tego człowieka i właśnie kogoś takiego szukamy". Nawet jeżeli po przyjęciu do pracy okaże się, że nie do końca pasujesz na swoje stanowisko i przerasta Cię zakres obowiązków, to nie jest to Twoja wina. O ile rekrutacja

odbyła się rzetelnie i szczerze (o czym jeszcze duuużo będziemy mówić w kolejnych rozdziałach), firma nie powinna wymagać od Ciebie więcej, niż jesteś w stanie z siebie dać. W przeciwnym wypadku zawiodły: procedura rekrutacji i osoby za nią odpowiedzialne. Wtedy musisz odbyć szczerą rozmowę z przełożonym i wspólnie postarać się znaleźć wyjście z tej sytuacji.

Powtórzę: pod warunkiem, że w rekrutacji nie brała udziału sztuczna, napompowana nierealnymi umiejętnościami i fałszywymi cechami wydmuszka w Twojej skórze. Widzisz, dlaczego to takie ekstremalnie ważne?

Jak napisać programistyczne CV?

Bez CV nie dostaniesz pracy. A CV to takie „ja w pigułce". Trzeba tam zawrzeć wszystko, co masz na swój temat do przekazania. Najlepiej maksymalnie na dwóch stronach (dlaczego? o tym zaraz).

Przez lata widziałem setki CV. Dziesiątki z nich recenzowałem i pomagałem ulepszać. Moje rady się sprawdzały. Po konsultacjach otrzymywałem podziękowania, bo dzięki nim „CV lądowało na górze stosu".

Po lekturze książki odwiedź stronę http://zawodprogramista.pl/kariera, gdzie znajdziesz jeszcze więcej materiałów na ten temat!

Wszystkie moje rady udzielone na przestrzeni lat można sprowadzić do kilku wspólnych punktów.

Kim jesteś?

Zacznij od imienia i nazwiska na samej górze. Nie od kwiatuszków, chmurek i kolorowych wygibasów. Nie wygłupiaj się; nie aplikujesz na stanowisko grafika czy designera. Kreatywność jest istotna w każdej pracy, ale takie zabiegi skutecznie redukują czytelność Twojego CV.

W programistycznym CV – tak jak w kodzie! – najważniejsza jest właśnie czytelność. Trzymaj fantazję krótko. Nie szalej.

W bardzo wielu CV brakuje jednego istotnego elementu: kto to? Pod imieniem i nazwiskiem napisz:

- **dwa lub trzy słowa podsumowania: kim chcesz być?**

Na przykład: „Programista PHP", „Front-end developer", „Programista i architekt .NET". Niekoniecznie, kim jesteś teraz. Wyraź swoją wizję. Po co piszesz to CV? Dokąd Cię ono zaprowadzi? Ja bym nie dodawał „młodszy", „starszy" – definicje tych pojęć, jak już wiemy, są różne w różnych firmach.

Ale jak to: technologia? Przecież trzeba być programistycznym poliglotą! Nie można przywiązywać się do jednego języka! Niby tak, ale… aplikujesz na konkretne stanowisko, prawda? Szukasz pracy w tej technologii, a nie dowolnej, byle jakiej. Dlatego na tym poziomie zaszufladkowanie jest w porządku. Bo poliglota poliglotą, ale na coś musisz się – zawodowo, tu i teraz – zdecydować!

Z biegiem czasu, w miarę ewolucji CV, ta linijka może stawać się coraz bardziej ogólna.

Dane kontaktowe

Standardowo: imię, nazwisko, adres, numer telefonu itd.

Często elementem wyróżniającym CV – w negatywny sposób – jest adres e-mail. Tutaj zdecydowanie radzę powstrzymanie się od „ciekawych" pomysłów; „kwiatuszek83@wp.pl", „asia200@kozaczek.pl", „procent@onet.pl" czy „wckaczek@hotmail.com" to nie są adresy do komunikacji biznesowej. Dodatkowo pamiętaj: z jakiegoś – nieważne jakiego – powodu gmail.com wygląda poważniej niż cała reszta darmowych alternatyw.

Omijałbym wszelkie wariacje poza imieniem i nazwiskiem. Starsze pokolenia mają prosto: imię.nazwisko@gmail.com – i już! Dziś jednak możesz nie mieć tego szczęścia: wszystko jest już pozajmowane. Ja swojej córeczce zarezerwowałem adres z gmail.com, jeszcze zanim się urodziła – kiedyś mi podziękuje.

Nie ma co się zżymać i ozdabiać adresu dodatkowymi liczbami czy podkreśleniami. Istnieje o wiele lepsze rozwiązanie. Własna domena!

Masz sporo możliwości: kontakt@imięnazwisko.pl, imię@nazwisko.pl. Możesz też wziąć końcówkę .com albo .me. To sprawa wizerunku, jak pójście do fryzjera przed rozmową kwalifikacyjną. A i koszt podobnie niski.

Wiek i zdjęcie a dyskryminacja

Trwa dyskusja: czy umieszczać w CV swoje zdjęcie oraz datę urodzenia? Przecież na tej podstawie można paść ofiarą dyskryminacji!

Na Zachodzie te informacje w CV nie są mile widziane. Tam wszyscy są bardzo uważni, czuli i przewrażliwieni. Ale my rozmawiamy o brutalnej, gruboskórnej jeszcze i momentami ciemnej Polsce. Tutaj masz dwie drogi: walkę z wiatrakami albo dostosowanie się do panujących reguł.

Pierwsza droga może być szlachetna. Don Kichot kasuje z CV zdjęcie i wiek, akceptując tym samym zmniejszenie swoich szans na rynku pracy. Może praca nie jest dla niego priorytetem.

Bardzo dobrze rozumiem rekruterów lubiących otrzymać pełen zestaw informacji o kandydacie czy kandydatce, włączając w to zdjęcie i wiek. Sam byłem na tym miejscu i – dyskryminacja czy nie – chciałem zdobyć jak najwięcej informacji o osobie starającej się o pracę.

Oczywiście to, że ktoś jest „brzydki", nie powinno mieć wpływu na ewentualne zatrudnienie. Ale zastanów się: jeżeli rekruter ma Cię na tej podstawie odrzucić, to i tak to zrobi po pierwszym spotkaniu! Chowanie zdjęcia w CV niczego nie zmieni.

A wiek? Bardzo istotne jest dla mnie, czy zatrudniam dwudziesto-, czy pięćdziesięciolatka. I tak jak ze zdjęciem: już podczas pierwszej rozmowy – nawet telefonicznej – wyjdzie to na jaw.

Nie ma więc o co kruszyć kopii. Zaznaczam, że nie usprawiedliwiam dyskryminacji w żadnej jej odsłonie, a ten rozdział wyglądałby inaczej, gdybym pisał książkę dla rekruterów. Ale piszę ją dla Ciebie – programisty i programistki.

Bądźmy pragmatyczni i dążmy do własnych celów, a ideologiczne wojenki uprawiajmy dopiero po ich osiągnięciu.

> **Czy kwestia zdjęcia w CV**
> **jest Twoim najważniejszym problemem w życiu?**

Nie utrudniaj sobie życia. Zachowaj siły na ważniejsze sprawy.

Doświadczenie

Pod danymi osobowymi przychodzi czas na doświadczenie.

Wymień firmy ze swojej przeszłości. Dodaj datę rozpoczęcia i datę zakończenia pracy w każdym z tych miejsc (wystarczy miesiąc i rok). Dopisz stanowisko oraz kilka słów o swoich obowiązkach.

Nie wchodź w zbytnie szczegóły, ale też nie zadowalaj się takimi ogólnikami, jak „stanowisko: programista; obowiązki: programowanie". W końcu wiadomo, że programujesz jako programista! Wymień używane technologie i główny zakres swojej odpowiedzialności.

Unikaj zamieszczania nieistotnych informacji. Praca przy malowaniu balkonów, na kasie lub jako ogrodnik na cmentarzu nijak się ma do naszej profesji. Nikogo to nie obchodzi. Nie masz doświadczenia zawodowego? To nie podawaj go na siłę! Każdy kiedyś zaczynał i szukał pierwszej pracy.

Zrealizowane projekty

To bardzo ważna sekcja.

Zamieszczenie w CV samego linka do GitHuba to za mało. Ten dokument ma być samowystarczalny! Bez dostępu do internetu, nawet po wydrukowaniu.

Dodaj tutaj informacje o przedsięwzięciach z Twoim udziałem. Napisz kilka słów wyjaśnienia: przeznaczenie, wykorzystane technologie, Twoja rola w projekcie. Idealnie, jeśli kod projektu jest w internecie ogólnodostępny. Ale jeśli nie, to i tak coś powinno się w tym miejscu znaleźć. To nie muszą być wdrożone, komercyjne projekty.

Tej sekcji u większości początkujących osób nie ma. A to bardzo niedobrze – dla nich! Dla Ciebie oznacza to, że możesz łatwo wyróżnić się na tle konkurencji.

Skąd te projekty wziąć? Chwila, moment – zaraz do tego wrócimy.

Wykształcenie

Wymień uczelnię, miasto, w którym się znajduje, wydział, kierunek, rok rozpoczęcia. Jeśli masz to już za sobą, podaj datę zakończenia i tytuł pracy dyplomowej. W przypadku wykształcenia w innej dziedzinie tytuł pracy można pominąć.

Jestem zwolennikiem umieszczania wykształcenia aż do szkoły średniej. Jest to jednak pierwsza informacja do skasowania, gdy zaczyna brakować miejsca i przestajesz się mieścić na dwóch stronach.

Umiejętności

Najważniejsze mięsko. To część, która będzie najbardziej interesowała osobę przeglądającą nasz dokument, dlatego o ten element trzeba zadbać wyjątkowo.

Warto podzielić je na odpowiednie kategorie:

- Technologie, czyli informacje o językach (i ewentualnie frameworkach).
- Narzędzia: kontrola wersji, IDE… (o ile mamy na to miejsce).
- Praktyki: testy jednostkowe, wzorce, metodyki (scrum). Konkret, bez wodolejstwa.

Można pokusić się o „Inne", ale tam znajdzie się zwykle tylko prawo jazdy, niewymagane przecież do pracy programisty. Ze swojego CV wyrzuciłem więc „Inne"; szkoda miejsca.

Trzeba uważać, żeby tej części CV zbytnio sztucznie nie napompować. Łatwo się zagalopować i wpisać wszystkie języki programowania omawiane na studiach. To zła droga. CV przepełnione pustymi słowami kluczowymi nie zrobi na nikim wrażenia, a może zadziałać wręcz odpychająco. Zamieść to, co faktycznie umiesz. Nawet jeśli miałby to być tylko jeden język. Ważne, żeby to był ten właściwy, potrzebny.

I uwaga: zamieść to, co faktycznie chcesz robić! Chcesz programować? To nie wpisuj umiejętności administrowania systemem Linux. Albo obsługi Office'a czy znajomości GIMP-a.

Stopień zaawansowania umiejętności to ciężki temat. Często widzę CV z wymienionymi dziesięcioma językami programowania. Przy każdym z nich – ocena w skali od jeden do pięciu. Jeden język ma ocenę 5/5, kilka – 3/5, pozostałe – 1/5. Co oznacza 1/5? Że danej umiejętności nie posiadasz! Więc po co wpisujesz to do CV? To dokument z aktualnymi – a nie planowanymi – umiejętnościami.

Jeśli z kolei dajesz sobie ocenę maksymalną, to przygotuj się na srogi sprawdzian. Wraz z doświadczeniem przychodzi nowa perspektywa. Zaczynasz bardziej zauważać swoje braki, mniej skupiając się na zdobytej już wiedzy. Z tego powodu sam nie postawiłbym sobie maksymalnej oceny nawet przy znajomości języka polskiego czy umiejętności oddychania.

Jak się więc ocenić? Najlepiej wcale.

Wystawienie oceny zostaw osobie weryfikującej Twoje przygotowanie.

Języki obce

Można o językach napisać w „Umiejętnościach", a można je wyodrębnić do oddzielnej sekcji. Ważne, by mieć tutaj coś do wpisania. Pamiętasz rozdział o najważniejszym języku dla programisty? No właśnie...

Podobnie jak w przypadku narzędzi – ostrożnie z oceną swojego poziomu. Jeśli wystawiasz sobie maksymalną ocenę w skali pięciostopniowej, to będę oczekiwał, że rozmawiasz na poziomie dobrze wykształconego native'a. A tak przecież nie jest, prawda? Nie stosuj skali liczbowej. Taka skala nie jest obiektywna.

Możesz opisać swój poziom słownie, szczególnie jeśli nie masz żadnych certyfikatów. Zwykle wystarczy wpisanie jednej z formułek: „podstawowy", „wystarczający do czytania dokumentacji ze zrozumieniem", „komunikatywny", „biegły". Z uzupełnieniem, czy tylko „w piśmie", czy także „w mowie". Są to kolejne poziomy wtajemniczenia potrzebne w zawodzie programisty.

U mnie angielski jest biegły w mowie i piśmie. A niemiecki – podstawowy, bo ogranicza się do rozumienia części tekstów Rammsteina.

Co bardzo ważne: pisz tylko i wyłącznie prawdę. I przygotuj się na weryfikację tych umiejętności w procesie rekrutacji.

Osiągnięcia i społeczność

Ta sekcja w większości CV jest całkowicie pusta i na zawsze taka pozostaje. A tymczasem jest to doskonała okazja, by się wyróżnić z tłumu!

Nawet najmniejsza rzecz, niewielka aktywność, zaangażowanie w jakiejś dziedzinie może przeważyć na Twoją korzyść. Bierzesz udział w konkursie? Występujesz na konferencjach? Prowadzisz bloga? Masz doświadczenie w przygotowywaniu szkoleń? Piszesz książkę?

Wpisuj to!

Masz ukończone szkolenia? Albo nawet zdane certyfikaty? Organizujesz spotkania koła naukowego albo innej grupy pasjonackiej?

Wpisuj!

Rozważ udzielanie się na forach, rozwiązując problemy innych programistów. Zaczniesz wyrabiać swoją markę, pochwalisz się tym w CV, a dodatkowo bardzo poszerzysz swoją wiedzę! Listę różnych społeczności związanych z programowaniem znajdziesz na stronie http://zawodprogramista.pl/spolecznosci.

Wystarczy chcieć!

Zainteresowania

Z Twoimi zainteresowaniami w CV to jest tak, że... nikogo innego one nie interesują. Serio. Szczególnie, jeśli zamierzasz tam wpisać „film akcji", „dobrą książkę" i „programowanie". Ja jestem smutasem i nie mam ciekawych zainteresowań poza IT, więc nie silę się tutaj na kreatywność. Bo i po co?

Jeśli Twoje hobby Cię wyróżnia, to wtedy możesz je zamieścić. Podróżujesz w oryginalne miejsca? Wpisz to! (Chyba że chodzi tylko o kurorty odwiedzane przez miliard ludzi rocznie). Piszesz blog parentingowy? Wpisz! Ścigasz się

w rajdach? Masz mistrzostwo w krav madze? Hodujesz piranie? Wpisuj! Będzie punkt zaczepienia podczas rozmowy.

Ale „siłownia, piłka nożna i rozwój osobisty" to marnowanie cennego miejsca na tych dwóch stronach w CV.

Limit dwóch stron

No właśnie, limit dwóch stron... Spotkasz różne opinie. Ja radzę trzymać się tej rekomendacji z kilku powodów.

Po pierwsze: szanuj czas odbiorcy tego dokumentu. Jak sądzisz, czy rekruter przeglądający setkę CV dziennie poświęci pół godziny na lekturę Twoich dwudziestu stron? No przecież nie!

I po drugie: jeżeli dwie pełne strony to za mało, by Cię przedstawić, to gratuluję niesamowitych doświadczeń! A może po prostu jest tam zbyt wiele bezwartościowych informacji?

> **Po osiągnięciu dwóch stron CV jest kompletne.**

Dodanie nowej informacji musi wiązać się z usunięciem czegoś mało istotnego. Wiesz, ten certyfikat zdobyty piętnaście lat temu... Czy on naprawdę ma teraz znaczenie? Albo olimpiada matematyczna wygrana w ubiegłym stuleciu... Ona na nikim nie zrobi już wrażenia. Szkoła średnia w takim przypadku także wylatuje.

Takie ograniczenie sprawi, że Twoje CV będzie aktualne i wartościowe. Nie zamieni się w śmietnik i całą biografię. Chcesz napisać coś więcej? Lepiej się przedstawić? Załóż blog albo stronę domową i podaj do niej link.

Jeśli dwie pełne strony nie wystarczą do zainteresowania kogoś Twoją osobą, to kolejne dwadzieścia stron nie pomoże.

Klauzula

Czasem zgłaszają się do mnie osoby z pytaniem: dlaczego nikt nie odpisuje na moje aplikacje?

Spoglądam na CV, a tam brakuje wymaganej klauzuli ze zgodą na przetwarzanie danych osobowych.

Bez tej klauzuli nie spodziewaj się odpowiedzi. Nie wyrażasz na nią zgody! Pamiętaj, że często dane z CV są wprowadzane do odpowiedniego systemu i dopiero potem przetwarzane. Bez Twojej zgody na ten proces CV nie znajdzie się w systemie.

Aktualną treść wymaganej klauzuli bez problemu znajdziesz w internecie.

I najważniejsze – po raz kolejny

Wszystkie informacje zawarte w CV muszą być absolutnie prawdziwe. Nigdy dość przypominania o tym! Żadnej ściemy! Żadnego „jak będą to sprawdzać, to najwyżej się douczę".

Znajomość języka programowania łatwo zweryfikować. Jeszcze łatwiej – znajomość języków obcych. Podobnie obsługę narzędzi.

U mnie osobiście ktoś, kto ściemnia w CV, nie miałby szans na stanowisko.

Kłamiesz? Do widzenia! Nie zasługujesz na moje zaufanie.

Już na samym początku znajomości zaczynasz coś kręcić? To okłamiesz mnie także w przyszłości.

Zarządzanie systemem Linux – co przez to rozumiesz? Czy na ewentualnej rozmowie kwalifikacyjnej poradzisz sobie z zadaniem z tego zakresu?

Obsługa pakietu Office: faktycznie w minutę wyklikasz jakieś wygibasy w Excelu? Czy po prostu uruchomisz Worda?

Znajomość języka Scala: napiszesz choćby *hello world* bez pomocy z zewnątrz? Czy tylko skopiujesz kodzik ze Stack Overflow?

Druga strona może zechcieć najzwyczajniej w świecie sprawdzić Twoją prawdomówność i zdolność oceny własnych umiejętności. I słusznie.

Jak wypełnić CV doświadczeniem?

Każdy kiedyś zaczynał. Każdy kiedyś nie miał doświadczenia komercyjnego i przed zdobyciem pierwszej pracy ta sekcja w CV świeciła pustkami. Ale „zrealizowane projekty" to zupełnie inna bajka!

Szukasz pracy? A co potrafisz? Programować w języku X... A jak można to zweryfikować bez potencjalnego marnowania czasu na godzinne interview?

Udowodnienie swoich umiejętności już na etapie CV to Twój obowiązek.

Zaczynasz czuć stresik? Element paniki: „ojej, nie mam co tam wpisać"? To dobrze. Refleksje na ten temat to bardzo pozytywny znak. Znak, który mówi: nadszedł Twój czas! Zakasaj rękawy i do roboty.

Każdy zaczynał z pustym CV

Spokojnie, nie ma się czego bać. Każdy miał kiedyś w CV wpisane tylko liceum i uczelnię. To nie powód do rozpaczy, ale... im wcześniej zaczniesz dbać o rozwój swojego résumé, tym lepiej dla Ciebie.

Pracodawcy poszukujący osób na staż albo stanowisko „junior dev" nie oczekują zbyt wiele. Wiedzą, że przychodzący człowiek będzie wymagał nauki. Inwestycji. Czasami: prowadzenia za rączkę. I bardzo dobrze, bo gdzieś trzeba zacząć. Jesteś w miejscu absolutnie zgodnym z oczekiwaniami rynku.

Jednak to nie powód, by nic nie robić. A dlaczego?

> **Takich jak Ty są tysiące!**

A potrzeba naprawdę niewiele – bardzo niewiele – aby się wyróżnić z tłumu.

Co możesz zrobić już dziś?

Dzięki kilku opisanym niżej zabiegom Twoje CV już od jutra będzie lądowało na górze stosu.

Wspomnieliśmy o doświadczeniu i zrealizowanych projektach. Czas te projekty uzupełnić! Ale co tam wpisać, jeśli nie masz żadnych realizacji?

Błąd! Masz te projekty, ale może nawet nie zdajesz sobie z tego sprawy. Więc po kolei.

Po pierwsze – zbierasz...

...i przeglądasz cały kod, jaki do tej pory wypluła Twoja klawiatura. Serio: cały kod. Z całego życia.

Wszystkie te nieudane eksperymenty z kolejnymi bibliotekami JavaScript. Zakończone łkaniem brutalne współżycia z wersjami alfa. Nierówna walka z CSS-em. Pierwsza stronka łącząca się z bazką.

Wreszcie: potworki stworzone na zaliczenie przedmiotów. Wszystkie projekty zrealizowane na potrzeby studiów. Standardowy „System bankowy" w Javie? Albo „Obsługa pizzerii" w C#? A może „Gabinet weterynaryjny" w Pythonie? Może jakaś inżynierka czy magisterka?

Wszystko!

W moim przypadku w tym miejscu znalazłyby się różne zabawy. A to implementacja stosu, a to aplikacja ściągająca rozkład jazdy autobusów, wygaszacze ekranu, klient Gadu-Gadu, strona o wampirach...

Powtórzę jeszcze raz: wszystko!

Potem odrzucasz połowę najgorszego, najbrudniejszego kodu. Tego, którego się najbardziej wstydzisz (spoko, każdy ma na swoim koncie takie bagno).

A co z resztą?

Po drugie – katalogujesz, opisujesz...

...i wrzucasz do CV! Tak jest, wypełniasz nimi sekcję „Zrealizowane projekty".

Wiele osób błędnie uważa, że w tym miejscu powinny znaleźć się tylko zamówienia komercyjne. Ależ skąd! Na pewno lepiej jest tam umieścić projekt ze studiów niż nic!

Dopisujesz więc tych kilka lub kilkanaście najmniej brzydkich projekcików. O każdym piszesz kilka słodko-gorzkich słów. Odpowiadasz na proste pytania: co ten projekt robi? czego mnie nauczył?

Nagle okazuje się, że masz z pół strony (albo i więcej?) doświadczenia! Wow! Pokazujesz, czym się bawisz, co robisz. To o wiele lepsze niż cały akapit słów kluczowych, pustych nazw technologii, o których wiesz tyle, że „było kiedyś na jednym wykładzie".

Jeszcze raz podkreślę: to nie musi być kod piękny, kod idealny. Ważne, żeby był. Przecież wiadomo, że się uczysz. W tym przypadku naprawdę lepszy rydz niż nic.

Po trzecie – GitHub!

Konto na GitHubie trzeba mieć – i koniec. Jeśli więc jeszcze go nie masz, załóż od razu. Tylko nie wybieraj nazwy użytkownika typu „uroczy_bobasek_98". Polecam „imięnazwisko" albo nazwisko z pierwszą literą imienia.

To od tej pory Twój dom. Twoja wizytówka. Film dokumentalny o Twojej podróży do świata wielkiego IT.

Napisz krótkie bio: dwa, trzy zgrabne zdania o sobie. Po angielsku. Wgraj zdjęcie. I nie porównuj swojego konta z kontami innych! Nie wszystkie są „ładne". Wiele osób traktuje GH tylko jako backup kodu. Często te osoby nie muszą się już starać – w przeciwieństwie do Ciebie!

Po czwarte – online

Na koniec uczysz się podstaw Gita i wysyłasz swoje projekty na GitHub. Omówienie tego procesu zdecydowanie nie zmieści się w zakresie tej książki, ale nie zajmie Ci to dłużej niż godzinę. Bo to proste.

Po wysłaniu projektów dołączasz linki do nich do swojego CV.

Zrobione! Brawo, masz już zrealizowane projekty.

Obawy

Na pewno masz jednak obawy. „To przecież taki słaby kod, wyśmieją mnie!".

Pytanie: kto Cię wyśmieje? Firma, do której aplikujesz? W takim razie to i tak nie jest miejsce dla Ciebie. Ale nie bój się; tak się nie stanie.

Nawet słaby kod to świetny punkt wyjścia do dyskusji. Dzięki temu będziesz mieć okazję, aby poopowiadać, co więcej wiesz o danej technologii. Jakie błędy w projekcie Ci się przydarzyły? Jak zrobisz to lepiej w przyszłości, czego Cię to nauczyło?

Ważny jest kontekst. W tym przypadku to Ty go budujesz!

Na GitHubie każdy projekt może mieć opis wyświetlany na stronie (plik readme.md). Dodaj taki opis. Napisz, kiedy powstał kod, czego cię nauczył i co ma udowodnić. Dzięki temu odwiedzający pokiwa głową i szepnie: „ooo, dobrze kombinuje! nie wstydzi się swojej pracy!".

Popatrz po swoim otoczeniu. Ile osób pokazuje kod światu?

Konsekwencje

A co jeszcze wyniknie z takiej praktyki? Samo dobro! Jak zmieni się Twój sposób pracy, jeśli z założenia cały kod będziesz traktować jako publiczny i wysyłać w świat? Prawdopodobnie bardziej się do niego przyłożysz, a co za tym idzie: Twoje umiejętności wzrosną szybciej niż skille w *Diablo* pozostawionym na noc z przyblokowanym przyciskiem myszy.

Inne CV zawierają ładne słówka; w Twoim znajdą się też dowody. Potwierdzenie tego, że:

- **Umiesz Gita.** To narzędzie, które trzeba znać. Kropka. Niektórzy umieją go bardzo dobrze, inni – średnio, a jeszcze inni – wcale. Napisanie w CV „znajomość Gita" nie jest nawet w jednym procencie tak dobre, jak dodanie linków do swoich repozytoriów.
- **Znasz GitHuba.** To jedno z dwóch najważniejszych miejsc w programistycznym internecie. Miejsce spotkań programistów i programistek z całego świata. Trzeba mieć tam konto.
- **Nie wstydzisz się swojej pracy.** Każdy popełnia błędy. Każdy na początku generuje tony śmieci. Ale nie każdy chwali się swoimi postępami, pokazując ewolucję, dokumentując proces. To piękna cecha!

- **Rozwijasz się.** Twój kod świadczy o tym, że – z lepszymi bądź gorszymi efektami, wolniej lub szybciej – idziesz do przodu. A nawet jeśli czasami nie do przodu, tylko w bok – to nieważne. Ważne, że nie stoisz w miejscu. Że działasz!
- **Świadomie oceniasz swoją pracę.** Pamiętasz plik *readme.md*? Budowanie kontekstu? Nie każdy będzie w stanie przyznać się do popełnionych błędów. Wytknąć sobie własne potknięcia i wskazać możliwe rozwiązania. A Ty pokazujesz, że potrafisz. To niesamowicie przydatna cecha w pracy zespołowej.

Pokazujesz, że już jesteś juniorem z górnej półki.

Ale ja nie mam swojego kodu!

Jeśli uczysz się programowania i naprawdę nie masz niczego do wrzucenia na GitHub, może to być najważniejszy akapit w Twojej dotychczasowej karierze. Bo... co właściwie robisz?

Programowania uczysz się, programując. Programujesz, pisząc kod.

Nie masz kodu = nie uczysz się.

Sorry, to brutalne, ale tak proste!

Jak spędzasz czas, który poświęcasz na naukę programowania? Dlaczego w tym czasie nie powstaje zawartość Twojego GitHuba?

Coś jest nie tak. Zajmij się tym, najlepiej od razu. Lepiej późno niż później.

Jak przygotować się do rozmowy kwalifikacyjnej?

Rekrutacja to poważna sprawa. Wiele może pójść nie tak. Ale na szczęście bardzo dużo zależy tylko i wyłącznie od Ciebie. I możesz się do tego przygotować!

Marnowanie czasu a przygotowania

Bardzo źle widziane jest marnowanie cudzego czasu. Marnowanie swojego czasu też nie jest najmądrzejsze, ale cudzego – wprost niewybaczalne. Rozmowa rekrutacyjna to umówione spotkanie biznesowe. Poważna sprawa. Spotkanie wymagające przygotowania. Bez tego prawie na pewno zmarnujesz sporo czasu po obu stronach stołu.

Przygotowanie do rozmowy rekrutacyjnej wbrew pozorom wcale nie jest trudne. A co najlepsze: robisz to tylko raz, a z efektów możesz korzystać wielokrotnie. Rozmowy mogą przebiegać różnie, ale prawie wszystkie będą miały elementy wspólne. Potrzebujesz tylko kilku minut, aby już na samym początku wywrzeć pozytywne wrażenie dzięki płynnym i pewnym odpowiedziom.

Jak sądzisz, jakie pytania padają prawie zawsze? Spróbuj wymyślić kilka, zanim przejdziesz dalej. Nawet tu masz dużo miejsca, żeby je wypisać:

Trzy.

Dwa.

Jeden.

Już?

Pamiętaj też, że przygotowanie to coś więcej niż „zastanowienie się w głowie przez trzydzieści sekund". Prawdziwe przygotowania wymagają zrobienia notatki i przećwiczenia odpowiedzi. Tak, przećwiczenia! Na głos, przed lustrem bądź zaufaną osobą. Dotyczy to każdego z poniższych tematów.

Nie chodzi bynajmniej o wyrycie na blachę i późniejsze recytowanie z pamięci, tylko o przetestowanie połączeń między zdaniami, zweryfikowanie ciągłości, uzyskanie płynności.

Ty

Jeżeli Twoje CV nie poszło do kosza i zaproszono Cię na rozmowę kwalifikacyjną, to czas na dalsze przedstawienie siebie.

Na rozmowę idziesz po to, aby dać się poznać. Bardzo ważne jest, aby na pytania „co sprawia ci największą satysfakcję?" czy „dlaczego jesteś lub chcesz być programistą?" mieć gotową odpowiedź. Może też paść lakoniczne: „opowiedz nam coś o sobie". I wtedy... no cóż, musisz opowiedzieć.

Czasami takie rzeczy wydają się oczywiste, ale przed rozmową zadaj sobie kilka takich pytań i zobacz, czy umiesz natychmiast na nie odpowiedzieć. „Yyyyy" i „eeee" nie sprawiają najlepszego wrażenia. Na jąkanie się jeszcze przyjdzie czas przy pytaniach technicznych. O sobie musisz umieć opowiedzieć w miarę płynnie!

Niech bycie bezpośrednim tematem rozmowy Cię nie krępuje. Niech nie sprawia, że poczujesz się niezręcznie. W końcu dokładnie po to tam jesteś! Uświadom sobie także, że osoba po drugiej stronie stołu prowadzi pewnie dziesiątki takich rozmów i nie musisz się wstydzić. To dla niej normalka.

Projekty

Na pewno padnie pytanie o Twoją przeszłość. „Opowiedz o swoich zrealizowanych projektach". „Jakie masz osiągnięcia?". „Co sprawiło Ci największą trudność?". O to po prostu nie wypada nie zapytać. I nie wypada nie znać na to odpowiedzi.

Rozmowa nie jest dobrym momentem na przypominanie sobie szczegółów dotyczących danych okresów w Twoim życiu. Dlatego wcześniej przejrzyj swoje CV. O każdej informacji tam zawartej opowiedz na głos, własnymi słowami.

Masz tam jakiś projekt? Skleć minutową opowiastkę o nim. Niech otrzyma od Ciebie trochę emocji i stanie się częścią Twojej historii. Po co powstawał? Jak wyglądała organizacja zespołu? Jakie narzędzia pomogły, a jakie się nie sprawdziły? Co miło wspominasz z czasów prac nad tym projektem, a co przyprawia Cię o dreszcze? Jakie problemy pojawiły się po drodze? Czy ktoś tego używa?

Co dało Ci stworzenie tego rozwiązania w dłuższej perspektywie? Może inne spojrzenie na jakiś framework, może nowe doświadczenie w pracy grupowej, może eksperyment z takim, a nie innym stylem kodowania?

Cokolwiek! Żaden projekt nie powinien być suchym, niemym, anonimowym wpisem w CV.

Oczekiwania

Pracodawca wie, czego oczekuje od Ciebie: Twojego czasu, umiejętności i zaangażowania. Jasna sprawa.

A Ty? Czego oczekujesz od firmy i projektów? Nie chodzi tylko o finanse. Zastanów się. Następnie powiedz o tym – jasno i klarownie. Rozmówca musi dokładnie zrozumieć, co jest dla Ciebie niezbędne do zawodowego zadowolenia, tak aby potem przekazać to dalej.

U mnie osobiście rzeczą absolutnie najważniejszą – od zawsze, od samego początku – był całkowicie uregulowany czas pracy. Na każdej rozmowie bardzo wyraźnie dawałem do zrozumienia, że nie akceptuję nadgodzin. Umawiamy się na X godzin – i koniec. Bo poza pracą też mam życie – nawet jeśli przez długi czas główną aktywnością w tym pozapracowym życiu i tak było programowanie.

Dzięki temu nadgodzin doświadczyłem może z pięć razy podczas dziesięciu lat pracy. Da się tak! Tylko trzeba zrobić z tego priorytet.

Przyszłość

Jest pewne pytanie, które może sprawić Ci nieco trudności. Brzmi ono: „Kim chcesz być za X lat?". I bardzo często pada.

Nie rozumiem sensu zadawania tego konkretnego pytania. Dlaczego? Bo nie mam zielonego pojęcia, co będę chciał robić za pięć lat. Za kwartał – OK. Za rok – no, powiedzmy, że mniej więcej przez mgłę mogę postarać się zobaczyć tak odległą przyszłość. Ale dalej? Nie wiem, nie chcę wiedzieć i wolałbym się nad tym nie zastanawiać. Może będę programistą, może kierowcą rajdowym, a może pisarzem bestsellerowych horrorów. Nie wiem, nie wiem, nie wiem.

I wiesz co? To nie ma znaczenia. Można nie wiedzieć. Ale nie można się na to pytanie nie przygotować. Zastanów się przez chwilę i znajdź szczerą odpowiedź. Może brzmieć: „nie wiem", jednak przyda się do niej jakieś uzasadnienie, historyjka, przedstawienie punktu widzenia.

W moim przypadku: staram się żyć dniem dzisiejszym. Kiedyś planowałem na lata naprzód i te plany prawie nigdy się nie spełniały, co nie tylko powodowało irytację, ale także było całkowitą stratą mojego czasu.

Firmy pytają o to nie bez powodu. Znalezienie, zatrudnienie i wdrożenie nowego pracownika to ogromne koszty. Jeśli zmienisz pracę raptem po kilku miesiącach, pracodawca będzie mocno na minusie. Zatem odpowiedź „nie wiem, co będę robić za X czasu" trzeba uzasadnić i nakreślić horyzont, gdzie urywają się Twoje przewidywania. Tak, żeby nie przedstawić się firmie jak piórko na wietrze, które dziś jest tu, a za trzy miesiące może być zupełnie gdzie indziej. I zrobić to przekonująco, co bez ćwiczeń i przygotowań będzie bardzo trudne.

Firma: dlaczego chcesz dla niej pracować?

Na rozmowie otrzymasz pytanie: „dlaczego chcesz pracować akurat u nas?". Może być nieco głupio odpowiedzieć wtedy: „a gdzie ja w ogóle jestem?". To brak szacunku.

Poświęć dosłownie kilka minut na zapoznanie się ze stroną internetową firmy. Zobacz, czym się ona zajmuje, ile osób tam pracuje, w jakich krajach i miastach są biura. Jakich ma klientów, w jakiej branży działa. Wystarczy parę najbardziej podstawowych informacji, wyszukanych na telefonie w drodze na spotkanie. O sensowną odpowiedź na pytanie „dlaczego chcesz u nas pracować?" może być ciężko bez wcześniejszego sprawdzenia, co to za firma (chyba że trafisz do Google, Microsoftu, Facebooka lub Amazona).

⇨ *Swego czasu szukałem pracy w odległości maksymalnie dwudziestu dojazdominut od domu. Było mi kompletnie obojętnie, co to będzie za firma.*

Ale na rozmowie nie wypadało tak powiedzieć, więc wymyśliłem jakiś inny powód – nie pamiętam już jaki.

I nie jest to już istotne. Ważne, że byłem na to pytanie przygotowany. Pracę dostałem.

⏎

Jak zachować się podczas rozmowy kwalifikacyjnej?

Wreszcie nadchodzi ten dzień. Dzień targowy. I to Ty jesteś towarem.

Emocje sięgają zenitu. Dogania stres. Ręce się pocą, nogi się trzęsą, w głowie pustka...

Spokojnie, to normalne. Trzymaj się kilku zasad i będzie dobrze.

Rutyna lub wspólny stres

Nawet jeżeli się stresujesz, to wszystko jest w porządku. Może pomóc Ci uświadomienie sobie, że masz do czynienia z jedną z dwóch możliwych sytuacji.

Sytuacja pierwsza: Twój rozmówca ma takich spotkań dużo i widział już wszystko. Możesz się stresować, ile wlezie – i tak przed Tobą byli kandydaci bardziej zestresowani. Za chwilę Ci przejdzie. Obowiązkiem rekrutera jest takie pokierowanie rozmową, aby Cię nieco rozluźnić.

Podobną sytuację miewam, nagrywając podcast: ja jako gospodarz muszę zadbać o to, by mój gość nie stresował się nagraniem. Dlatego często przez pierwszych kilka minut, przed wciśnięciem record, gadamy o pierdołach.

Sytuacja druga: Twój rozmówca dopiero zaczyna i ma taką samą tremę jak Ty. Wtedy wspólnie znajdziecie jakąś drogę porozumienia.

Tak czy siak: to minie.

Prawda i tylko prawda

Zaczniemy od absolutnie najważniejszej zasady. Ja tę zasadę staram się stosować w całym życiu, ale w tym kontekście po prostu nie można postąpić inaczej...

Pisanie naciąganej nieprawdy w CV to idiotyzm – to już mamy za sobą. Ale jest coś jeszcze gorszego: mówienie półprawd na rozmowie prosto w oczy potencjalnemu pracodawcy! To idiotyzm do kwadratu.

> **Jeżeli coś umiesz – pochwal się tym.**
> **A jeżeli czegoś nie umiesz – przyznaj się do tego.**

Pójście inną drogą jest po prostu głupie. Jesteś na rozmowie, aby przekonać do siebie drugą stronę, a nie naściemniać. Jeżeli dostaniesz pracę za pomocą kłamstewek, prawda i tak wyjdzie na jaw już na początku pracy. Temat konsekwencji prześlizgnięcia się przez proces rekrutacji za pomocą półprawd już poruszyliśmy, pamiętasz?

Doświadczony rekruter zobaczy od razu, że coś nie gra. A wtedy wstyd taki, że hej! I to nie tylko w tej jednej firmie. Wbrew pozorom nasze środowisko nie jest tak wielkie, jak Ci się może wydawać. Informacje potrafią rozejść się błyskawicznie.

Tak, tak; programiści też plotkują!

Zadawaj pytania

Pamiętaj o tym, że rozmowa kwalifikacyjna to nie jednostronny wywiad. To egzamin dla obu stron. Firma testuje Ciebie, ale Ty także masz prawo do testowania firmy. Dopiero oba egzaminy zakończone pozytywnym wynikiem są podwaliną udanej współpracy.

> **Pytaj o wszystko, co ma dla Ciebie jakąkolwiek wartość.**
> **To właśnie teraz jest ten czas. Potem może być za późno.**

Zapewne masz swoje wyobrażenie „pracy idealnej". To na rozmowie możesz dowiedzieć się, czy właśnie do niej prowadzi aktualna rekrutacja.

O której zaczyna się i kończy praca? Jak często trzeba zostać po godzinach? Jakich narzędzi używa zespół projektowy? Jaki ma on stosunek do testów jednostkowych? W jaki sposób tworzy warstwę dostępu do danych? Jakie frameworki wykorzystuje w warstwie prezentacji? Czy firma pozwala na wyjazdy na konferencje i szkolenia, czy trzeba brać urlop? Czy odbywają się szkolenia wewnętrzne, wyjazdy integracyjne? Czy zespół spotyka się po pracy? Którego dnia miesiąca wpływa na konto pensja? Po jakim czasie można rozmawiać o awansie? Jak wygląda standardowa ścieżka kariery? Jak często następuje rotacja w zespole?

Chaotycznie? Tak! Bo to tylko kilka przykładów i to Ty musisz ułożyć własną listę kwestii dla Ciebie istotnych.

Nawet jeśli pytanie wydaje Ci się głupie, zadaj je, jeżeli ciekawi Cię odpowiedź na nie. Po to – między innymi – tam jesteś.

Po rozmowie musisz mieć wszystkie informacje potrzebne, by zdecydować, czy chcesz tam pracować. Po podpisaniu umowy jest za późno na odkrywanie przykrych niespodzianek w stylu „od dziś przez najbliższy kwartał będziesz klepać procedury składowane" albo „a tak przy okazji: twój projekt jest w Sharepoincie".

Rozglądaj się i obserwuj

Zwracaj szczególną uwagę na miejsce, które niedługo może być Twoim drugim domem. Czy pracujący tam ludzie wyglądają na zadowolonych? A może tylko czekają na fajrant? A nuż spostrzeżesz znajomą twarz, którą później wypytasz o to, jak jest „naprawdę".

Postaraj się zerknąć do pokoju programistów. Panuje tam grobowa cisza czy może gra muzyczka? Rozmawiają ze sobą czy każdy siedzi nieruchomo wpatrzony we własny ekran? Zobacz też, jak wygląda praca o tej porze dnia.

⇨ *Byłem kiedyś na rozmowie o godzinie osiemnastej. Praca szła pełną parą. Właściwie mogłem od razu obrócić się w progu i wyjść. Momentalnie*

wiedziałem, że to nie jest odpowiednie miejsce dla mnie. O tej porze chcę być już od dawna w domu.

Pamiętasz swoje przygotowania do mówienia o tym, co jest dla Ciebie ważne? Jak brak nadgodzin w moim przypadku. Bacznie obserwuj rozmówcę, gdy wymieniasz te kwestie. Niedowierzający uśmiech może dużo powiedzieć o realności wprowadzenia Twoich wymagań w życie.

Jeżeli któryś z kluczowych dla Ciebie warunków nie będzie spełniony, to szczerze radzę: poszukaj sobie innego miejsca.

Bądź sobą

W pracy spędzamy znaczną część życia. I – niezależnie, czy to dobrze, czy źle – decyzja o przyjęciu lub odrzuceniu naszej kandydatury zapada nie tylko na podstawie kompetencji czysto technicznych. Liczy się też charakter, osobowość, dopasowanie do zespołu. Tak, to naprawdę ma znaczenie, więc ostrożnie z przedwczesnymi zarzutami o „dyskryminacji".

Potencjalny pracodawca musi wiedzieć, kim naprawdę jesteś. W ten sposób będzie potrafił zaplanować Twoją rolę w zespole w razie podjęcia współpracy. Będzie także w stanie odrzucić Twoją kandydaturę, jeżeli uzna, że absolutnie nie nadajesz się do tego konkretnego zespołu – z korzyścią również dla Ciebie.

Jesteś ponurym mrukiem, który najchętniej nałoży słuchawki na uszy i zajmie się pracą, a wspólne obiadki nie są mu potrzebne? Tak jak – nie przymierzając – ja na początku swojej kariery? To nie sil się podczas rozmowy na sztuczne heheszki. Przecież już w pierwszych dniach wyjdzie szydło z worka.

Nie staraj się na siłę prowadzić zabawnej konwersacji, jeżeli nie leży to w Twojej naturze. Będzie to wyglądało żałośnie. Bardzo łatwo jest rozpoznać nienaturalne zachowania. Tak samo jednak jak w kwestiach technicznych – w personalnych również dobrze zachowywać się szczerze i uczciwie. Chcesz przecież, aby pracodawca wybrał Ciebie. Konkretnie Ciebie, a nie kukłę wyglądającą jak Ty, powstałą na potrzeby jednej rozmowy.

SPRZEDAJ się!

Programiści to często osoby skromne i nie lubią mówić o sobie. Nie umieją. To źle, bo jeśli Ty siebie nie zareklamujesz, to kto inny to zrobi?

Chwal się! Na pewno masz czym. Nie umniejszaj swojej roli w realizowanych projektach. Nie mów: „na pewno każdy by sobie poradził" ani „w sumie moje osiągnięcia to nic nadzwyczajnego". Bierzesz udział w procesie rekrutacji! Musisz przedstawić siebie w jak najlepszym świetle, podkreślić swoje mocne strony. W programistycznym projekcie najważniejszą i najbardziej pożądaną cechą nie będzie przesadna skromność ani nieumiejętność zaprezentowania sukcesu.

> **To nie jest konkurs na kokieterię.**

Oczywiście bez przesady! Nie chcesz wyjść na zadufanego w sobie buca. Ordynarnie narcystyczna kreatura nie wzbudzi sympatii.

Jak zatem nie zagalopować się za daleko? Przećwiczyć! O tym była mowa nieco wcześniej. Po prostu powiedz to, co masz do powiedzenia, przed lustrem albo kimś zaufanym. Możesz też nagrać się i przeanalizować wideo. Musisz po prostu potrafić w sensowny sposób pokazać pracodawcy, kim naprawdę jesteś. Bez przeginania w żadną stronę. Ale pamiętaj: najpierw Ty musisz docenić siebie, by później Ciebie docenili inni.

Durne zagadki

Niekiedy na rozmowie padają bardzo głupie pytania, typu: „ile kilogramów salcesonu zjadają co kwartał mieszkańcy Barcelony?". Na szczęście odchodzi się już od tego absurdu, ale czasami można jeszcze trafić na takie kwiatki.

Co wtedy? Jeżeli lubisz grać w takie gierki, to śmiało!

Ja nie lubię. Nie chcę się w te bzdury zagłębiać. Od lat prowadzę dyskusje na temat zadawania takich pytań podczas programistycznych rekrutacji i nie usłyszałem jeszcze sensownego argumentu przemawiającego za tą praktyką. Może w przypadku rekrutacji innych specjalistów – marketingowców? sprzedawców? – ale nie programistów.

Gdyby dzisiaj takie pytanie zadano mi na rozmowie kwalifikacyjnej, prosiłbym o uzasadnienie. Czy w codziennej pracy będę zajmował się takimi problemami?

⇨ *Na jednej z rekrutacji otrzymałem pytanie tego typu. Odpowiedziałem: „Ostatnio na Twitterze widziałem pewną praktykę rekrutacyjną. Grupie programistów zadaje się bezsensowne pytanie. Pierwszy, który powie* fuck it *i wyjdzie – dostaje pracę. Czy o to chodzi?”.*

Z dwóch rekrutujących mnie osób tylko jedna się zaśmiała.

Rozmowa techniczna

Rekrutacja składa się zwykle z kilku etapów. Pierwsze są prowadzone przez dział HR, by po wstępnym prześwietleniu przekazać Cię na rozmowę techniczną. Taka rozmowa może przebiegać… różnie.

Możesz odnieść wrażenie, że osoba po drugiej stronie stołu wcale nie chce tam być. Że wykazuje w Twoim kierunku zniecierpliwienie, antypatię. I wiesz co? Możesz mieć rację. Twój rozmówca prawdopodobnie nie szedł do pracy z zamiarem rekrutowania swoich przyszłych kolegów. On wcale nie jest zadowolony, że na niego padło. Ale ktoś to musi robić. Nie zrażaj się.

Wkrótce to Ty możesz być na jego miejscu i, zamiast wesoło klepać kodzik, będziesz uczęszczać na spotkania z kandydatami. I też będziesz się zastanawiać: „czemu akurat ja? nie może tego zrobić ktoś inny?”. Cóż, życie.

Zwróć jednak uwagę na poziom wiedzy takiego programistycznego rekrutera. Jeżeli jest mądry i doświadczony, to w nowej pracy będzie się od kogo uczyć!

Może to Twój przyszły mentor?

Byłoby super! Bo niestety nie jest to regułą.

Porady dla tej drugiej strony, czyli programistycznego rekrutera, znajdziesz w dalszej części książki.

4 Nowa praca

Jak rozwiązywać zadania rekrutacyjne?

Niektóre rekrutacje są formalnością: jeden telefon, jedno spotkanie – i już. Na innych musisz się bardziej postarać.

Czy zadania rekrutacyjne to zło?

Wielokrotnie spotkałem się z poglądem, że podczas rekrutacji pracodawca nie powinien wymagać od kandydata dużego zaangażowania.

Niektórzy uważają nawet, że za sam projekcik sprawdzający umiejętności powinno się płacić!

Stąd prosta droga do absurdu. To firma organizuje rekrutację i jej prawem jest sprawdzanie Cię tak, jak chce. Twoim prawem jest zrezygnowanie z rekrutacji. I po problemie.

Ale domaganie się zapłaty za pojawienie się na rozmowie kwalifikacyjnej albo za rozwiązanie zadania? To już nasza, programistyczna, patologia.

Dobra rekrutacja zawiera element kodowania. Nie na tablicy, nie przy biurku rekrutera – ale w swoim środowisku. W domu. Po godzinach. Bo jak inaczej zweryfikować prawdziwe programistyczne umiejętności? Jak zasymulować warunki codziennej pracy?

Zadanie rekrutacyjne świadczy o firmie

Żaden „test wyboru" ani żadna rozmowa kwalifikacyjna nie pozwalają na sprawdzenie kandydata czy kandydatki tak dobrze, jak niebanalny problem do zaimplementowania. Powinien być on choć trochę zbliżony do zadań na co dzień wykonywanych w firmie.

Już sam fakt otrzymania zadania do rozwiązania powinien podrażnić ośrodek zadowolenia w Twoim mózgu. O, coś jest na rzeczy! To może być naprawdę dobre miejsce do dalszego rozwoju. Potraktuj to jako znak. Ta firma faktycznie przykłada wagę do jakości zatrudnianych programistów. To bardzo dobrze – także dla Ciebie! Jeśli pomyślnie przejdziesz test, będziesz pracować

ze specjalistami, którzy również go rozwiązali, a nie z osobami całkowicie przypadkowymi, z łapanki.

Sposób rozwiązania (lub nawet nierozwiązania) zadania bardzo wiele o Tobie powie. Myślisz, że zapotrzebowanie na programistów i programistki jest teraz ogromne, więc możesz rozwiązać zadanie rekrutacyjne w dowolny sposób, a i tak każda robota jest Twoja? Nie! Na co zwrócić szczególną uwagę? Uwierz mi; nawet w naszej branży zła sława może mocno zaszkodzić przyszłej karierze. Wspominałem już o plotkujących programistach, co nie?

Zapewnij oczekiwany rezultat

Opowiem Ci pewną historię.

⇨ *Za czasów uczelnianych studiowałem przez rok w Danii. Zaliczenie jednego z przedmiotów wymagało dobrania się w międzynarodowe zespoły i realizację dowolnego projektu. Nasz ośmioosobowy, polsko-chińsko-rumuński team przeszedł do historii tej uczelni ze swoim kierowanym do zagranicznych studentów portalem internetowym z forum dyskusyjnym. W stan osłupienia bynajmniej nie wprawiło egzaminatorów rewolucyjne podejście do tematu aplikacji internetowych.*

Nasz egzamin składał się z dwóch części. Najpierw w siedem osób zaprezentowaliśmy rozwiązanie, poklikaliśmy po stronce, pokazaliśmy forum. Demo, pytania, wyjaśnienia i odpowiedzi.

Ósmy członek naszego zespołu stał jednak z boku i nie powiedział ani słowa. Na koniec komisja zapytała się go więc, dlaczego nie brał udziału w demo i jaki był jego wkład. Jego odpowiedź zwaliła wszystkich z nóg:

„Ja napisałem niezależny komponent. Jest to program w C++, który po zainstalowaniu powoduje blokowanie autorun płyt CD po włożeniu ich do czytnika. To bardzo potrzebne narzędzie, ponieważ w ten sposób często instalują się wirusy. Zaimplementowałem to, bo nie umiem programować WWW, ale umiem inne rzeczy".

Domyślasz się, jaki związek z tematem ma ta historyjka?

Ubiegając się o stanowisko w firmie dającej do rozwiązania ciekawe zadanie, jesteś na dobrej drodze. To sprawdzian. Zachowaj się tak, jakby to była normalna codzienna praca.

Jedne z najważniejszych zdolności programisty to czytanie ze zrozumieniem i realizacja implementacji zgodnie z założeniami.

Komunikacja dwukierunkowa

Proces rekrutacji sprawdza nie tylko umiejętności techniczne, ale również zdolność komunikacji – zrozumienie cudzych oczekiwań oraz określenie swoich założeń i ewentualnych wątpliwości. Ma to szczególne znaczenie w przypadku pracy zdalnej, ale przecież nie tylko.

Jak zachowasz się, nie mając stu procent informacji koniecznych do wykonania zadania? To test! Samodzielnie podejmiesz (prawdopodobnie błędną) decyzję czy dopytasz o szczegóły?

Taki trik stosuję czasem, prowadząc szkolenia: daję wybrakowane instrukcje i obserwuję, co się dzieje. Mało kto prosi o więcej danych. W tym zawodzie mamy w zwyczaju podejmować decyzje na własną rękę. I często są to decyzje ukryte w kodzie, niezakomunikowane, a niejednokrotnie – całkowicie nietrafione.

Załóż, że w zadaniu może celowo brakować pewnych informacji. Dopytuj, proś o szczegóły oraz kontekst. Informuj o własnych założeniach i decyzjach. Dociekliwość nie oznacza głupoty i natręctwa; wręcz przeciwnie! To bardzo pożądana – i niestety nieczęsto spotykana – cecha. To oznacza, że dogłębnie analizujesz problem i chcesz go rozwiązać raz a dobrze.

Umiarkowanie

Mając wszystkie informacje wymagane do rozwiązania zadania, zrób to, o co Cię proszą. Najprawdopodobniej będzie to napisanie kodu typu „logika", realizującego pewne „wymagania biznesowe". I na tym się skup. To koduj.

Chcesz się pochwalić znajomością modnych trendów? Pewnie, zastosuj CQRS! Użyj bazy NoSQL! Zaimplementuj mikroserwisy! Bo dlaczego nie?

Ale... dlaczego TAK? Każdą taką zabawkę koniecznie uzasadnij.

> **Do rozwiązania nie wrzucaj wszystkiego,**
> **co znajdziesz w najnowszym Technology Radar**
> **od ThoughtWorks.**

Każda nieuzasadniona komplikacja to długoterminowy koszt, ciągnący się w projekcie latami. Pokaż, że to rozumiesz.

Niekoniecznie na 100%

Jeżeli nie masz czasu na realizację zadania w stu procentach, zrób chociaż trochę. Tyle, ile jesteś w stanie. A potem udokumentuj, co zostało zrobione, a co należy jeszcze zrobić. Firma zrozumie, że robisz to po godzinach i masz swoją pracę, swoje życie.

Napisz też, jakie byłyby Twoje kolejne kroki, gdyby nie brak czasu. Dostarczenie kompletnego rozwiązania nie zawsze jest konieczne do kontynuowania rekrutacji.

Pracowałem w firmie korzystającej podczas rekrutacji z zadania, którego rozwiązanie mogło zająć nawet trzydzieści godzin. Tak! Oczywiście nie każdy jest w stanie tyle wygospodarować. Ja bym nie był. Ale to nie oznacza, że się takie osoby z góry skreśla.

Umiejętność raportowania trudności („nie jestem w stanie zrobić wszystkiego"), najlepiej z wyprzedzeniem, to cenna cecha.

Zwróć jednak uwagę, by w przypadku oddania niekompletnego rozwiązania skupić się na tej właściwej części. Zrób to, co najważniejsze. To, o co głównie chodzi w zadaniu. Niech „częściowe rozwiązanie" nie będzie kawałkiem generycznego kodu, przeniesionym z dowolnego innego projektu.

Decyzje

Napisałem wcześniej, by dokumentować swoje decyzje. Dlaczego to rozwiązanie? Dlaczego to narzędzie? Dlaczego ta biblioteka?

Ale nie jest to jednoznaczne ze skopiowaniem dokumentacji! Wiesz; tego marketingowego fragmentu, wynoszącego wybrany element pod niebiosa. Chodzi o Twoją opinię, Twoje rozumienie problemu i Twoją propozycję rozwiązania. W tym konkretnym kontekście. Jeżeli Twoją argumentację można zastosować w odpowiedzi na każde dowolne zadanie rekrutacyjne, odpuść je sobie, bo nie spełnia powyższego warunku.

To, co potrzebne

Ktoś (może nawet więcej niż jedna osoba) musi to wszystko później przeczytać. Przed napisaniem linijki kodu zastanów się, czy jest ona bezpośrednio powiązana z zadaniem. Wiem, że być może w obecnej pracy nie masz możliwości zaprogramowania czegoś fajnego i że przy tej okazji chcesz się wyżyć, więc ciężko powiedzieć Ci „stop". Ale pomyśl o recenzentach Twojego rozwiązania.

To nie jest projekt „do zabawy". Dlatego nie dłub własnej infrastruktury. Nie pisz swojego load balancera (chyba że było to Twoim zadaniem). Nie pisz swojej biblioteki do komunikacji (chyba że zadaniem było napisanie biblioteki do komunikacji).

> **Nie pisz „programu w C++, który po zainstalowaniu powoduje blokowanie autorun".**

Pomyśl, jak w Twoich oczach wyglądałby programista tak tłumaczący swój błąd: „co prawda nadal nie działa, ale za to zrobiłem masę fajnych dodatków". Raczej można by to zinterpretować jako: „nie umiem tego, czego wymagacie, ale umiem milion innych rzeczy i to na nie zmarnuję swój czas w przyszłości".

Dobrze, jeśli zauważasz potrzebę infrastrukturalnych rozwiązań. Podkreśl to, definiując odpowiedni interfejs. Ale nie dokładaj do niego kilkusetlinijkowej implementacji! Możesz ewentualnie słownie zwrócić uwagę, że „w tym miejscu

przydałby się taki i taki rodzaj *cache*". Nie zmuszaj jednak osób po drugiej stronie do analizy Twojej implementacji tego *cache*.

Jeżeli zaintrygujesz rekruterów swoim podejściem, poproszą o uzupełnienie rozwiązania przykładową implementacją. I wtedy: hulaj dusza!

Czas

Czytaj ze zrozumieniem. Lista oczekiwanych od Ciebie rezultatów zadania nie wzięła się znikąd. Zrób to, o co Cię proszą. Nie wymyślaj dodatkowych artefaktów dla zabawy. To nie jest hobbystyczny projekt *open source*, tylko biznes!

Im więcej dostarczysz, tym bardziej kosztowna czasowo będzie Twoja rekrutacja. A czas to przecież najcenniejsza waluta. Nie marnuj swojego czasu, implementując to, co niepotrzebne. Rób, czego od Ciebie oczekują, i nie marnuj czasu osób oceniających Twoje rozwiązanie. Czytanie Twojego tekstu zajmuje komuś część życia, więc niech wszystko zawarte w dostarczonej paczce będzie wartościowe. Czytanie kodu również zajmuje komuś czas – nawet więcej czasu niż tekst. Niech więc i kod będzie zwięzły, skupiony na rzeczach istotnych.

Z perspektywy osoby oceniającej rekrutację zdradzę Ci: przebijanie się przez kilka z rzędu definicji CQRS może człowieka bardzo zmęczyć. To najdelikatniejsze określenie moich uczuć w tym temacie.

Bądź kimś, kogo firma chce – a nie musi – zatrudnić!

Pierwsze dni w nowej pracy

Prędzej czy później każdy staje przed wyzwaniem: dołączam do stada! Niezależnie od doświadczenia i stanowiska – jest to trudne zadanie.

Ja mam za sobą wiele takich „dołączeń" w najróżniejszych okolicznościach. Dołączałem i jako junior, i jako senior, i jako „noname" i jako lider, i jako „znany

bloger", i jako jeden z trzech polskich MVP w kategorii C#... Raz robiłem to lepiej, raz gorzej. Za każdym razem analizowałem: co poszło w porządku, a co mogę poprawić?

⇨ *Zobacz scenariusz idealny. W jednej z pierwszych firm na powitanie dostałem zadanie: sprawdź, dlaczego nasz system wolno działa. Program był napisany w najnowszych wówczas technologiach i korzystał z niestabilnej wersji (chyba nawet alfa) biblioteki Microsoftu: LINQ2SQL. I zespół miał problem: fajnie się w tym pisało, ale działało baaaaardzo wolno.*

W ciągu pierwszego miesiąca pracy w nowej firmie udało mi się znaleźć przyczynę. Wymagało to dużego zaangażowania, dekompilowania, czytania zagmatwanych źródeł. Ale w końcu: eureka! W LINQ2SQL jest błąd! Normalna sprawa, bo biblioteka nie była jeszcze gotowa do produkcyjnego wykorzystania. Ale oprócz tego, że znalazłem błąd, zaproponowałem także rozwiązanie, które w naszym przypadku poprawiło wydajność wielokrotnie.

To było idealne „wejście w zespół".

Nie narzekaj i nie krytykuj

Najgorsze, co możesz zrobić, to krytykować cokolwiek już od pierwszego dnia. Narzekacze i krytykanci skutecznie utrudniają sobie integrację z zespołem. Od razu pojawiają się negatywna atmosfera, stres i smutne buzie.

Poniżej znajdziesz kilka fraz, które nigdy nie powinny paść z Twoich ust. Przynajmniej przez pierwszy miesiąc:

Ale ten kod słaby! Kto to pisał??

To wasze podejście do agile jest beznadziejne! Dziwne, że cokolwiek dostarczacie.

A w poprzedniej firmie to mieliśmy fajniejsze narzędzia!

Jeszcze nie przesiedliście się na najnowszą wersję frameworka??

Naprawdę muszę czekać tydzień na konfigurację komputera? Marnujecie mój czas!

Przez pierwsze dni nie możesz wiedzieć, w jakich okolicznościach powstawało dane rozwiązanie, z czego wynika kultura pracy, jakie są *best practices* i jaka jest ich geneza. Może kod, który tak bardzo Cię uwiera, pisał o drugiej w nocy osamotniony programista na kilka godzin przed wdrożeniem? A może to zaszłości sprzed dziesięciu lat, nad których poprawą cały zespół – w swoim spokojnym tempie – pracuje od miesięcy?

Nie stawiaj zasieków separujących „wy" od „ja". Teraz jesteś w zespole, jest tylko „my". Bardzo trudno będzie zatrzeć pierwsze złe wrażenie.

Nie wywracaj wszystkiego do góry nogami

Twoje praktyki mogą być najsłuszniejsze na świecie. Lubisz rozpoczynać nazwę zmiennej od podkreślenia, a oni stosują notację węgierską, jak w średniowieczu? To od dzisiaj także stosujesz notację węgierską.

Zwykle nie jestem zwolennikiem tego powiedzenia, ale przy dołączaniu do zespołu niech to będzie Twoje przykazanie:

> **Kiedy wejdziesz między wrony, musisz krakać jak i one.**

Praktyki, konwencje, zwyczaje... To wszystko dojrzewało w Twojej nowej firmie latami. Owszem, niektóre z nich faktycznie mogą być bez sensu, ale inne wymagają odpowiedniego doświadczenia, by je zrozumieć! Odpowiedniego kontekstu. Postaraj się ten kontekst uchwycić, jednocześnie stosując się do nich. Ja takie zastane praktyki traktuję jak znak zakazu wyprzedzania: nie zawsze rozumiem, dlaczego akurat w tym miejscu go postawiono, ale jest on święty i nienaruszalny. No, w dziewięćdziesięciu pięciu procentach przypadków, *wink, wink*.

Masz uwagi do codziennej pracy w swoim nowym miejscu? Świetnie! Zapisz je, wynotuj. I zachowaj na później. Po kilku tygodniach może okazać się, że masz całkowitą rację i swoim świeżym spojrzeniem faktycznie poprawisz procesy i reguły panujące w zespole. Chwała Ci za to. Ale potrzebujesz do tego szerszej perspektywy i zrozumienia przyczyn obecnego stanu rzeczy, więc nie wyrywaj się z tym za wcześnie.

Postaw na samodzielność!

W jednym zespole dostaniesz swojego anioła stróża – osobę, której zadaniem będzie wprowadzenie Cię w projekt. W innym zespole rzucą Cię na głęboką wodę z błogosławieństwem i okrzykiem: „no to jedziesz!". W obu przypadkach jednak wszyscy niezmiernie się ucieszą, jeśli zminimalizujesz koszt swojego dołączenia do zespołu. Koszt liczony godzinami cudzej pracy.

Na początku potrzebujesz pomocy. To oczywiste i całkowicie normalne. Jednak im mniej będziesz zawracać głowę pozostałym członkom zespołu, tym bardziej Cię będą szanować. Nie lataj z każdą pierdołą do swojego anioła stróża. Nie odrywaj ludzi od pracy, nieustannie, non stop, cały czas. Eksperymentuj. Próbuj. Eksploruj. Poznawaj projekt, poznawaj nowy świat.

Wcześniej upewnij się jednak, że niczego poważnie nie zepsujesz. Zapytaj wprost, gdzie są granice dowolności i eksperymentów. W większości przypadków prawdopodobnie możesz robić, co chcesz, dopóki kod nie opuści Twojej maszyny. Ale lepiej jest to potwierdzić, niż potem zbierać szczątki produkcyjnej bazy danych, do której ktoś niechcący dał Ci dostęp.

Oczywiście nawet proste zadania mogą na początku sprawiać trudności. Każdy to zrozumie i na pewno otrzymasz konieczną pomoc. Jednak po raz kolejny zwrócimy szczególną uwagę na wielokrotnie poruszony już temat. Bycie świeżakiem w zespole nie zwalnia Cię z obowiązku szanowania czasu. Zarówno swojego, jak i cudzego.

Jak to osiągnąć?

Nie marnuj swojego czasu

Swój własny czas bardzo łatwo zmarnować, wykręcając się teoretycznie sensownymi wymówkami. „Nie wiem, jak rozwiązać zadanie, więc szukam w Google podpowiedzi". Brzmi dobrze. Dopóki nie dodasz na koniec: „od czterech dni".

Samodzielność jest pożądaną cechą, ale każda Twoja godzina kosztuje. Ustaw sobie okienko czasowe. Kwadrans? Pół godziny? Godzina? Tyle czasu maksymalnie powinno zająć Ci ruszenie do przodu z jakimkolwiek problemem.

Pokonanie choćby minimalnego dystansu. Jeżeli zegar tyka, a Ty wciąż stoisz w miejscu, to czas spojrzeć prawdzie w oczy: potrzebujesz pomocy. Nie traktuj tego jako „poddanie się", lecz jako „wezwanie posiłków". To w końcu praca zespołowa!

Nie marnuj cudzego czasu

Niekiedy okaże się, że masz wiele małych problemów. Z pierdołami, jakich wiele każdego dnia. Bardzo kuszące może wydawać się natychmiastowe wołanie o pomoc i odrywanie innych od pracy – i tak co kilka minut. Ale to zła praktyka, bo paraliżujesz pracę zespołu!

Zawsze zadaj sobie pytanie: dostanę odpowiedź – i co dalej? Zakładając, że już uzyskasz rozwiązanie dręczącego Cię problemu: jaki będzie Twój kolejny krok w realizacji zadania? Zapisz problem, a potem udaj, że został rozwiązany. Co robisz?

Najprawdopodobniej na dalszej drodze znowu napotkasz przeciwności. I dobrze, tak ma być! Posłuchaj jednak mojej rady i zadaj od razu kilka pytań. Oderwiesz kogoś od pracy tylko raz. Z pewnością zostanie to docenione.

Warto pójść o krok dalej i – zamiast od razu uderzać pytaniami, bez pardonu, nie zważając na okoliczności – ustalić reguły komunikacji. Zapytaj: „Mam parę problemów, z którymi możesz mi pomóc. Kiedy znajdziesz dla mnie chwilę?". Kulturalnie umówisz się na „okienko pytań", zamiast odrywać pomagającą Ci osobę od pracy w wybranym przez Ciebie momencie. To pomagający powinien decydować o czasie Waszego słodkiego *tête-à-tête*.

⇨ *Miałem kiedyś ekstremalny przypadek: pewien programista, nawet w wypadku drobnych problemów, atakował mnie od razu rozmową wideo na Skypie. Niezależnie od tego, co robię: buch, dzwonek, kamerka – jedziemy! Kilka razy dziennie.*

Nie rób tak, bo to naprawdę przeszkadza i irytuje (z trudem powstrzymuję się przed użyciem bardziej dosadnego określenia).

Proś o *code review*

Niezależnie od stanowiska i doświadczenia: zawsze postaraj się, aby ktoś zerknął na Twój kod. Nawet jeśli jesteś Linusem Torvaldsem i nigdy nie popełniasz błędów, los może Cię zaskoczyć.

Skąd dowiesz się, że nazwa klasy nie może kończyć się na *Service*? Kto powie Ci, że dodając zależność od dodatkowego pakietu, musisz dodatkowo zmodyfikować specjalny plik znajdujący się w katalogu */utils/docs/mgmt/dep.xml*? Jak domyślisz się, że po zmianie numeru wersji systemu trzeba dodać wiersz do tabeli *application_versions* w systemowej bazie danych?

Takie zależności i nieoczywiste praktyki zdarzają się bardzo często, a wiedza o nich bywa zagrzebana w tonie nieaktualizowanej dokumentacji. Oraz w głowach Twoich współpracowników.

Prośba o *code review* to nie wstyd. Nie jest to równoznaczne ze stwierdzeniem: „potrzebuję Twoich oczu, bo umiesz więcej ode mnie". To raczej przyjacielskie:

> **„Hej, spójrz! Wyciągnąłem zawleczkę!**
> **Pomóż mi rzucić ten granat jak najdalej od nas".**

Zintegruj się

Ja zawsze miałem praktykę: zero nadgodzin. Do pracy przychodziłem o stałej porze, a biuro opuszczałem równo osiem godzin później.

Nie potrzebowałem przerwy obiadowej. Jadłem własne rozpaćkane kanapki przy komputerze. Nie potrzebowałem „przerwy na reset" przy firmowych piłkarzykach czy konsoli. Chciałem wyjść do domu równo osiem godzin po otwarciu drzwi. To taka moja praktyka. Choć to nie oznacza, że nie byłem pasjonatem albo że się nie starałem! Po prostu miałem także inne rzeczy do zrobienia poza pracą.

Jednak teraz, po dłuższych refleksjach, nawet ja nagiąłbym tę swoją świętą zasadę podczas dołączania do nowego zespołu. Warto lepiej poznać ludzi, z którymi będziesz pracować. Przez pierwszy tydzień pójdź na ten wspólny lunch

– nawet jeśli nie planujesz tego w przyszłości. Postój chwilę w firmowej kuchni na kawie – choćby przez to trzeba było zostać w pracy kwadrans dłużej.

Po tygodniu wróć do swoich zwyczajów – będziesz już wtedy częścią zespołu, a nie „tym nowym dzikusem". Ja nim często byłem. Nie warto.

Jak często zmieniać pracę?

Odpowiedź jest prosta: „co trzy lata albo częściej".

To oczywiście żart. Ha, ha! Chyba że zapytasz nieodpowiednią osobę. Na przykład kogoś, kto zaraz potem zaproponuje: „znajdę Ci nową pracę w chwilę, *pick me, pick me*!". Ale nie doda: „i zgarnę za to kilkanaście tysięcy prowizji".

Tak naprawdę tylko jedna osoba na świecie zna odpowiedź na to pytanie: Ty! Ale mogę pomóc Ci tę odpowiedź znaleźć.

Wszystko w porządku? To o co chodzi?

W naszej branży przepływ ludzi między firmami jest dość duży, ale nie warto na to patrzeć. Każdym kierują inne motywy, każdy ma własne priorytety. Sam faktycznie podczas swojej kariery w żadnej firmie nie spędziłem trzech lat. Ale znam programistów pracujących po pięć, siedem czy nawet dziesięć lat w jednym miejscu. I są zadowoleni.

Niech każdy działa wedle swoich potrzeb.

Nie ma sensu zmieniać pracy, w której jest Ci dobrze. Z banalnego powodu: naprawdę łatwo trafić gorzej. Nie zwalnia to jednak z obowiązku dbania o swoją przyszłość! Bardzo niebezpieczne jest wygodne życie, jak pączek w maśle, bez choćby odrobiny planowania na wypadek wystąpienia złego scenariusza. Nieodpowiedzialnie jest zakładać, że „zawsze wszystko będzie dobrze, tak samo, bez zmian". Nie zazdrościłbym programiście ani programistce, którzy

po wielu latach spędzonych u tego samego pracodawcy, w tym samym zespole, nagle budzą się z... ręką w nocniku. Gdy okaże się, że nie mają dziesięciu lat doświadczenia, ale jeden rok doświadczenia powtórzony dziesięciokrotnie (uwielbiam to zdanie).

Co wtedy? Cóż... bieda!

Polisa

Zostać na lodzie można z różnych powodów. Nigdy nie zakładam złej woli, po niczyjej stronie, w żadnej sytuacji. Jestem nieco naiwny, ale dobrze mi z tym. Jednak firmy padają, zamykają oddziały, przenoszą biura. Żaden biznes nie zrezygnuje z interwencji w kryzysowej sytuacji tylko dlatego, że jednemu lojalnemu programiście nie będzie to na rękę. I wtedy... klops.

Dlatego niezmiernie istotne jest, aby nieustannie się rozwijać. Nie musi to oznaczać ciągłych zmian pracy! W jednej firmie można podjąć się różnych obowiązków. Uczestniczyć w różnych projektach, używać różnych technologii i pracować w różnych zespołach.

Słowo „różny" jest tutaj kluczowe: liczy się różnorodność. W ten sposób zdobywasz tak cenne doświadczenie i poszerzasz swoją perspektywę.

Niekiedy może się wydawać, że „się nie da". Firma przykleiła Cię w jednym miejscu – i koniec. Jednak to Twoim obowiązkiem jest zasygnalizowanie, że oczekujesz czegoś innego, że potrzebujesz innych wyzwań, zmian. Dopiero gdy takie środki zawiodą, zacznij rozważać bardziej drastyczne ruchy.

Zdobywanie kolejnych kwalifikacji i poszerzanie zakresu odpowiedzialności niekoniecznie wiąże się ze skakaniem po firmach. Przecież wszędzie pracują różni ludzie, na różnych stanowiskach. Często w ramach jednej organizacji można bezpiecznie spróbować całkowicie różnych pozycji. Z programisty na managera? Z managera na analityka? Z analityka z powrotem na programistę? Pewnie, dlaczego nie?

Kluczem jest jasne komunikowanie swoich potrzeb oraz odrobina cierpliwości i wyrozumiałości.

Beznadzieja?

Warto pamiętać, że życie jest krótkie. Czas to najważniejsza waluta. Jedyny zasób, którego nie kupisz. Dlatego grzechem byłoby marnowanie go w sytuacji beznadziejnej.

Nie podoba Ci się coś, ale po klarownych wyjaśnieniach nikt nie reaguje? Ciągle kończy się na zapewnieniach, że „kiedyś będzie dobrze"? Męczysz się, i końca nie widać? W takiej sytuacji nie ma sensu czekać nie wiadomo na co. Bo... pamiętasz? Życie jest krótkie. I jest Twoje!

⇨ *Kiedyś miałem sytuację, w której z takich właśnie powodów musiałem odejść z firmy. Z firmy, którą bardzo lubiłem (i lubię nadal)! Lubiłem organizację, lubiłem szefostwo, współpracowników. Lubiłem wszystko. Poza projektem, w którym się znalazłem.*

Przez kilka miesięcy dawałem do zrozumienia, że długo tak nie wytrzymam. Nie dało się z tym nic zrobić. Rozumiałem, że to nie jest wina ludzi w firmie. Ci ludzie rozumieli, że po prostu nie jestem w stanie tak dłużej działać. Rozstaliśmy się ze smutkiem, ale w zgodzie. Kontakty utrzymujemy do dziś.

Szczera komunikacja jest kluczowa nawet w sytuacji bez wyjścia.

Checkpoint

Odpowiedzialność za własny los, za własne szczęście i spełnienie wymaga, by raz na jakiś czas zapytać siebie: czy robię to, co chcę teraz robić? Czy zmierzam w dobrym, świadomie obranym kierunku? A może dryfuję z prądem, kompletnie nie kontrolując swojego życia?

Takie pytanie warto sobie zadać chociażby raz na rok, w luźnym okresie świątecznej zadumy. Muszę Cię przestrzec, że szczera odpowiedź może nieco skomplikować życie. Ale lepiej wykryć takie sytuacje dzięki chwili autorefleksji, niż czekać na objawienie się ich w inny sposób. Cały czas dojrzewamy, z roku na rok inaczej patrzymy na świat. Zmieniają się nasze priorytety i cele. Sukces

może oznaczać dla Ciebie zupełnie co innego niż jeszcze kilka lat temu. Wiem, że u mnie takie pojęcia nieustannie ewoluują.

Może się okazać, że wymarzona kiedyś praca teraz wcale nie jest taka wspaniała. Powtórzę: lepiej się o tym dowiedzieć w kontrolowanych warunkach, dzięki chwili zastanowienia nad swoją sytuacją, a nie po niespodziewanym ataku wypalenia zawodowego czy czegoś jeszcze gorszego (ćśśś).

Przy najbliższej okazji zadaj sobie to pytanie: „czy teraz robię w życiu to, co chcę robić?". Wiem, wiem, to takie głębokie... Ale każdy człowiek na to zasługuje. To przysługa, którą możemy sobie wyświadczyć.

Ciągła gotowość

Warto być przygotowanym na ewentualne zmiany. Lojalność wobec firmy to – w przypadku poprawnych stosunków – bardzo ważna cecha. Nie oznacza ona jednak, by zachowywać się biernie i liczyć na to, że zawsze wszystko będzie dobrze.

Nieustannie dbaj o swoją wartość na rynku pracy. Trzymaj rękę na pulsie. Przygotuj się na nieznane. To prawie nic nie kosztuje, a jest Twoją niezawodną polisą ubezpieczeniową.

Jak to robić? To proste: co jakiś czas aplikuj do innej pracy!

Co to da? Jeśli Cię odrzucą, dostajesz sygnał ostrzegawczy. Ups, coś jest nie tak! Może jest Ci teraz za wygodnie i zaczynasz osiadać na laurach? Może świat poszedł do przodu, a Ty tkwisz w przeszłości? Może nie rozwijasz się wcale tak bardzo, jak Ci się wydaje?

To akurat bardzo dobry i pożądany znak. Konieczność zastanowienia się nad sobą i swoją sytuacją.

> **Czy na pewno chcesz tkwić w całkowitej zależności od obecnego pracodawcy?**

Prawdopodobnie nie. Nawet jeśli to „najlepsiejsza" i „najmilejsza" firma na świecie.

A jeśli przejdziesz rekrutację, to... gratulacje! Świetna informacja! Wcale nie musisz przyjmować oferty. Po prostu przybij sobie piątkę, uśmiechnij się i ruszaj dalej ze swoim życiem.

Ruszaj ze spokojem, że w razie czego masz inne opcje.

Bonus

Dodatkowa zaleta regularnego uczestniczenia w rekrutacjach: skąd wiesz, że teraz nie omijają Cię wspaniałe okazje, o których nawet nie śnisz? Omijają, bo nie dajesz sobie szansy na ich poznanie. Wzięcie udziału w rekrutacji raz na jakiś czas jest bardzo zdrowe. Ot, taka programistyczna higiena.

Nie ma w tym niczego złego. Ja bym nawet nie krył tego faktu przed pracodawcą. To spowoduje, że Wasze stosunki pozostaną na odpowiednim poziomie w kontekście biznesowym. Twoja praca nie będzie, jak to się mówi, *oczywistą oczywistością, taken for granted*.

I jeszcze jedno: nie miej wyrzutów sumienia. Ty nie marnujesz czasu ludzi odpowiedzialnych za rekrutację w firmie, do której aplikujesz bez zamiaru przyjęcia oferty. Prowadzenie rekrutacji to ich praca. Dostają za to pensje. Oni robią swoje, ty robisz swoje. I tyle.

⏎

Z jakiego powodu zmieniać pracę?

Różne słyszało się historie o programistach i programistkach zmieniających pracę. Przechodzenie z jednej firmy do drugiej, bo dali MultiSport. Powrót do tej pierwszej, bo rzucili kilka złotych więcej. I tak w kółko, z kwiatka na kwiatek.

Jestem zdecydowanym zwolennikiem zasady „żyj i daj żyć innym". Niech każdy robi, jak uważa. Ale ciężko ten temat pominąć, bo bezmyślnym postępowaniem łatwo jest sobie zaszkodzić.

W poprzednim rozdziale rozpoczęliśmy temat zmiany pracy. Motywacje mogą być najróżniejsze, ale pewne minimum przyzwoitości powinno jednak obowiązywać.

Kiedy warto?

Pierwszy całkowicie uzasadniony powód to zastój i nuda. Klepanie ciągle tego samego, w ten sam sposób. Nie będę po raz kolejny przytaczał zdania o powtórzonym dziesięciokrotnie jednym roku doświadczenia (ups!), ale dokładnie o to chodzi. Dopóki Ci to odpowiada i świadomie decydujesz się na stanie w miejscu (lub cofanie) – jest w porządku. Ale przy pierwszych oznakach niepokoju albo wypalenia zawodowego wiedz, że to właśnie dlatego. Potrzebna jest interwencja. Musisz zasygnalizować w firmie, że coś jest nie tak. W wypadku braku reakcji nie ma co czekać, trzeba działać. Tu chodzi o Twoje zdrowie. Naprawdę!

Do rozglądania się za nowym miejscem może też pchnąć jakość pracy. Niska. Zalepianie dziur kolejnymi dziurami. Byle jak, byle szybciej. Robienie na łapu-capu, byle działało. Posklejać taśmą, pochuchać – i na produkcję! No nie, na dłuższą metę to się nie sprawdzi. Zmiana takiego stanu rzeczy to długotrwały proces, ale samo się nie naprawi. Próbujesz interweniować i nic? Albo słyszysz wieczne „w następnym kwartale się postaramy?". Kiedyś trzeba powiedzieć „dość". Tu chodzi nie tylko o uczciwość względem klienta i użytkowników, ale także o Twoje umiejętności. Tkwiąc w takim bagnie, potrzebujesz dużo sił, by utrzymać się na powierzchni. Te siły w końcu się wyczerpią. Trzeba spojrzeć w przyszłość i zastanowić się, co będziesz reprezentować sobą w potencjalnym kolejnym miejscu pracy. Jeśli będzie to tylko: „znam wiele sposobów, jak nie należy wytwarzać oprogramowania, ale druga strona medalu jest mi obca", to średnio to wygląda.

Biorąc pod uwagę fakt, że praca jest drugim domem (czasami spędzamy tam przecież większość doby, a przynajmniej jej „świadomej" części!) – musimy się w niej dobrze czuć. Wychodzisz z pracy jako kłębek nerwów? Stres Cię zżera? Przynosisz negatywne emocje do domu? Czas to zmienić! W takiej sytuacji

często nie ma nawet marnych szans na poprawę przez rozmowę i zakomunikowanie problemu. Trzeba odpowiedzieć sobie na pytanie:

> **Żyjesz, by pracować, czy pracujesz, by żyć?**

Co jest ważniejsze? Wiadomo, że w życiu zawodowym zdarzają się różne sytuacje. Ale jeśli z tego powodu cierpi Twoje otoczenie – rodzina i przyjaciele – wiedz, że nie musi tak być.

A może zupełnie nie masz życia pozazawodowego? Może w ramach sztucznie wymuszonego zaangażowania, wyścigu szczurów, żyjesz tylko pracą? Wyrabiasz nadgodziny, bo ktoś musi? Bo obiecali, że to już ostatni raz, choć w to nie wierzysz? W swojej kilkunastoletniej karierze miałem do czynienia z nadgodzinami raptem kilka razy. Owszem, pracowałem bardzo dużo, zdecydowanie za dużo. Ale pracowałem na własny rachunek, po pracy „kontraktowej", umówionej na określoną liczbę godzin.

Nie daj się wykorzystywać. Nie daj zabrać sobie czasu spędzonego z rodziną. Żadna firma nie ma prawa zawłaszczyć Twojego życia prywatnego wbrew Twojej woli. A że to wbrew Twojej woli (nawet jeśli uważasz inaczej), bardzo łatwo udowodnić:

> **Czy firma miałaby Ciebie na tyle godzin dziennie, codziennie, gdyby na Twoim koncie znajdowało się 10 milionów?**

No właśnie...

Skoro o milionach mowa: problemy z płatnościami są również na liście sygnałów alarmowych mogących uzasadniać poszukanie lepszego miejsca. Nawet jeśli lubisz pracę, to przecież nie wykonujesz jej charytatywnie: umowa jest umową i opóźnione albo obniżone wypłaty nie powinny się zdarzać. Nigdy.

⇨ *Pracowałem kiedyś przy świetnym systemie. Zarobki dobre, praca zdalna, swoboda olbrzymia. Projekt ciekawy, przyszłościowy, globalny. Nawet nieco rewolucyjny w swojej klasie.*

Klient też w porządku. Dwa lata współpracy upływały bardzo miło. Aż tu nagle jedna nieopłacona faktura. Potem druga. I kolejne. Uwierzyłem ciągłym zapewnieniom o tymczasowych problemach. Pracowałem dalej i płaciłem podatki za nowe wystawiane faktury.

W końcu kontakt się urwał. I zostałem z nieopłaconymi fakturami na prawie trzydzieści tysięcy złotych. Ba, byłem sporo na minusie, bo dodatkowo skarbówka zabrała swoje. Na kwartał przed ślubem.

Kasy nie odzyskałem do dziś, a w podróż poślubną pojechaliśmy w Góry Świętokrzyskie zamiast do Peru. I tak było fajnie, ale mam nauczkę do końca życia.

Takie scenariusze pewnie można by mnożyć, ale widać wyłaniający się wspólny obraz. Zmiana pracy jest zdecydowanie potrzebna, gdy zaczyna negatywnie wpływać na życie prywatne albo uniemożliwiać jego prowadzenie.

Jeśli coś Cię mocno uwiera, zawsze, ale to zawsze pamiętaj: nie musi tak być. Zdecydowanie rekomenduję rozmowy, konfrontacje, komunikację i rozwiązywanie problemów zamiast uciekania przed nimi, ale też nie ma co walczyć z wiatrakami. Czasami trzeba po prostu powiedzieć „dość", wziąć sprawy w swoje ręce i szukać innej drogi.

Najważniejsze: robić to w zgodzie ze sobą. Tak, by z uśmiechem i bez zniesmaczenia spoglądać rano w lustro.

⏎

Wstydliwa historia o odejściu z pracy

Było tak: „Co to za masakra, jakie warunki; nie mogę siedzieć tutaj ani miesiąca dłużej!". I faktycznie, różowo pod wieloma względami nie było. Jedyna słuszna decyzja po kilku godzinach dumania to: „odchodzę!".

Minęły dwa tygodnie od tego postanowienia.

Stał przed drzwiami gabinetu właściciela firmy z lewą dłonią zaciśniętą w pięść ze stresu, a prawą ułożoną w gest „zaraz będę pukał". Zdecydowane, pozamiatane, nie ma na co czekać. Teraz zostało najtrudniejsze: powiedzieć to szefowi oficjalnie, prosto w oczy.

The good

Stał tak z wyciągniętą ręką i nie mógł się ruszyć.

Do głowy powróciły wspomnienia sprzed zaledwie paru miesięcy. Przecież nie wszystko było złe...

Przecież kilka dni po rozpoczęciu pracy dostał bardzo fajne zadanie. Odpowiedzialne, trudne, ambitne. Nie wszędzie by tak zaufali świeżakowi! Poradził sobie. Słowa uznania, gratulacje, świetna atmosfera. Wtedy wydawało się jasne: po takim początku, ugruntowaniu pozycji w nowych warunkach i już na starcie pokazaniu swojej wartości nie może być inaczej: „to miejsce dla mnie!".

The bad

Ale, kurde, „ja chcę mieć swoje życie"! Warszawka Warszawką, ale przychodząc do pracy na ósmą, zwykle był ostatni. A opuszczając biuro po siedemnastej – po dziewięciu godzinach zapierdzielania! – mówił „do jutra!" do pokoju pełnego programistów. Po dziewięciu godzinach pracy czuł się jak leń, bo wszyscy pracowali dłużej.

Jeden z programistów zwierzył się, że jest w firmie od samego początku i zależy mu na sukcesie, więc siedzi, ile da radę. Drugi – że w domu nie ma co robić, więc spędza całe życie w biurze.

„Na łby poupadali, to nie moja bajka!".

I ta pieprzona obowiązkowa godzinna przerwa w trakcie dnia, podczas której i tak się przecież pracuje, bo co innego tam robić? Powód niby racjonalny: zagraniczny klient tego wymaga. I to było z góry wiadome, od samego początku, od pierwszej rozmowy. „Ale nie wiedziałem że będzie tak cholernie ciężko".

Nigdy więcej takich warunków. Osiem godzin dziennie to absolutne maksimum, przecież to i tak jest dużo. Bardzo dużo! „To dobra decyzja, spadam stąd!".

Za drzwiami słychać rozmowę. Szef gada przez telefon. Nie można tak teraz wparować i rzucić papierami.

Tup, tup, do swojego biurka; spróbujemy ponownie za godzinkę. Ale ta godzinka to męczarnia. Co powiedzieć? I jak powiedzieć?

The good

W poprzedniej pracy to dopiero było strasznie. Nudy przeokropne, zastój i kompletna stagnacja.

A tutaj? Ciągle coś nowego! I ta swoboda!

Młodziak jeszcze; gdzieś indziej pewnie by tylko stare bugi poprawiał albo testy dopisywał, wykonując rozkazy korpogenerałów. Fu!

Dzisiaj natomiast wymyślił sobie, że sprawdzi to nowe fajne rozwiązanie, o którym czytał jakiś czas temu na którymś blogu. I nikt się nie sprzeciwił! Ba, pojawiły się nawet słowa zachęty!

O tym marzył jeszcze na studiach. Tego szukał.

The bad

Ale co z tego?

Z domu trzeba wyjść przed siódmą – po to, żeby wrócić po dziewiętnastej. Jak robot, mechaniczny zwierzak, to chore!

Od poniedziałku do piątku tylko praca, tramwaj i sen. Wszelkie próby obejrzenia filmu wieczorem kończą się natychmiastowym zaśnięciem. A wyjście ze znajomymi na browara to tylko wstyd, bo po dwóch piwach czuje się jak po ośmiu.

Sobota przeznaczona na odespanie i dojście do siebie. Zostaje niedziela: jeden dzień w tygodniu, w którym można coś porobić. Pożyć. Jeden dzień w tygodniu… trochę mało.

Jak pozostali mogą działać w ten sposób? No tak, przecież nie mają życia poza firmą... „To dobra decyzja, spadam stąd".

W końcu wstaje z krzesła, dowleka się ponownie do drzwi gabinetu szefa. Ręka zawisła centymetr od drewna i... i się zablokowała.

Krótki przebłysk: to dokładnie takie samo uczucie jak pierwszy skok do basenu z wysokiej wieży! Stajesz przy krawędzi i nie możesz się ruszyć. Musisz zamknąć oczy, wyłączyć mózg i po prostu zrobić krok naprzód.

Zamknął oczy. Zapukał.

The great finale

– Proszę!

Szef jak zwykle uśmiechnięty. Nigdy nie powiedział złego słowa. Po prostu złoty człowiek. Wizjoner. Na właściwym miejscu.

– Słuchaj... Wiesz co, ja odchodzę...

Uffff, co za ulga! „Nareszcie mam to za sobą".

Ale wyraz twarzy bossa nie pozwala się w pełni zrelaksować. To jednak jeszcze nie koniec. Choć w sumie... czego się można było spodziewać? „OK, miło było, zatem nara"? Po raptem trzech miesiącach?

Uśmiech pozostał, ale jakiś taki krzywy. Mimowolnie w głowie pojawiło się skojarzenie: „to doskonała demonstracja powiedzenia, że kogoś zatkało". Ale wcale nie było mu do śmiechu. Czuł się gorzej niż przed wejściem do pokoju.

– Zaraz, zaraz, porozmawiajmy. Jesteśmy z ciebie bardzo zadowoleni. Ty też wydawałeś się zadowolony. O co chodzi? Wszystko możemy rozwiązać.

– No tak, bo ogólnie to jest fajnie, ale wiesz...

– Kasa? Nie ma sprawy, pogadajmy. O jaką kwotę chodzi?

Ups! To będzie naprawdę trudne. A rozmowa dopiero się zaczyna.

– Nie, nie chodzi o kasę. Wiesz, te dziewięć godzin dziennie to jednak strasznie dużo. Czuję, jakbym całe życie spędzał w biurze. Przez cały tydzień tylko i wyłącznie praca i sen. A że klient tego wymaga...

– Dobra, dobra, moment. Klient klientem, ale nie musi o wszystkim wiedzieć. Co, jeśli od następnego miesiąca ustalilibyśmy nieoficjalnie, że tę godzinną przerwę będziesz sobie robił po pracy? Naprawdę się tutaj przydajesz i byłoby głupio tracić cię z takiego powodu.

Oh fuck! „Czemu nikomu nie powiedziałem o tym wcześniej?". Wcześniej, zanim...

– Ale to nie tylko to. Ja prawie trzy godziny dziennie spędzam na dojazdach. To mnie zabija.

Rozpoczynamy desperacką walkę na argumenty.

– No tak, rozumiem, to ciężki problem... Ale może to dlatego, że przyjeżdżasz na ósmą? A gdybyś był w biurze trochę wcześniej albo trochę później? Ominiesz korki, a przy zrezygnowaniu z tej dodatkowej godziny będzie to dość łatwo zorganizować. Pewnie da się ten czas znacznie skrócić. Co ty na to?

„Zaraz zapadnę się pod ziemię". Głowa pęka, oczy pieką, brzuch zaczyna boleć, kolana miękną. Jak to możliwe, że nie rozpoczął tej rozmowy inaczej? I wcześniej? Wcześniej, zanim...

– Słuchaj, bardzo mi głupio, ale... Ale ja już podpisałem umowę w innej firmie. Już nie ma odwrotu.

Chciał teraz tylko jednego: spalić się ze wstydu.

Na pożegnanie usłyszał jeszcze:

– Gdybyś kiedyś zmienił zdanie, to nasze drzwi są dla ciebie zawsze otwarte.

Ale wiedział, że to nieprawda. Na ich miejscu nie chciałby siebie więcej widzieć.

To było ponad dziesięć lat temu. Uczucie dokładnie pamiętam do dziś. I wstydzę się także do dziś.

To między innymi dzięki temu doświadczeniu wielokrotnie przeczytasz w tej książce, że szczera komunikacja rozwiązuje problemy.

⏎

Spalone mosty

Może się wydawać, że IT jest tak bardzo rozległe! Robię, co chcę, i konsekwencje mnie nie dosięgną. W końcu jestem programistą! Każdy mnie pożąda, każdy o mnie zabiega! Kto mi coś może zrobić?

To droga donikąd. Na manowce.

Korzec maku?

Nasze środowisko jest o wiele mniejsze, niż Ci się wydaje. Jak się wejdzie dostatecznie głęboko, można odnieść wrażenie, że każdy każdego zna! Wspólny projekt, wspólny klient, wspólna konferencja...

Znasz teorię, że w sieci kontaktów między Tobą a dowolną osobą na świecie (włączając w to papieża czy prezydenta USA) jest maksymalnie sześć innych osób? Ta koncepcja nazywa się *six degrees of separation*.

> **W polskim IT spokojnie możesz założyć prawdziwość „two degrees of separation".**

Wiadomo, konfliktów i różnicy zdań nie da się uniknąć. Ale absolutnie kluczowe jest zrozumienie, że nikt nie jest niezastąpiony. I że ludzie wolą dłużej szukać partnera biznesowego (w tym programisty), niż pakować się na minę, na której ucierpiał ich znajomy.

Twoja przyszła firma zadzwoni do Twojej obecnej firmy i zapyta o referencje, rekomendacje. To normalna praktyka. Twój przyszły szef porozmawia z aktualnym. Nie warto stawiać się w sytuacji, w której usłyszy: „to zdolna bestia, ale ma paskudny charakter... tak naprawdę to cieszę się, że odchodzi, w jego miejsce wolę zatrudnić dwie słabsze technicznie, ale fajniejsze osoby". Postaw się w roli nowego potencjalnego pracodawcy. Co robisz?

Sam jestem dość często pytany o opinie na temat kandydatów i kandydatek. Przez lata działalności w środowisku spotkałem na swojej drodze setki programistów, których pamiętam z twarzy, imienia, nazwiska i miasta. Najczęściej

po takim pytaniu odpowiadam pozytywną rekomendacją (bo większość nas, w IT, na taką zasługuje). Nigdy jednak nie kłamię – to jedna z moich najważniejszych życiowych zasad – więc zdarzało mi się szczerze odradzić czyjąś kandydaturę na dane stanowisko. Jak inaczej mógłbym zrobić, skoro sam ze współpracy z tą osobą mam bardzo złe wspomnienia? Skoro ktoś zawalał każde przydzielone zadanie, nie wywiązywał się z obietnic, całkowicie olewał wcześniejsze ustalenia, a na koniec mówił: „to nie moja wina?".

Nie oceniam słuszności takich „rekomendacji szeptanych". To oczywiście zawsze perspektywa tylko jednej strony i nijak ma się do sprawiedliwości. Ale opisuję rzeczywistość, a nie idealny świat. To nie sąd. Po otrzymaniu złej opinii nie zawsze będziesz mieć szansę na wytłumaczenie się i wyjście obronną ręką.

Chichot losu

A los płata figle! I sytuacja może się odwrócić.

To oczywiście nie tyczy się tylko pracowników! W dokładnie taki sam sposób możesz dzisiaj być szefem, team leaderką, panem i władcą, a za kilka lat wylądować pod jednym ze swoich obecnych podwładnych. I nawet nie musisz zmieniać pracy, by się to stało.

⇨ *Słyszałem o przypadku, w którym nękany pracownik zmienił pracę z powodu nienawiści do swojego szefa. W nowej firmie szybko awansował. Za rok stara i nowa firma połączyły się w jedną. Niegdysiejszy pracownik został kierownikiem działu – szefem osoby będącej powodem jego odejścia z pracy zaledwie kilkanaście miesięcy wcześniej.*

Przyznasz, że to nieco głupia sytuacja?

Ploteczki

Świat się zmienia. Struktury zespołów się zmieniają. Napaskudzisz dzisiaj? Rachunek może zostać wystawiony i za kilka lat, gdy się tego najmniej

spodziewasz. Nawet w IT ploteczki i opinie rozchodzą się zastraszająco szybko. Sam tego doświadczam od lat!

Po jednym wystąpieniu na konferencji podchodzi do mnie człowiek i zagaduje: „Przyszedłem tutaj głównie po to, żeby cię poznać. Słyszałem, że jesteś strasznie arogancki buc, i chciałem to zweryfikować".

Jest gdzieś jakiś programistyczny Pudelek – czy co?

Nie twierdzę, że taka poczta pantoflowa to zjawisko dobre albo złe. Chcę Cię tylko uczulić, że świat IT jest mniejszy, niż Ci się może wydawać. Wymiana informacji następuje błyskawicznie. *D'oh!* Jesteśmy *Information Technology* w końcu!

Dbaj o relacje z ludźmi. Nawet z tymi (a może szczególnie z tymi?), z którymi się rozstajesz. Nawet z tymi, których nie chcesz więcej widzieć. W każdym środowisku sieć znajomości znaczy więcej niż pieniądze czy nawet kwalifikacje. *Deal with it*. Na ten temat można poczytać więcej w różnych ciekawych książkach. Moje rekomendacje znajdziesz na: http://zawodprogramista.pl/ksiazki.

Bardziej niż fair

Jeśli kiedykolwiek przyjdzie Ci do głowy nieczyste zagranie względem firmy, zastanów się: czy to naprawdę Ty? Czy może ten mały brudas, tkwiący w każdym z nas, łasy na okazję do znalezienia błyskotki na branżowym śmietniku?

⇨ *W ostatnich latach dwa razy zwalniałem się z pracy, którą uważałem za „najlepszą do tej pory". To takie „nieszczęście w szczęściu". Bardzo ciężkie doświadczenie.*

Wyznaję zasadę, że po roku zatrudnienia należy się podwyżka. Czy to będzie tylko wyrównanie inflacji, czy coś więcej – to już kwestia do dyskusji. Ale podwyżka musi być. Zawsze była.

Pierwszego czerwca 2015 roku wypadała moja druga rocznica zatrudnienia w firmie, w której pracowałem wtedy jako programista. Jak zawsze dwa tygodnie wcześniej napisałem do szefa, że czas się spotkać i porozmawiać.

Spotkanie ustalone, gadu-gadu, przechodzimy do rzeczy. Do ostatniej chwili nie wiedziałem, czy chcę dostać podwyżkę, czy odejść z pracy. Decyzję podjąłem chyba dzień przed spotkaniem.

Finalnie się zwolniłem. Mało tego: nie przyjąłem nawet podwyżki, którą mi na tym spotkaniu zaoferowano. A wiesz dlaczego? Bo dobre relacje są ważniejsze. Bo zawsze gram fair.

Mało tego! W umowie mieliśmy standardowy kwartalny okres wypowiedzenia. Ale mój ostateczny okres wypowiedzenia tyle nie wyniósł. Wiedziałem, że jestem kluczowym pracownikiem w firmie. Wiedziałem, że moje odejście to duży problem. Dlatego też zgodziłem się na pracę do końca roku.

Pierwszego czerwca złożyłem wypowiedzenie. Weszło w życie wraz z końcem roku. Po siedmiu miesiącach! Po takim czasie to niektórzy dziecko rodzą.

Oczywiście: nie było łatwo. Ostatnie miesiące były wypełnione ogromnym oczekiwaniem. Prawie odkreślałem kolejne kreski na ścianie. Nie dlatego, że na koniec dostawałem beznadziejne zadania. Wręcz przeciwnie! Po prostu bardzo chciałem już podążyć swoją własną ścieżką. Ale zaciskałem zęby i robiłem swoje. Bo tak trzeba.

Gdybym wtedy uciekł jak najszybciej, zupełnie nie przejmując się losami firmy, do dziś nie miałbym czystego sumienia. Uczciwa zmiana pracy co jakiś czas jest zdrowa. Jak najszybsze porzucenie firmy grającej nieczysto to Twój obowiązek. Ale skutki nieodpowiedzialnych decyzji mogą później doskwierać latami.

Jak ktoś rzuca w Ciebie chlebem, Ty rzuć w niego watą cukrową.

Bądź *fair* – to wystarczy. I pamiętaj: role mogą się zamienić.

Jak znaleźć fajną pracę?

Zalewają nas informacje o „rynku pracownika". Widzimy tysiące ogłoszeń o pracy dla programistów. I tak dalej.

Czy łatwo o fajną pracę?

Czy to wszystko prawda? Jeśli tak, to dlaczego jest tylu nieszczęśliwych programistów? Wypalonych?

Owszem, ofert pracy nie brakuje. Często nawet proponowane wynagrodzenie przewyższa to, które może zdobyć większość społeczeństwa. Warto jednak zadać sobie pytanie: skąd biorą się takie stawki i dlaczego mimo to stanowiska te bywają nieobsadzone całymi miesiącami? Brak specjalistów i specjalistek – to jasne! Ale jest ich mimo wszystko więcej niż zero.

Przez większość swojej kariery uczestniczyłem w rekrutacjach nie po to, by naprawdę zmienić pracę. Przeglądałem ogłoszenia i czasem wybierałem się na rozmowy, aby być na bieżąco, a nie z faktycznej potrzeby. Widziałem jednak, że gdyby taka potrzeba nastała, to... nie byłoby za wesoło. I to nie dlatego, że ja nie spełniałem wymagań. Odwrotnie!

Ogłoszeń mamy faktycznie tysiące. Ale wystarczy zacząć je przeglądać, by zdjąć różowe okulary. Często ta ogromna kasa wiąże się z zadaniami, których po prostu nikt nie chce robić. Zachwalane stanowisko wypala i wypruwa. Atmosfera w firmie zabija. Z kolei obowiązki można by rozłożyć na kilka osób i jeszcze nieco by zostało. Nie ma nic za darmo.

Nawet jeżeli teraz jesteś w komfortowej sytuacji i pracujesz dokładnie tam, gdzie chcesz, ze świetnymi ludźmi i przy ciekawym projekcie, wykonaj pewne ćwiczenie. Poświęć dosłownie pół godziny na poszukanie oferty programistycznego stanowiska spełniającego wszystkie Twoje wymagania. Odfiltruj magiczne słówka typu: „ninja", „piłkarzyki", „MultiSport", „dynamiczny zespół" i zobacz, co zostanie. Czy faktycznie masz tak ogromny wybór, jak Ci się wydawało? Dla mnie był to swego czasu szok.

Rzeczywistość nie jest taka tęczowa, jak myślałem. Jak się nam wmawia.

Taka konfrontacja może zmienić nasze zachowanie. Pozytywnie wpłynąć na postrzeganie dzisiejszego dnia. A jednocześnie zmotywować do dalszej, nieustannej pracy nad sobą. Bo... co dalej? Wcale nie musi być tak super!

Czy to dobrze, czy źle? To zależy. Przyda się wiadro zimnej wody wylane na pączka w maśle, jakim programista staje się czasami po kilku latach na ciepłej posadce.

Jak znaleźć fajną pracę?

Skakanie po kolejnych firmach na chybił trafił jest pewnym rozwiązaniem, lecz ma granice. Jak sądzisz, czy dziesiąty w ciągu dwóch lat pracodawca podejdzie do Ciebie z zaufaniem? Czy może oczekiwać lojalności? Albo chociaż zwrotu z inwestycji, jaką jest dołączenie Cię do zespołu? No właśnie...

Ja podjąłem ryzyko i swoją karierę rozpocząłem od trzech bardzo szybkich zmian pracy już w pierwszym roku po studiach. Później nie opierałem się na oficjalnych ogłoszeniach i możliwościach rekrutacji. Działałem inaczej: z polecenia.

Zmiana pracy ze słabej na słabszą to bułka z masłem! I bardzo demotywuje. Należy zatem odpowiednio przygotować się przed podjęciem decyzji o zmianie. Sprawdzić, co dokładnie na nas czeka. Obserwować środowisko podczas rozmowy kwalifikacyjnej. Zwrócić uwagę na biuro, nastroje innych pracowników.

A najlepiej... poznać tych pracowników! I otrzymać informacje z pierwszej ręki, z faktycznej linii frontu. Nie opierać się jedynie na komunikatach HR-owych i lukrze płynącym z oficjalnych ogłoszeń.

Nie musisz znać pracowników danej firmy już dzisiaj, osobiście. Ale dlaczego by ich nie poznać? Zainteresuj się społecznościami programistycznymi działającymi w Twoim mieście. Więcej na ten temat znajdziesz pod adresem: http://zawodprogramista.pl/spolecznosci. Jeżeli programiści z firmy X mają jeszcze siłę i chęci, by po pracy uczestniczyć w nieobowiązkowych programistycznych spotkaniach, to świadczy to o firmie raczej pozytywnie!

Tam, w nieoficjalnej atmosferze, można dowiedzieć się bardzo wiele. Ustrzec przed błędami popełnianymi przez innych albo wręcz przeciwnie: powtarzać udane kroki.

Meetupy nie przyciągają jednak wszystkich zadowolonych programistów. Warto rozejrzeć się także na własną rękę. Portale, takie jak LinkedIn czy GoldenLine, często dostarczą Ci dane do dalszych poszukiwań. Dlaczego by nie zaprosić kogoś z aktualnych pracowników danej firmy na niezobowiązującą kawę? Podpytać o atmosferę w pracy, projekty, możliwości rozwoju; o to, dlaczego pracuje właśnie tam! Abstrahując już od korzystnego wpływu takiej rozmowy na karierę, może zawrzesz nową znajomość!

Polecam kontakt bezpośredni, osobisty, z żywym człowiekiem. Rozumiem jednak, że może to być problem. Ja sam, jeszcze kilka lat temu, miałbym z tym duży kłopot. Wtedy wstydziłem się nawet zwrócić uwagę pani w sklepie, że mi źle wydała resztę, a co dopiero pójść gdzieś z kimś, kogo nie znam!

Co zatem zostaje? Społeczności *online*!

Możesz podyskutować na polskim programistycznym czacie (http://zawodprogramista.pl/slack) i tam podpytać o polecane firmy. Zapraszam Cię też na moją grupę na Facebooku, gdzie znajdziesz tysiące przyjaznych programistów i programistek. Wpadaj na: http://zawodprogramista.pl/facebook.

Niektóre firmy bardzo aktywnie działają w programistycznych społecznościach. Organizują własne meetupy i konferencje lub angażują się w działania oddolne. Można pokusić się o przypuszczenie, że taki pracodawca rozumie potrzebę rozwoju i wartości płynące z bycia częścią *dev-community*. Oczywiście, prawie zawsze za takimi działaniami kryją się po prostu potrzeby rekruterskie, ale to zdecydowanie dobry sposób, by zaistnieć w świadomości lokalnych programistów i znaleźć się na ich radarze. Tak jak programista może się wybić ponad konkurencję, tak firma może zrobić to samo. Wsparcie wszelakich inicjatyw wzbogacających programistyczne życie jest krokiem w dobrą stronę.

Krótko mówiąc: przyszły pracodawca prawdopodobnie zrobi „wywiad środowiskowy" na Twój temat. Odwdzięcz się i zrób to samo przed podpisaniem

umowy! Praca w poleconym miejscu ma dla Ciebie większe szanse na strzał w dziesiątkę niż losowe ogłoszenie w necie.

⏎

Na co zwracać uwagę, poszukując pracy?

Ustaliliśmy już, że o rozwijającą pracę z interesującymi projektami wcale nie jest tak łatwo, jak by się mogło wydawać. Z niektórymi aspektami rzeczywistości można próbować walczyć, ale z innymi lepiej się pogodzić.

Radzę pogodzić się z tym, że praca programisty nie jest nieustannym ekstatycznym rollercoasterem z ambitnymi wyzwaniami „eksperckimi". Często jest to po prostu żmudna orka. Przecież nawet kierowca rajdowy – jak dla mnie wykonujący najlepszą profesję we wszechświecie – nie śmiga po torze przez czterdzieści godzin tygodniowo, ale ma też inne smutne obowiązki.

Warto więc spojrzeć także na inne aspekty działalności firmy, z którą być może zwiążesz się na najbliższe lata. Takie rozpoznanie w czasach LinkedIna i Facebooka nie będzie trudne. Znajdziesz kogoś, kto powie Ci, jak jest.

Podzieliłbym cechy godne uwagi na dwie grupy.

Tech

Dla mnie pytanie numer jeden to: czy piszecie testy jednostkowe? Wbrew pozorom odpowiedź powie Ci o firmie bardzo wiele! Firma praktykująca testy jednostkowe inwestuje w jakość. Nawet jeśli pisane testy nie są najwyższej klasy, firma poświęca czas na ulepszanie kompetencji oraz tworzonych produktów. Myśli bardziej perspektywicznie niż tylko o najbliższym deadlinie. Oczywiście testy nie gwarantują wspaniałego kodu, ale zmniejszają niebezpieczeństwo trafienia do programistycznej kopalni, gdzie liczy się tylko liczba „kodogodzin" – jak dużo kodu człowiek jest w stanie naklepać w ciągu godziny.

> **Kto wygeneruje najwięcej, wygrywa Amulet Nietykalności!**
> **W najbliższym sprincie nie musi poprawiać bugów,**
> **tylko spokojnie dalej sadzi nowe!**

Koniecznie dowiedz się, jakiego systemu kontroli wersji używa Twój przyszły zespół. Ciężko mi było w to uwierzyć, ale nawet w 2017 roku niektórzy dzielą się kodem poprzez archiwum *zip*! Nie chcesz znaleźć się w takim miejscu. Ja pewnie miałbym opory przed dołączeniem do jakiejkolwiek firmy nieużywającej Gita. Po prostu zbyt wiele nerwów straciłem, pracując z SVN-em i TFS-em.

⇨ *Jedna z firm, z którymi miałem styczność, używała oprogramowania o nazwie Sonar. To software badający między innymi jakość kodu. Można w nim ustawiać reguły, które kod powinien spełniać. Każda złamana reguła otrzymuje wartość w dolarach: system szacuje, jak kosztowne będzie jej poprawienie.*

Pewnego razu jeden z managerów wpadł na pomysł, że pięćdziesiąt procent kodu powinny stanowić komentarze. To sprawi, że praca z kodem będzie przyjemniejsza w przyszłości. Idiotyzm – ale niech tam!

Oczywiście wprowadzenie tej reguły spowodowało alarm w Sonarze i pokolorowanie wszelkich raportów na czerwono. System wycenił koszt zastosowania nowej reguły na dziesiątki tysięcy dolarów! To tyle, jeśli chodzi o sztuczną inteligencję.

Do walki stanęła inteligencja ludzka. Zespół programistyczny znalazł cudowne narzędzie: generator komentarzy. Ów generator potrafił sprawdzić stosunek komentarzy do kodu, ustawiony w Sonarze, a następnie automatycznie dodawał odpowiednią ich liczbę. Treść komentarzy brzmiała jakoś tak: „// this comment was generated to comply with Sonar rule". Takich komentarzy pojawiły się tysiące.

Sonar przeanalizował nowy kod. Raporty się zazieleniły. A program do generowania komentarzy został przez managera okrzyknięty odkryciem tygodnia, bo zaoszczędził dziesiątki tysięcy dolarów...

Takie historie chodzą po branży. I nie robią firmie dobrej reklamy.

Warto zapytać też o *code review* i *pair programming*. Nie trzeba wszystkiego opracowywać w parach ani przeglądać każdej linijki kodu, bo to strata czasu. Ale jeżeli zespół zna te praktyki i korzysta z nich chociaż wybiórczo, jest to dobry znak. Taki sam jak w przypadku testów: ktoś tu dba o jakość i rozwój, a nie tylko o tony nowego kompilowalnego tekstu.

Socjal

Nie samym kodem człowiek żyje. Zapytaj siebie: jak wyobrażasz sobie idealną pracę? I zweryfikuj, czy masz szansę do takiej trafić.

Dla mnie – jak już doskonale wiesz – absolutnym numerem jeden w kwestiach nietechnicznych, w kulturze pracy, jest brak nadgodzin. To kwestia szacunku do mojego czasu i życia pozazawodowego. Zawsze działałem bardzo dużo po godzinach, ale były to inicjatywy własne. Z firmą umawiam się na X godzin w tygodniu i się z tego wywiązuję. X, nie więcej.

Warto też dowiedzieć się, jak wygląda sprawa deadline'ów. Czy zespół często pracuje w trybie gaszenia pożaru, czy raczej można liczyć na spokój i stabilny rozwój projektów? Sam cenię sobie bezstresowość i nie znoszę zbytniego ciśnienia ze strony terminów. To bardzo źle wpływa na jakość kodu, komfort pracy i morale zespołu.

⇨ *W jednej z firm szefem mojego zespołu był taki „stary" wówczas dla mnie koleś. Trzydziestopięciolatek! W pracy był przez prawie cały czas. A potem wracał do domu, do rodziny… by dalej pracować.*

W wieku trzydziestu pięciu lat wyglądał na czterdzieści pięć, a czuł się na pięćdziesiąt pięć. Wiedziałem, że nigdy nie pozwolę żadnej firmie doprowadzić się do takiego stanu.

Istotną kwestią jest wsparcie rozwoju pracowników przez firmę. Czy możesz liczyć na dofinansowanie szkoleń albo wyjazdów na konferencje? A może nawet

na specjalny program pomagający zostać programistycznym prelegentem? Włączenie się w życie społeczności w czasie pracy? To coś pięknego! Ale... dość spoilerowania. Te kwestie rozwijam w jednym z następnych rozdziałów.

Zauważ, że ani razu nie padło pozytywne *wow* na hasło „piłkarzyki w biurze".

Dlaczego piłkarzyki są niebezpieczne?

Piłkarzyki, Xboxy, rzutki, kręgle... Może kryje się za tym coś więcej?

Jestem przeciwnikiem robienia placu zabaw z biura. Wiem bowiem, jaki to ma wpływ na ludzi takich jak ja, czyli chcących mieć życie poza pracą.

Może nie chodzi tylko o to, aby nam było dobrze? Może kryje się za tym coś więcej? Wszyscy znamy całkiem popularny slogan: „biuro Twoim drugim domem!". Oooooo, moment, chwila! Z tym mam już duży problem.

Mój „normalny" dom jest moim domem zarówno pierwszym, jak i drugim czy trzecim. Biuro to miejsce, do którego przychodzę pracować. I opuszczam je po zakończeniu pracy. Koniec.

Nie mam nic przeciwko dobrej, przyjaznej atmosferze w biurze. Ale bez przesady. Podczas rozmów z oferentami bądź beneficjentami opisywanych uciech zadawałem czasami pytanie: czy jeśli przez króciutką, przyjemniutką godzinkę dziennie będę korzystał z tych dogodności, to muszę zostawać o godzinę dłużej w pracy? W większości przypadków odpowiedź brzmi: „tak". Więc mam o kilkadziesiąt minut mniej czasu na życie poza pracą.

To mi wystarczy – dalej nawet nie drążę tematu.

Pojawiają się argumenty, że przerwa na piłkarzyki pozwala programiście oderwać się od kodu, zebrać myśli, pomyśleć nad problemem... I to na pewno prawda. Ale do piłkarzyków potrzeba więcej osób. Wszyscy przestają robić to, co robili, by jeden pomyślał. Przerwy w pracy oczywiście są potrzebne. Ale niekoniecznie angażujące nie wiadomo ilu współpracowników.

Mój stosunek do rozporządzania moim czasem przez inne osoby jest wręcz nieprzyzwoicie agresywny. Wielokrotnie mówiłem, że czas to nasza najcenniejsza waluta. Wiesz, jak ja interpretuję takie zagrywki?

Doba ma przecież tylko dwadzieścia cztery godziny. Sporo z nich przeznaczam na sen. Jeśli tylko każdego dnia będę spędzał w biurze godzinkę więcej, to godzinka mniej dziennie zostanie mi na „pierdoły" niezwiązane z pracą. Coraz bardziej będę od tej konkretnej pracy uzależniony.

Skoro pogram w piłkarzyki z ludźmi w biurze, to tego dnia raczej nie pogram z innymi znajomymi. Bo nie będę miał na to czasu!

Tak bardzo przyzwyczaję się do postukania w Fifę w przerwie w pracy, że nie przejdę do innej firmy, która nie ma u siebie Xboxa. Do moich wymagań stawianych pracodawcy dojdą kolejne, zupełnie bezsensowne. Ograniczę sobie pole manewru.

Do ośmiu godzin pracy i dwóch godzin na dojazdy w obie strony dorzucam kolejną godzinę na lunch i jeszcze jedną na rozrywkę. Wychodzi mi *de facto* nie osiem godzin, tylko dwanaście. Chore.

Wielokrotnie słyszałem żale w stylu: „Wychodzę rano do pracy i już nie mogę się doczekać, aż wrócę do domu, żeby malucha wykąpać i położyć spać! Niestety widzę go tylko przez pół godzinki dziennie". To jest oczywiście kwestia priorytetów i nic mi do tego, ale może gdyby uciąć z „czasu pracy" to, co nie jest pracą, rzeczywistość wyglądałaby trochę inaczej? Nie jesteśmy prezydentami USA, żeby musieć być w obiegu przez całą dobę.

Ale to nie wszystko! Pojawia się jeszcze niewidoczne na pierwszy rzut oka niebezpieczeństwo: społecznościowy przymus.

Peer pressure: „Nie pograsz pół godzinki w ping-ponga? Co za smutas!"

Niby nikt nie „każe", ale jeśli nie ulegniesz, zostaniesz biurowym zamulakiem. Ja takim byłem. Nudziarzem, co nie lubi się bawić. A ja po prostu nie lubiłem nieoficjalnych nadgodzin! Prawie wszyscy inni lubili… albo im się tak wydawało.

Akurat mnie taka opinia niewiele obchodziła. Przynajmniej wszystko było jasne. Ale możliwe, że ktoś spędza w biurze więcej czasu, niż musi, bo tak wypada.

Nawet jeśli masz takie wygody i Cię to wielce uszczęśliwia, zadaj sobie pytanie: czy na pewno za każdym razem poświęcasz ten dodatkowy czas w pełni

kontrolowanie? Świadomie? Czy robisz to, bo tego chcesz, a nie tylko dlatego, że to siła przyzwyczajenia? Albo że szef przy Xboxie jest taki przyjacielski, i jak z nim pograsz, to łatwiej będzie o podwyżkę? Bo w końcu wtedy to już kumpel!

Jeśli wszystko jest pod kontrolą, to świetnie. Ale jeśli nie – czy na pewno powinno tak być? Każda godzina jest na wagę złota. Czy nie ma w życiu niczego fajniejszego, na co można wykorzystać czas spędzony w chill roomie?

Na koniec: nie twierdzę, że firma ma złowrogie zamiary. Może robi to, bo taki jest rynek, bo konkurencja tak działa? Ale to nie ma znaczenia. Nie wszystko złoto, co się świeci.

Ustal swoje priorytety

Dowiedz się (od siebie!), co jest dla Ciebie ważne. Zrób to świadomie.

U mnie preferencje kształtowały się latami, a efekty opisałem powyżej. Wcale nie musi to oznaczać, że jest to jedyna słuszna droga. Choć jestem w stanie długo się spierać, broniąc swojego zdania.

⏎

Praca zdalna

Charakterystycznym trybem pracy w IT jest praca zdalna. Że niby tak można, że to fajne, wygodne i wyjątkowe. Czy to prawda?

Owszem, praca zdalna jest możliwa. I bywa super. Ale w pewnych okolicznościach!

Miód i dziegieć

Dzięki pracy zdalnej nie tracisz czasu na dojazdy. Nie wydajesz pieniędzy na benzynę czy bilety. Nie stoisz w korkach. Nie musisz zrywać się skoro świt, by o dziewiątej rano odbić kartę w zakładzie. Ba, nie musisz się nawet myć!

No właśnie... wszędzie jest łyżka dziegciu. Łatwo jest się zapuścić, w imię wolności i oszczędności czasu. To nie żart!

Dzięki pracy zdalnej nie musisz opuszczać przyjaznego środowiska. Twoje kwiatki będą podlane w odpowiednich godzinach. Zrobisz sobie pyszny zdrowy lunch, a nie ciągle tylko ta pizza i kebaby w biurze. Twojemu kotkowi nie będzie smutno, bo ciągle będzie miał się kto z nim bawić. W końcu jesteś w domu! Więc w przerwie ogarniesz odkurzanie i skoczysz na pocztę, żeby popołudnie mieć całkowicie wolne. Ups, a kiedy pracować? Znowu przekroczyliśmy cienką granicę między zaletami a wadami pracy zdalnej. Masz w sobie na tyle dużo samodyscypliny, żeby faktycznie skupić się na pracy?

Dzięki pracy zdalnej nie musisz się ściskać w niewygodnym, zatłoczonym biurze. Nikt nie przeszkadza, nie pokazuje ciągle na telefonie zdjęć tych swoich dzieci. Ile można, co nie?! Ale... komu pokażesz zdjęcia swoich? Czasem fajnie jest zamienić z kimś kilka słów.

No i gdzie się ulokujesz? Praca bez odpowiedniego stanowiska, na chybotliwym kuchennym stołku przy malutkim skrawku blatu, szybko przestanie być takim rajem. A i plecy zaczną doskwierać, i nadgarstki, i kolana... Oj, tak!

Przez wiele miesięcy pracowałem z domu, nie mając „domowego biura". Na dłuższą metę nie jest to fajne, wcale a wcale. A nawet teraz moje „biuro" mieści się po prostu w salonie, ponieważ w mieszkaniu brakuje jednego pokoju – tego właśnie dla mnie. Dlatego też praca w obecności domowników jest bardzo utrudniona.

⇨ *Na konferencji wymieniałem się kiedyś spostrzeżeniami z jednym z uczestników na temat wyzwań stojących przed praktykami pracy zdalnej. Jednym z problemów okazało się przekazanie najbliższym, że się jest w pracy, mimo przebywania w domu.*

U mnie ten problem zwykle nie występuje. Żona pracuje poza domem, a Córka chodzi do przedszkola, więc w dzień mam całe mieszkanie dla siebie.

U niego było inaczej. Wymyślił ciekawy sposób, który się sprawdził. W godzinach pracy zawsze wkładał krawat. Choćby miał na sobie dresy albo piżamę, krawat oznaczał: „pracuję, nie przeszkadzać". Działało. Bardzo sprytny trik.

Niewygodne, zatłoczone biuro nagle zaczyna mieć pewne niedostrzegane wcześniej zalety. Oprócz samej powierzchni potrzebne są też środki na jej urządzenie! No, chyba że zdecydujesz się na pracę zdalną, ale poza własnym domem. „Kawiarniany nomad" to całkiem przyjemna profesja. Tyle że... w pełnym rozrachunku może wyjść i drożej, i mniej komfortowo niż w normalnym biurze.

Nie zrozum mnie źle; ja osobiście pracę zdalną uwielbiam. Większość swojej kariery spędziłem we własnym wygodnym domu, kawiarni czy parku – i niezmiennie to sobie chwalę. Trzeba jednak mieć świadomość nie tylko zalet, ale również wad i dodatkowych zobowiązań.

Zdalna < asynchroniczna

Kluczowym pojęciem może okazać się nie tyle praca zdalna, ile praca asynchroniczna. To dzięki niej możliwe są elastyczne godziny pracy. To asynchroniczność daje wolność i umożliwia „zdalność".

Asynchroniczna komunikacja (i praca) polega tym, że więcej niż jedna osoba nie musi znajdować się jednocześnie w określonym miejscu i czasie. Zainteresowani są w stanie współpracować, nie oczekując na nikogo, nie wymuszając na otoczeniu żadnych konkretnych zachowań. Przy zastosowaniu odpowiednich praktyk i narzędzi praca przebiega płynnie, niezależnie od tego, gdzie (i kiedy) znajdują się wszystkie zaangażowane osoby. To życie bez spotkań, bez telefonów. Bez marnowania czasu.

Więcej na ten temat dowiesz się pod linkiem podanym na końcu tego rozdziału – między innymi z książki wydanej przez firmę Arkency oraz z mojego wywiadu z Andrzejem Krzywdą, jej założycielem.

Prerequisites

Co istotne: udana praca zdalna wymaga, aby cała firma (albo chociaż cały zespół) rozumiała taką kulturę i się do niej stosowała. Jeśli jesteś jedynym „zdalnym" członkiem zespołu, to na pewno będziesz jednostką odizolowaną. Po prostu nie może być inaczej. Nie da się w równym stopniu zintegrować na odległość jak osoby spędzające ze sobą fizycznie przynajmniej jedną trzecią życia.

Gwałtowne przeskoczenie z trybu „normalnego" w tryb „zdalny" wiąże się z wieloma trudnościami i bez odpowiedniego przygotowania jest niemal skazane na porażkę. Trzeba znaleźć i opanować nowe narzędzia. Wprowadzić pewne zasady i regulacje, które w normalnej sytuacji są po prostu zbędne. Upewnić się, że każdy rozumie specyfikę takiego sposobu pracy i komunikacji.

I mamy jeszcze jeden aspekt: wzajemną naukę. Dopóki nie doczekamy się technologii wprowadzającej hologramy pod strzechy, nie będziemy w stanie w pełni odwzorować walorów edukacyjnych pracy w jednym biurze. Wraz ze wzrostem dojrzałości, samodzielności i doświadczenia członków zespołu może to mieć coraz mniejsze znaczenie. Ale, szczególnie na początku drogi, bycie częścią teamu zlokalizowanego we wspólnym miejscu zdecydowanie pomaga osiągać kolejne poziomy programistycznego wtajemniczenia. Obserwacja nawyków, słuchanie rozmów i analiz problemów, wspólne zastanawianie się nad możliwymi rozwiązaniami – to wszystko jest bezcenne. Telekonferencje są tylko namiastką.

Dlatego też, jeśli dopiero zaczynasz swoją drogę jako programista czy programistka, nie daj się oczarować pracy zdalnej. To faktycznie jest świetna opcja, ale ma swój koszt. Ten koszt to prędkość i jakość Twojego rozwoju. Przyspieszenie Twojej edukacji, czas wskoczenia na wyższe poziomy. „Zdalność" może po prostu nie być tego warta. A co najważniejsze: ona nie ucieknie.

> **Im więcej masz doświadczenia, im bardziej wyróżniasz się na tle innych kandydatów, tym łatwiej wynegocjujesz w przyszłości możliwość spróbowania pracy zdalnej.**

I wcale nie musi Ci się to spodobać.

Więcej materiałów na temat pracy zdalnej znajdziesz pod adresem: http://zawodprogramista.pl/praca-zdalna. Zamiast pompować tę książkę transkrypcjami wywiadów, umieściłem tam linki do dodatkowych treści o tej tematyce.

5 Programistyczny rozwój

Możliwości awansu w IT

W rozdziałach poświęconych początkowym etapom kariery poruszyliśmy temat dewaluacji nazw stanowisk w IT. Tak naprawdę po dziesięciu latach możesz być wszystkim i robić wszystko. Jeść chleb z niejednego pieca, a potem położyć się i czekać na emeryturę.

Nie każdy potrzebuje ciągłych zmian, nowych wyzwań. Perspektyw na „coś innego". Zazdroszczę, naprawdę! Ja jednak w pewnym momencie doszedłem do miejsca, w którym kompletnie nie wiedziałem, „co dalej". A raczej odwrotnie: dokładnie wiedziałem, co dalej, i to było jeszcze gorsze! Brak perspektyw na coś nowego, świeżego. Dziesięć lat temu byłbym pewny, że to moje marzenie: wiedzieć dokładnie, gdzie jestem i gdzie będę. Bez niespodzianek. Ale to się zmieniło i stało się jednym z powodów poszukiwania przeze mnie alternatyw.

Standardowe początki

Standardowa ścieżka programistycznej kariery wygląda – na początku – prawie zawsze tak samo. Najtrudniejszy jest etap między „chcę być programistą" a „jeeeeju! mam pierwszą pracę!". Potem już z górki.

Po kilku miesiącach lub latach wskakujemy na stanowisko mida. To wtedy eksplorujemy, badamy, ryzykujemy. To właśnie regularów sprzedaje się w internecie na kilogramy. „Mam 240kg .NET developera *bez kości* do wynajęcia!". Bawimy się rynkiem. Żonglujemy technologiami.

Następnie *señor developer*.

Jeśli trzeci krzyżyk – starość! – się nieuchronnie zbliża, albo już nawet minął (to ludzie tyle żyją??), to... czyż nie tak wyglądała Twoja ścieżka?

Rozdroże

Robiłem już analogie do innych zawodów, a konkretnie do prawników czy lekarzy. Kim staje się prawnik po -nastu latach w zawodzie? Lepszym prawnikiem! A lekarz? Lepszym lekarzem!

Natomiast programista nie zawsze będzie „lepszym programistą", bo po jakimś czasie staje przed wyborem. Z jednej strony: „och, jak ja bardzo chcę programować!"; z drugiej: sufit. Dalsza chęć czerpania czystej radości z programowania często oznacza stagnację zawodową, zastój kariery.

Czego się od nas oczekuje? Zdecydowania: albo idę dalej w programerkę (czytaj: stoję w miejscu – bo nie ma już nic nad seniorem!), albo muszę się „zeskalować". Czyli...

> **Tracimy świetnych programistów,**
> **robiąc z nich beznadziejnych managerów.**

Stosunkowo wcześnie na programistycznej ścieżce musimy podjąć trudną decyzję. IDE, kodzik, zera i jedynki? Czy spotkania, telefony, excele, maile, raporty i szkolenia coachingowe?

Co robić?

Prawie wszyscy programiści czują opór przed wejściem w rolę team lidera. Bo to już ociera się o znielubiany „menedżment". Fuuuuj; nie po to od lat koduję, żeby teraz takie coś! Ale wiesz co? To naturalna ewolucja. Nie podchodź do tego w ten sposób.

Radzę pójść następującym tokiem rozumowania. Sam nim podążyłem i oszczędziło mi to wielu frustracji. Zauważasz u siebie smykałkę do kodowania, prawda? Lubisz to robić, robisz to dobrze. Wykonujesz zadania lepiej i szybciej, niż się od Ciebie oczekuje. Ja w pewnym momencie swojej kariery mogłem założyć się o całą pensję, że przy fajnym projekcie jestem w stanie zastąpić troje programistów. Troje! Bo zawsze moim mottem było (i zresztą jest nadal):

> **Robić dużo. Robić szybko. Robić dobrze.**

I tak działałem. Tak działam. Zostałem jednak poproszony o próbę „zeskalowania się". I zrozumiałem argumentację. Dlaczego ograniczać się od trojga programistów w jednej – mojej – osobie, skoro mogę spróbować poprowadzić zespół

złożony z sześciu czy dziesięciu osób? Dbając o rozwój, dobre praktyki, jakość i zadowolenie. Początkowo potraktowałem to jako podstęp. „Ooo, menedżment chce mnie wciągnąć w excele, odebrać mi radość życia!". Ale z drugiej strony... faktycznie, mieli rację. Więc dlaczego by nie spróbować?

Podstęp i przeszkodę zmieniłem – w swojej głowie! – w wyzwanie i możliwość rozwoju. (Przyrzekam na zera i jedynki: te coachowe wstawki piszą się same! Przestań widzieć 6 i zacznij widzieć 9, etc...).

Spróbowałem. Bardzo wiele się nauczyłem. Zespół działał dobrze.

Po kilku miesiącach wróciłem do roli „tylko" programisty. Bo programowanie na pół etatu – drugie pół poświęcając zespołowi – to było dla mnie za mało.

„Po prostu koduj"?

Po raz kolejny powtórzymy jedną z najważniejszych myśli w tej książce: decyzje są odwracalne. Nie mów „nie". Najwyżej się wycofasz – z bagażem nowych doświadczeń. Wiem, że się momentami powtarzam, ale to niezmiernie istotne:

> **Przygotuj się na ten tragiczny moment, w którym poczujesz, że to koniec.**

Wyobraź sobie, że właśnie z Twoich palców wypłynęła ostatnia linijka kodu. Nie ma w Tobie już ani jednego zera, ani jednej jedynki.

Miejmy nadzieję, że ta chwila nigdy nie nadejdzie. Ale... ja byłem pewny na sto procent, że mnie to nie spotka. Nigdy. Dałbym sobie rękę uciąć. I – cytując klasyka – bym teraz nie miał ręki.

Nie wiem, jakie są Twoje obserwacje, ale ja spotykam się z tym całkiem często. Po kilku, kilkunastu latach kodowania coś w człowieku pęka i aż chce się zmiany. Zaskakująco niewielu znajomych ze studiów jest teraz „po prostu programistami". Po dziesięciu latach byłem jednym z nielicznych, którzy na co dzień uprawiali kodowanie przez większość czasu.

Dlaczego? To temat bardziej na wywiad z wykwalifikowanym psychologiem niż na rozważania w tym miejscu. Ale im więcej doświadczeń zdobędziesz po drodze, tym więcej drzwi otworzy się przed Tobą.

A skoro tu zabrnęliśmy, mam dla Ciebie specjalną stronę. Zapamiętaj, że w gorszych chwilach możesz wpaść na: http://zawodprogramista.pl/psycho.

Kaganek

Sytuacji opisanej w poprzednim punkcie nie przeskoczysz. Zawodowo masz do wyboru dwie drogi: kod lub biznes. Na dłuższą metę nie połączysz jednego z drugim, niezależnie od tego, jaką formę działalności przyjmiesz.

Warto jednak otworzyć oczy nieco szerzej. Życie nie kończy się na pracy – czy to dla kogoś, czy to dla siebie. Zdobywane latami umiejętności można wykorzystać na bardzo różne sposoby. Przed Tobą (i przede mną zresztą też) niepewna przyszłość. Nie wiadomo, co stanie się z naszą branżą za pięć lat. Nie wiadomo, co stanie się z naszym zaangażowaniem i pasją nawet za rok.

Wiadomo natomiast, co daje ogromną radość i pozwala odepchnąć czarne myśli. Zbudować wewnętrzną pewność, że „będzie dobrze", że co by się nie działo – powstaje dobro. Że tworzy się realna wartość, pomagająca innym w trudnych chwilach. Tym źródłem radości jest dzielenie się własnym doświadczeniem i własną wiedzą. Jaki ma to związek z niniejszym rozdziałem? Otóż: rola lidera w pracy może znakomicie przygotować Cię do dalszej drogi, wybiegającej poza codzienne obowiązki zawodowe.

Programowanie jest fascynujące, ale na dłuższą metę prawdopodobnie nie zmienisz nim świata (no chyba że...). Jeśli nie masz takich ambicji, możemy zmodyfikować definicję „zmieniania świata". Niech to będzie pozytywny wpływ na życie jednej konkretnej osoby, nie anonimowego „klienta" czy „użytkownika".

Wyobraź sobie, że sześciolatek może odkryć nasz magiczny świat i świadomie stawiać swoje pierwsze kroki!

Wyobraź sobie, że sfrustrowany ojciec rodziny może odkryć alternatywę dla znienawidzonej roboty i zdobyć się na odwagę, by spróbować szczęścia gdzieś indziej.

Wyobraź sobie, że młoda mama na macierzyńskim może zobaczyć światełko w tunelu, błyskające znad garów i mopa.

Ktoś odkryje nowy sens życia. Dzięki Tobie.

To jest możliwe. Wszystko to widziałem, i jest to widok godny pozazdroszczenia. A Ty możesz być istotną częścią tego procesu. Wykorzystać swoje umiejętności do czynienia... cudów. Pozwól, by ktoś kontynuował Twój ślad. Niech Twoja praca ma głębszy sens.

Nie przesadzam.

Jak negocjować podwyżkę?

O pieniądzach w polskim IT napisano już bardzo wiele. Większość to niestety wierutne bzdury. Można traktować pieniądze jako coś, co psuje cywilizację. Albo udawać, że pieniądze szczęścia nie dają. Warto się zastanowić jednak, co by było, gdyby spadło na Ciebie sto milionów. Czy nadal codziennie rano witałoby Cię to samo biuro? Czy aktualny projekt w danej technologii wciąż byłby spełnieniem Twoich marzeń? Czy koledzy w pracy, szef, klient byliby tak samo perfekcyjni?

A może jednak w takiej sytuacji serce pchnęłoby Cię gdzieś indziej, gdzieś dalej? No właśnie... To jest praca. Za pracę należy się wypłata. Im większa, tym lepiej. Jak ją zwiększyć, jednocześnie będąc *fair*?

Zdobądź minimum wiedzy

Negocjacja to umiejętność. Istnieje zawód: negocjator. Ba, nagrano o nim nawet niejeden kasowy film! Widzisz tutaj analogię do programowania? To też umiejętność! Istnieje zawód: programista. Ba, napisano nawet o nim bestsellerową książkę! Właśnie ją czytasz, bajdełej.

Skąd programista wie, jak programować? Nauczył się! A negocjator – skąd wie, jak negocjować? Też się nauczył! Naucz się i Ty. Choć trochę. Zdobądź minimum

wiedzy na ten temat. Posłuchaj podcastów – jest tego sporo. Przeczytaj książkę lub dwie. Na początek warto zerknąć do klasyków; mogą zresztą okazać się wystarczające. Polecam:

- Roger Fisher, William Ury, „Getting to Yes";
- William Ury, „Getting Past No" (kontynuacja);
- Roger Dawson, „Secrets of Power Negotiating".

Używane egzemplarze, nawet w oryginale, kupisz za grosze na aukcjach. Sam anglojęzyczne książki kupuję na eBayu.

To naprawdę otwiera oczy. Dekamufluje manipulacje, tłumaczy przyjacielskie poklepywania po ramieniu i uczula na śliskie – ale bardzo popularne – praktyki. Może wydawać się, że to zwykła, banalna rozmowa z szefem. „Hej, słuchaj, chcę więcej kasy!" – „OK, masz" albo „nie teraz, spadaj". Ale tak nie jest. To proces.

Ja zacząłem interesować się tym dość późno. Zbyt późno. I dopiero po latach zobaczyłem, ile w swojej karierze popełniłem błędów. Mówiąc wprost: ile pieniędzy straciłem. Bardzo dużo.

Krótko po zdobyciu nowej wiedzy postanowiłem sprawdzić ją w praktyce. Efekt? W jednej pracy wynegocjowałem czterokrotną podwyżkę. Tak, moje wynagrodzenie wzrosło czterokrotnie! Przy jednoczesnym skróceniu czasu pracy, z miesiąca na miesiąc. W innym przedsięwzięciu z kolei dostałem pięćdziesiąt procent stałej premii.

To działa. Tylko trzeba poznać pewne mechanizmy, warunki oraz… swoją wartość.

Towar jest wart tyle…

…ile ktoś za niego zapłaci. Można zorganizować ekskluzywne szkolenie dla uberprogramistów i żądać kilkunastu tysięcy za wejście. Ale kiedy to szkolenie będzie faktycznie tyle warte? Dopiero wtedy, gdy ktoś kupi bilet!

Można wystawić swój ukochany, pełen sentymentu i wspomnień samochód marzeń na aukcję za milion. Ale kiedy ta fura będzie tyle warta dla świata, a nie samego właściciela? Dopiero, gdy ktoś ją kupi.

Podobnie jest z naszymi programistycznymi usługami. Naczytaliśmy się portali i gazet, naoglądaliśmy telewizji śniadaniowych i nagle zamarzyło się nam kilkanaście czy kilkadziesiąt tysięcy złotych wypłaty? Owszem, jest to do zrobienia, ale wypada to zweryfikować. Jak? Znaleźć kupca, klienta zainteresowanego usługami w takiej cenie.

Na dzień dzisiejszy wszyscy jesteśmy warci – oczywiście pod względem finansowym – tyle, ile zarabiamy. Nie tyle, ile ktoś gdzieś kiedyś potencjalnie mógłby nam zapłacić.

Najlepszy argument podczas negocjacji to: „jest firma, która da więcej".

Ale uwaga! I to uwaga bardzo, bardzo ważna.

Kłamstwo ma krótkie nogi.

Uciekaj się do tej strategii tylko i wyłącznie wtedy, gdy naprawdę ktoś jest skłonny spełnić Twoje oczekiwania. Nie oszukuj, nigdy. To wróci jak bumerang.

Nie skrywaj oczekiwań

W tej kulturalnej przepychance nie chodzi o to, by wygrać albo przegrać. Chodzi o to, by każdy wstał od stołu usatysfakcjonowany. Tylko wtedy negocjacje można uznać za udane.

Nie będzie to możliwe, jeśli nie powiesz, co Ci leży na sercu. Ja szczerze argumentuję swoje potrzeby. Można to traktować jako nieco naiwne podejście, ale zawsze jestem nieco naiwny i momentami zbyt szczery. Czasami odbija się to czkawką, ale najczęściej druga strona docenia tę szczerość i nagradza ją zaufaniem.

Przygotuj się na zaprezentowanie i uzasadnienie swoich oczekiwań. W procesie przygotowania do rozmowy najważniejsze jest zrozumienie własnych pobudek. I chodzi nie tylko o poważne podejście do tematu, ale o zrozumienie… siebie! Co mnie zadowoli i dlaczego? To bardzo ważne pytanie. Czasami okaże się, że jedynym argumentem będzie: „bo kolega ma więcej". Wtedy czas wrócić do początku i zastanowić się nad całością jeszcze raz. Na spokojnie, od nowa.

A jak nie, to... !

Ostrożnie z groźbami! „Dostaję trzydzieści procent podwyżki albo odchodzę!" nie jest najlepszym startem w negocjacjach. Z tak rozpoczętej rozmowy nie ma dobrego wyjścia.

Dostaniesz trzydzieści procent bez gadania? To znaczy, że można było powiedzieć więcej! Wychodzisz z pokoju z dużą podwyżką, ale bez satysfakcji.

Dostaniesz mniej niż trzydzieści procent i nie odejdziesz? To ile są warte Twoje groźby? Następnym razem możesz nie dostać nic.

Tak czy inaczej – stawiasz firmę pod ścianą. Z partnera i współpracownika stajesz się najemnikiem. Zapytaj siebie: czy Twoje zaufanie do autora czy autorki takich słów zostałoby nadszarpnięte? Moje – na pewno. Nie znasz dnia ani godziny. Tak to jest z szantażem.

Najpewniejszy sposób na podwyżkę

Mimo wszystko najpewniejszym sposobem na uzyskanie podwyżki jest zmiana pracy. Akurat mnie zdarzało się zmienić pracę na mniej płatną, ale jestem raczej pod tym względem wyjątkiem.

Ciągłej pogoni za jak najwyższą kwotą na comiesięcznym kwicie jednak nie polecam. Praca to przecież o wiele, wiele więcej niż wysokość wypłaty. To nie jest tylko wymiana godzin na złoto. To także całe środowisko, atmosfera, znajomi, projekty, biuro, dojazdy, klienci...

Zdecydowanie zalecam rozwagę podczas zmian pracy podyktowanych tylko i wyłącznie podwyżką. Chyba że będzie to kwota naprawdę zwalająca z nóg i pozwalająca na dużo lepsze życie.

Pamiętaj tylko, że bardzo dużą część tego życia spędzasz w pracy i ewentualne frustracje z nią związane przełożą się na wszystkie inne aspekty Twojej codzienności.

⇨ *Od długiego już czasu jestem w miejscu, do którego dążyłem latami. Nie potrzebuję dodatkowego zajęcia, nie poszukuję możliwości pracy dla kogoś.*

I w takiej rzeczywistości dzwoni do mnie telefon. Głos w słuchawce mówi, że uruchamia nową firmę i poszukuje kogoś na stanowisko chief technical officer. Dofinansowanie za miliony złotych, więc będziemy żyć jak królowie.

Nawet nie pytałem o konkretne kwoty. Po minucie uciąłem rozmowę, mówiąc, że „dzisiaj nie zostanę niczyim CTO, nawet gdybym dostał całe to dofinansowanie na swoje prywatne konto".

U rozmówcy – niedowierzanie.

Warto spojrzeć nieco szerzej i docenić też inne aspekty życia, co prowadzi nas do ciekawego spostrzeżenia. To zrewolucjonizowało moje podejście do kwestii wynagrodzenia za pracę.

Mieć ciastko i zjeść ciastko

Podczas uczciwych i szczerych przygotowań do rozmowy o podwyżce dowiesz się, czego tak naprawdę chcesz. Dlaczego akurat tego i po co jest Ci to potrzebne.

Czy naprawdę brakuje Ci tylko i wyłącznie pieniędzy? Warto sobie uzmysłowić, że pieniądze to zasób, który może być najtrudniej uzyskać od firmy. Bo to właśnie pieniądze kolorują różne raporty na czerwono. To pieniądze trzeba najbardziej uzasadniać przed „górą". Wokół kasy wszystko się kręci.

Zalecam poszerzenie perspektywy. *Zoom-out*. Jak inaczej, poza dorzuceniem złota do skarbca, pracodawca może Ci dogodzić? Jeśli nie przymierasz głodem, to możliwości masz ogromne! I może się okazać, że „trzydzieści procent pensji" wcale nie będzie dla Ciebie najcenniejszym zyskiem.

Mało kto wykorzystuje takie opcje. A przecież – szczególnie w naszym zawodzie – mamy tak wiele do zdobycia!

Zaczynamy. Zamiast stałej podwyżki w pensji: dofinansowanie do wyjazdów na konferencje? Albo jedno lub dwa dowolne szkolenia w roku? Albo praca z domu przez jeden lub dwa dni w tygodniu? Albo wykupienie lekcji nauki angielskiego? Albo fajny sprzęt? Albo (do czego jeszcze dojdziemy) dzień pracy własnej? To tylko kilka pierwszych lepszych propozycji.

Jest jeszcze jedna, dająca najwięcej radości możliwość. Sam z tego korzystałem i bardzo, ale to bardzo polecam. Chodzi o redukcję etatu. „Będę pracować za to samo wynagrodzenie, ale przez cztery dni w tygodniu". Nie uwierzysz, jak bardzo pozytywnie zmieni się wtedy Twoje życie. Jednak o tym także jeszcze się rozpiszę.

Da się osiągnąć wiele. Trzeba się tylko postarać. I nie zamykać w ustalonych z góry ramach.

Jedna sprawdzona formuła?

W internecie możesz znaleźć gotowe szablony rozmów do negocjacji podwyżki. Ba, czasami są nawet podane konkretne teksty – faktycznie zdania! – jakich zaleca się użyć. Nie zamierzam ich tutaj powielać. Z jednego prostego powodu: co, jeśli nie zadziałają?

Zdajesz sobie sprawę z tego, że nie jesteś mistrzem negocjacji? Ja też nim nie jestem. Nigdy nie będę. A osoba po drugiej stronie stołu? Mistrzem też może nie być, ale prawdopodobnie przeprowadziła takich rozmów setki i zna się na tym temacie o wiele lepiej od nas. Co, jeśli rozpozna tanie chwyty? Co, jeśli wyczuje manipulację? To nie jest łatwy temat i nie chcę go upraszczać na potrzeby książki. U wielu gotowy skrypt może zadziałać, ale kilku osobom wyrządzi krzywdę. To dla mnie ryzyko nie do zaakceptowania.

Praktykuj!

Często programistyczne negocjacje są albo bardzo niepewne, albo wręcz impertynenckie. Dwa ekstrema. Wynika to nie tylko z niewiedzy, ale także z braku praktyki. Więc... ćwicz! Przełam się i spróbuj coś zyskać nawet w banalnych codziennych sytuacjach!

Podam Ci kilka przykładów, kiedy to ja się przełamałem. Czasami wystarczy po prostu grzecznie poprosić o to, czego się chce. Na początku bywa to krępujące. Wydaje się głupie. Ale dzięki takim małym kroczkom o wiele pewniej będę się czuł przy negocjowaniu prawdziwych projektów.

Popularny dyskont spożywczy oferujący promocję: „zrób zakupy za określoną kwotę i odbierz kartę ze zwierzakiem!". Moja córeczka uwielbiała te karty. Rzecz się dzieje dzień po zakończeniu rzeczonej promocji.

Przed nami pani z dzieckiem płaci za zakupy. Nie dostaje karty. „Przepraszamy, promocja już się skończyła". Dziecko w płacz. Wychodzą ze sklepu.

Przychodzi nasza kolej. Płacę. I mówię, znajdując w sobie odwagę na coś takiego chyba po raz pierwszy w życiu: „na pewno ma tam pani pod kasą jeszcze kilka tych kart; mojej córeczce brakuje raptem jednej do całego kompletu, może w drodze wyjątku...?". Wyszliśmy ze sklepu z trzema kartami. Cieszyłem się z tych kart chyba jeszcze bardziej niż córeczka.

Czasem wystarczy zapytać.

Rodzinny wyjazd. Obiad w barze. Bierzemy zestaw dziecięcy z soczkiem pomarańczowym. Po zapłaceniu córeczka decyduje, że woli inny. Niestety, pan w kasie stwierdza, że w zestawie jest pomarańczowy, więc on musi dać pomarańczowy.

Dzień później, ten sam wyjazd, ten sam bar, ten sam dziecięcy zestaw. Przed zapłaceniem uśmiecham się przy kasie i pytam, czy można inny smak soczku. „Nie ma problemu, niech sobie dziewczynka wybierze dowolny". I ten malinowy był naprawdę przepyszny!

Czasem jedna odmowa nie oznacza, że nigdy się nie uda.

Szukam oprogramowania wspomagającego moją produktywność. Znajduję. Widzę, że w ofercie nie ma wersji testowej. Do wyboru albo darmowa z ograniczonymi funkcjami, albo płatna – premium. Piszę mail do supportu z prośbą o testowe konto w wersji premium na kilka tygodni. Po kwadransie mam dostęp do wszystkich funkcji programu na kolejny kwartał.

Czasem trzeba wiedzieć, czego się chce, i to po prostu zakomunikować.

Więcej niż „tylko programista”

Poprzedni rozdział przedstawia lektury i strategie, którymi może posłużyć się każda osoba w zawodzie programisty. Jest jednak jedno „ale”.

„Tylko” programista

Programista z obowiązkami programisty będzie tylko programistą, skazanym na obracanie się w programistycznych widełkach płacowych, na zasadach przewidzianych dla stanowiska „programista”.

Oczywiste, prawda? Prawda!

Niektórym ten szklany sufit może przeszkadzać. „Wyżej... nie podskoczysz”. I koniec! Wynika to między innymi z faktu, że programista – nawet ten bardzo dobry – jest zastępowalny. Programistów i programistek mamy na rynku pracy mnóstwo, i choć trudno jest dzisiaj o szczególnie dobrych, doświadczonych, to jakoś świat się kręci. Rekrutacja wre.

Czym zatem uzasadnisz swoją potrzebę „wyjątkowych warunków” przy podejmowaniu pracy? Albo przesunięcia w górę widełek przewidzianych na danym stanowisku akurat dla Ciebie? Dlaczego ktokolwiek miałby traktować Cię inaczej?

Cóż, rzeczywistość jest nieubłagana: jesteś tylko programistą. Więc co można zrobić?

Zmienić rzeczywistość!

Przyjrzyj się swoim obowiązkom i kompetencjom. A teraz przeanalizuj otoczenie. Czym się wyróżniasz? Trochę lepiej znasz składnię języka? Dłużej klepiesz aktualny projekt? Klient Cię lubi? Dorzucasz po pięć nadgodzin tygodniowo bez słowa skargi? Hmm... to nie zmienia faktu, że nadal jesteś programistą.

A gdyby tak posiąść dodatkowe umiejętności? Takie, które wyciągną Cię sprzed monitora i uczynią uberspecem? Kolejnym etapem programistycznej ewolucji? Ekspertem łączącym w jednej osobie zakres odpowiedzialności

na kilku różnych stanowiskach? To da się zrobić. Ba, to bardzo satysfakcjonujące i dodaje pracy kolorytu!

Możesz założyć bloga, już dziś. Brzmi banalnie i głupio? Tylko dla tych, którzy nie mają o tym pojęcia. Konsekwentne, rozsądne i wartościowe treści, promowane w umiejętny sposób, szybko zaowocują nowymi kontaktami i rozpoznawalnością w środowisku.

Możesz prowadzić prezentacje na meetupie. Podzielenie się nietuzinkowymi obserwacjami na scenie – co paraliżuje większość ludzi – doda pewności siebie i przyklei nieusuwalną łatkę „prelegent". To daje szacunek. Stamtąd już tylko krok do ogólnokrajowych, a być może i zagranicznych, konferencji. Specjalista czy specjalistka na scenie przed setkami innych programistów, na dużym, znanym wydarzeniu? To najlepsza wizytówka firmy!

Ale chodzi nie tylko o działalność społecznościową. Może poza kodowaniem warto nieco zbadać świat biznesu, stając się połączeniem kodera i analityka? Albo nawet negocjatora! Jeśli aktualnie masz w ręku średnie karty, to niekoniecznie musisz czekać na nowe rozdanie. Możesz po prostu powiedzieć „dość" i rozpocząć zupełnie inną grę. Na własnych zasadach.

Tego rodzaju aktywności czynią z Ciebie niewymienny trybik w maszynie. Trybik wymagający specjalnego oliwienia, szczególnego podejścia.

Nie namawiam do tego, by zostać primadonną. Ale trzeba znać swoją wartość i wytrwale budować własną pozycję. Wymyślić, „czego mi brakuje", a następnie dążyć do uzupełnienia tych braków.

Temat ten rozwiniemy w bonusowym e-booku „Programista... i co dalej?". Nie musisz jednak go kupować, by zacząć pracować nad swoimi dodatkowymi superskillami.

Jeśli wychodzisz poza „tylko programowanie", to Twoje stanowisko również przestaje się nazywać „tylko programista". Stajesz się jedyną w swoim rodzaju kombinacją. Trzeba się tylko o to postarać.

Najlepsze, co możesz zrobić dla swojej kariery

Przez dużą część tej książki przewija się jeden motyw: co może spowodować, że programista będzie szczęśliwszą osobą? Jak wykorzystać wyjątkowy potencjał drzemiący w tej branży, by faktycznie stworzyć sobie

niecodzienne okoliczności codziennego życia?

Najcenniejsza waluta. Znowu.

Jako gatunek *homo sapiens* XXI wieku zmagamy się z problemem braku czasu. „Gdyby tylko doba miała trzydzieści godzin, to...". Albo: „Gdybym tylko dostała jeden dzień w tygodniu gratis, to...", „Stoję w miejscu i jest mi źle, ale gdybym miał więcej czasu, to...".

Fajnie jest mieć odwagę, żeby w końcu zapytać: to CO??

Ja sobie takie pytanie zadałem i okazało się, że faktycznie miałbym co z tym czasem zrobić. I postawiłem sobie za cel: chcę mieć go więcej! Tak zmieniłem priorytety, by w ciągu normalnego życia, bez zarzynania się, móc bardziej niż inni skupić się na tym, co chcę – a nie muszę – robić.

Proces ten rozpocząłem w 2012 roku. Odzyskany czas poświęcałem wtedy głównie na pisanie bloga, pobocznych projektów, nagrywanie podcasta i przygotowywanie szkoleń.

Zainwestowałem więc ten czas, by mnożył się i wrócił do mnie w większym zakresie. Bym w przyszłości mógł czerpać z niego do woli.

Teraz jest ta przyszłość. Mimo że jestem bardzo aktywny, pracuję na wielu frontach, realizuję wiele inicjatyw, to jednocześnie dbam o zwykłe, proste przyjemności. Zrozumiałem, że codzienność musi składać się nie tylko z pracy i snu. Dlatego też teraz prawie codziennie – w trakcie dnia, żeby nie skracać popołudnia spędzanego z rodziną – spaceruję po parku, chodzę po lesie, pływam na basenie, piję kawę w kawiarni, biorę gorącą kąpiel albo czytam książkę. Czasami nawet obejrzę w domu ulubiony film, pogram na konsoli albo pójdę do kina.

Kilka lat temu nie było to możliwe, bo musiałem pracować. Czy teraz nie muszę? Nadal muszę, ale... inaczej.

Ja też tak chcę! Co robić?

Rozwiązanie – a przynajmniej jego początkowy, najważniejszy etap – jest naprawdę proste. Kto powiedział, że w pracy masz spędzać minimum czterdzieści godzin tygodniowo? No kto?

Jak to kto? Oczywiście: świat. Wszyscy! Przecież tak się robi. Przecież to „kontrakt". Zachodni, cywilizowany styl życia. Ba, niektórzy będą zazdrościć, bo chodzą w kieracie dwukrotnie dłużej!

Jednak zapytaj siebie: czy to nie jest za dużo? Czy spędzanie ośmiu z zaledwie dwudziestu czterech godzin na wykonywaniu „pracy", i to najczęściej „dla kogoś", to nie przesada? Kiedy żyć?

O ile kosmiczne zarobki programistów to często *urban legend,* o tyle rzeczywiście mamy w rękawach wiele asów.

W wielu przypadkach – najprawdopodobniej także Twoim, tylko wystarczy się odpowiednio postarać – uda się wynegocjować w firmie jeden dzień pracy zdalnej. I już odzyskać nieco czasu spędzanego na dojazdach. Niby niewiele, ale to dobry początek. Jeden dzień „bardziej dla siebie". Nawet jeśli masz do pracy niedaleko i spędzasz na dotarciu tam i z powrotem (i przygotowaniach do wyjścia!) tylko godzinę, to... masz już o jedną godzinę tygodniowo więcej. Dla siebie. A godzina to wbrew pozorom bardzo dużo!

Wiesz, ile można zrobić w ciągu godziny, jeśli robi się tylko i wyłącznie to, na co ma się ochotę? Straszliwie wiele. A ile się można nauczyć! A jak fajnie zrelaksować!

Ale to nie jest nawet początek. Prawdziwym początkiem jest postanowienie, w pełnej zgodzie ze sobą, że:

> **mogę zarabiać więcej, ale wybieram zarabianie mniejszych pieniędzy...**
> **i to jest w porządku!**

Po zaakceptowaniu tego postanowienia otwierają się kolejne możliwości.

Wiem, że temat pieniędzy wałkuję w tej książce na wszystkie możliwe sposoby, ale robię to celowo. To naprawdę bardzo istotne. I trudno jest się wyrwać z tego dzikiego pędu. Tymczasem tuż obok tego wyścigu szczurów czeka wolność. Wyskocz na pobocze, do mnie, zapraszam. Inni niech się zabijają, a my sobie spokojnie poobserwujemy. Z niedowierzaniem.

⇨ *Jeden z moich międzyfirmowych transferów odbył się na imprezie. Święta, relaks, znajomi, flaszka za flaszką. W pewnym momencie padają słowa: „wiele bym dał, żebyś u mnie pracował". Akurat byłem na etapie odchodzenia z pracy oraz odkrywania tego wszystkiego, o czym piszę w tym rozdziale. Natychmiast odpowiedziałem: „mnie wcale wiele nie potrzeba, daj mi tylko czterodniowy tydzień pracy". Dosłownie pół godziny później byliśmy dogadani. I okazało się, że te trzydzieści zamiast czterdziestu godzin tygodniowo to aż nadto!*

Spróbuj tak. Zredukuj liczbę godzin spędzanych w tygodniu na zarabianiu. Drobnymi kroczkami, zaczynając od jednego dnia pracy zdalnej, wynegocjuj w firmie możliwość pracy na cztery piąte etatu. Wymień podwyżkę finansową na taki przywilej. Niech to będzie Twoim priorytetem na najbliższy rok. „Za rok będę pracować przez maksymalnie trzydzieści godzin tygodniowo". To może być Twoje najważniejsze zawodowe postanowienie. Przekonaj do tego i siebie, i swoją rodzinę. A następnie zacznij działać.

To oczywiście nie stanie się z dnia na dzień. Ale jest to stan, do którego warto dążyć. Bo wówczas otrzymasz aż dziesięć godzin tygodniowo na wszystko inne! Własne projekty, naukę nowych technologii, zdobywanie nowych kompetencji, lekturę latami oczekujących na swoją kolej książek, zbudowanie bloga... Albo po prostu: relaks, regenerację.

Ja zredukowałem etat do trzech czwartych w 2012 roku. I nie planuję powrotu do modelu *full-time*. Nigdy, przenigdy.

W chwili pisania tej książki – w połowie 2017 roku – osiągnąłem ciekawy stan. Moje zobowiązania zawodowe to dwie, trzy godziny tygodniowo, a wynagrodzenie to połowa pełnoetatowej pensji sprzed raptem sześciu lat. W połączeniu z zarobkami żony pozwoliłoby to nam przetrwać przez bardzo długi czas. Dzięki temu mogę spokojnie skupiać się na innych przedsięwzięciach, jak chociażby napisanie tej książki.

Fajnie? Bardzo! Staram się zauważać i doceniać ten niezwykły stan każdego dnia. Ale jak widzisz, wiele „fajnych" okoliczności wymaga lat starań, skupienia i determinacji.

> **The journey of a thousand miles begins with a single step.**
> ***Lao Tzu***

Jak zmotywować się do pracy po pracy?

Jak już się domyślasz, daleko mi do stwierdzenia, że prawdziwy programista powinien pracować przez osiem godzin w pracy, a potem kolejne trzy godziny w domu, codziennie. Rozumiem logikę stojącą za taką rekomendacją, ale... Programista to przede wszystkim człowiek! Jak każdy człowiek – ma prawo do życia prywatnego i wyjrzenia poza monitor.

Mam nadzieję, że na tym etapie książki wizja ta raczej nie budzi wątpliwości.

Jeden szczegół

Jest jednak jeden mały szczegół wart uwagi. Ograniczając się do działania (celowo nie piszę: „programowania") tylko i wyłącznie w ramach pracy, stoimy w miejscu. Jeżeli to miejsce nam odpowiada i chcemy w nim stać, to wszystko w porządku. (Prawie wszystko w porządku – dalsze refleksje na temat konsekwencji stania

w miejscu znajdziesz w innych rozdziałach tej książki; rób co chcesz, byle świadomie!). Ja od samego początku swojej kariery – a właściwie to nawet jeszcze przed jej rozpoczęciem – robiłem wiele ponad wymagane minimum. A to udzielanie się na forum, a to prowadzenie bloga, nagrywanie podcastu, pisanie artykułów, wystąpienia na konferencjach, organizowanie konkursów, prowadzenie szkoleń... Przez wiele lat łączyłem to wszystko z „normalną" pracą. Dzięki temu, z upływem czasu, otwierały się przede mną kolejne możliwości.

Rok w rok znajdowałem się w coraz ciekawszym miejscu. Rok w rok przecierałem oczy ze zdumienia, jak wiele udaje się osiągnąć i jak bardzo los sprzyja pracowitym. Bardzo dobrze podsumowuje to uwielbiona przeze mnie mądrość:

> **I find that the harder I work, the more luck I seem to have.**
> ***Thomas Jefferson***

To piękne spostrzeżenie i szczera prawda. Ale jak to zrobić? Z pomocą przychodzi kolejny cytat:

> **Stop watching fucking "LOST"!!!**
> ***Gary Vaynerchuk***

Gary wykrzyczał to ze sceny dekadę temu, gdy serial bił rekordy popularności, a setki milionów „hobbygodzin" w skali świata były codziennie przeznaczane na odkrywanie tajemnic wyspy razem z jego bohaterami. Można w to miejsce wstawić *Klan* czy też, w roku 2017, *Grę o tron* albo innego *Narcosa*.

Dwie przeszkody

Kluczowe jest uzmysłowienie sobie, że nie trzeba wiele. Jakakolwiek aktywność ponad wymagane przez los minimum to ogromny krok naprzód. Wykonanie tego kroku mogą nam jednak uniemożliwić dwie okoliczności.

Pierwsza to zmęczenie spowodowane codzienną pracą.

Normalna, etatowa programerka potrafi wykończyć. Znam to doskonale i wiem, jak trudne może być nawet myślenie o robieniu czegoś dodatkowego.

A skoro nawet myśleć jest ciężko, to tym bardziej robić. W tym przypadku można wziąć sprawy we własne ręce i sprawić, by praca nie była aż tak wyczerpująca!

Jeśli po standardowych ośmiu godzinach nie mamy na nic sił, warto się zastanowić: dlaczego? Może nieodpowiednie godziny pracy? Może dojazdy? Może nuda? A potem... trzeba coś zmienić. Niekoniecznie pracodawcę. Ale zadaj sobie pytanie: czy chcesz w taki sposób dociągnąć do emerytury? Nie ma na co czekać! Spraw, żeby było inaczej, lepiej. Nikt za Ciebie tego nie zrobi.

Druga przeszkoda w „drodze do lepszego życia" to konieczność zmuszania się do aktywności, gdzie „zmuszanie się" jest słowem kluczem.

Owszem, czasem trzeba samego siebie do czegoś zmusić. Zacisnąć zęby, zakasać rękawy, powiedzieć sobie „dam radę!" – i po prostu to zrobić. Niektóre cele wymagają przejścia przez nieprzyjemne etapy. Dyscyplina, samozaparcie i walka z samym sobą się przydaje. Można, co prawda, takie czynności zlecić komuś, ale na pewno nie od razu będziesz w stanie skorzystać z takiej opcji.

Jednak „zmuszanie się" nie może stanowić większej części takiej nieobowiązkowej aktywności. Na dłuższą metę to po prostu nie wypali. Ciągłe myślenie: „o nie! znowu muszę jeszcze..." w kontekście prywatnych przedsięwzięć to krótka droga do porażki. W takim wypadku przyspiesz tę porażkę!

Jeśli aktualne przedsięwzięcie nie sprawia Ci frajdy, to po prostu przestań się w nie angażować i poszukaj czegoś innego. *Simple as that.* To nie będzie porażka ani poddanie się, tylko dojrzała decyzja.

Celem „wychodzenia poza wymagane minimum" jest spełnienie, znalezienie swojej drogi. Być może zapewnienie lepszej, radośniejszej przyszłości. Osiągnięcie tego celu wymaga znalezienia aktywności sprawiającej Ci przyjemność.

W moim przypadku także nie wszystko było trafnym wyborem. Niejednokrotnie zaangażowałem się w bezsensownego złodzieja czasu. Całe „imperium devstyle" to moje ukochane dziecko, ale próbowałem też wejść w świat blogów parentingowych. Uruchomiłem blog perspektywataty.pl oraz największy w Polsce agregator takich blogów – portal parentingowe.pl. Wszystko

samodzielnie, w pojedynkę. Jeśli jednak wejdziesz teraz na te strony, to albo zobaczysz nieaktualizowane starocie, albo wyświetli Ci się błąd („strona nie istnieje"). Po kilku miesiącach takiej działalności po prostu przestałem znajdować w nich radość. Spróbowałem, uruchomiłem i okazało się, że to bez sensu. Wtedy postanowiłem bardziej skupić się na innych inicjatywach. Przedsięwzięcia parentingowe odpuściłem. Z żalem – to prawda! – bo kosztowały mnie trochę pracy i znalazły grono wiernych odbiorców. Ale i ze świadomością, że nie da się złapać wszystkich srok za ogon.

Jak znaleźć swoje „powołanie"?

Wiem, jak trudne może być postanowienie: „robię!", jeżeli nikt Ci nie mówi, co masz zrobić. To niestety efekt żmudnego procesu odmóżdżania nas przez lata systemowej edukacji, zabijania kreatywności i nawyku podążania za kimś, zamiast wytyczania własnych ścieżek. Ale i na ten problem znajdzie się rada. Daj sobie trochę czasu; to nie przychodzi z dnia na dzień! Weź nawet dzień lub dwa dni wolnego, pójdź na kilkugodzinny spacer do lasu. Pozwól myślom pokłębić się w głowie. I spisuj wszystko, co się w tej głowie dzieje.

To bardzo ważne:

> **Weź notatnik, długopis i fizycznie spisuj swoje myśli.**
> **Wszystkie. Od najbardziej banalnych po te genialne.**

Po jakimś czasie zacznie klarować Ci się wizja. Zacznie rodzić się pomysł. Coś fajnego. Coś większego. Ta iskierka może być ukryta pod wieloma warstwami odkładanego przez lata tłuszczu, „złogów nierobienia". Cierpliwości; postaraj się i ją odkop.

A jeśli ten pomysł nie wypali? Albo znudzi się po miesiącu? Jak nie zmarnować tego czasu, jak wybrać coś, co okaże się strzałem w dziesiątkę?

Nie da się. Wybierz cokolwiek, zrób cokolwiek. W najgorszym wypadku po kilku tygodniach powiesz: „ech, to jednak nie dla mnie" i zaczniesz szukać czegoś innego. Te tygodnie nie będą jednak zmarnowane! Wręcz przeciwnie.

Przyzwyczaisz się do niezależnego myślenia. Nauczysz się działania na miarę swoich możliwości, ponad poziom narzucony przez pracę.

Każdy kolejny krok jest coraz prostszy, bo będzie wymagać coraz mniej wewnętrznej walki. A pamiętaj, że

> **nawet wygrana na loterii nie przychodzi sama z siebie.**

Najpierw trzeba kupić los. Kiedy kupujesz swój?

Skąd wziąć na to czas?

Każdy ma tyle samo czasu w ciągu doby. Ale dwadzieścia cztery godziny to baaaardzo dużo. Nawet jeżeli odejmiemy od tego konieczne godziny na sen, pracę, dojazdy. Nawet jeśli mamy rodzinę i chcemy być z nią codziennie.

Ba, nawet jeżeli uwielbiasz pooglądać film czy serial, poczytać książkę albo pograć w grę! Nadal zostaje niezerowa ilość czasu.

> **Wszystko jest kwestią priorytetów. Wyboru.**

Możesz zadowolić się stanem dzisiejszym i nie robić niczego dodatkowego. I – powtórzę, bo to ważne – nie ma w tym niczego złego! Pod warunkiem, że jest to decyzja świadoma i że wszystko w Twoim życiu Ci się podoba.

Jednak decydując się na jakiekolwiek działanie, nie musisz jednocześnie rezygnować z uciech dnia codziennego. Ba, nawet bym to odradzał! Robiłem tak – i finalnie dotarłem do ściany.

Tylko warto zrozumieć, że nie musisz także popadać w drugie ekstremum i poświęcać tym uciechom sto procent swojego wolnego czasu. To pułapka, w którą często wpadamy. Działamy binarnie. Albo tak, albo nie; albo na zero procent, albo na sto. A przecież między tymi dwiema wartościami jest tak dużo przestrzeni!

Zdarza mi się zrobić bardzo dużo w ciągu jednego dnia, a do tego poczytać książkę, obejrzeć film, posłuchać w spokoju muzyki i pograć w grę. Spędzam także czas z rodziną, co jest dla mnie priorytetem.

Jak to możliwe? Przyznaję – kiedyś to możliwe nie było. Bo dopiero po dłuuugim czasie frustracji uświadomiłem sobie, że nie muszę przeczytać stu stron książki za jednym razem. Nie muszę poświęcić dwóch czy trzech godzin na obejrzenie filmu ani przejść całej mapy w StarCrafcie już dziś. I to była prawdziwa rewolucja w moim „życiu po godzinach".

Okazało się, że mnie

> **do regeneracji wystarczy nawet 15 albo 30 minut.**

I czas ten jest wystarczający do jednoczesnego zaspokojenia potrzeby „relaksu". Wtedy przestałem wściekać się i wybierać: to albo to. Mogę zrobić i jedno, i drugie, i jeszcze trzecie! Tylko... krócej.

U mnie zadziałało wspaniale. Może zadziałać i u Ciebie!

To Ty wybierasz. To Twoje priorytety. Wszystko inne jest wymówką.

Uwaga na...!

Nie wykończ się. Nie pozwól, by Twój organizm działał tylko napędzany kawą i energetykami. „Twoje ciało Twoją świątynią", bla, bla, bla... Ale jadąc nieustannie na najwyższych obrotach, dojedziesz tylko do szpitala.

Wiesz, ile wytrzymują świetne silniki w najdroższych samochodach wyścigowych? Sprawdź! W tamtym pięknym świecie wymiana jednostki napędowej następuje co chwilę. U nas nie, bo mamy tylko jedną.

Tak, wracamy do tego spostrzeżenia co jakiś czas. Nie jesteś robotem. Potrzebujesz regeneracji.

Kiedy dzieje się magia?

Powtórzymy zapytanie z poprzednich rozdziałów: ile jest dla nas warta dodatkowa godzina w ciągu doby?

Ale wiesz co? Ta dodatkowa godzina tam jest! A godzina dziennie to w przybliżeniu milion godzin rocznie (i bez kalkulatorów, proszę). Godzina dziennie poświęcona regularnie na coś innego niż zawodowa praca robi przeogromną różnicę! Zmienia życie, zmienia perspektywę, otwiera tysiące nowych możliwości i ukazuje świat w zupełnie innym świetle. Uszczęśliwia.

Ale gdzie jest ta godzina?

Programista w poszukiwaniu utraconego czasu

Najpierw zróbmy symulację typowego dnia typowego programisty.

Wstajesz o godzinie pozwalającej na zdążenie do roboty. Prysznic, śniadanko, ząbki, ubranie – i sio! Elastyczny czas pracy w tym nie pomaga, bo można tę procedurę przeciągać i przeciągać. Mijają pierwsze godziny dnia. Czas, w którym masz najwięcej zapału i energii. Energię tę radośnie pożytkujesz na higienę, transport i poranną kawę z ludźmi w biurze.

W pracy robisz swoje. Pojawia się zmęczenie, szczególnie jeśli w środku dnia załadujesz się w stołówce kartoflami czy zamawianą zespołowo pizzą. A jeśli – o zgrozo! – jesteś „prawdziwym pasjonatem", to wychodzisz z firmy dopiero późnym wieczorem. Bo jest fajnie, i *team spirit*, tak trzeba, sratatata.

Wracasz do domu; poziom energii bliski zeru. Trochę czasu dla rodziny czy znajomych. No i *Gra o tron* sama się nie obejrzy. *Call of Duty* samo się nie popyka. Browarek, winko, bo czemu nie, jak tak.

I w kimę, dobranoc. Kurtyna. Gdzie tu wcisnąć rozwój?

Sową być

Przez wiele lat moim czasem na rozwój była noc. To takie programistyczne, co nie? W nocy mogłem posiedzieć tak długo, aż padłem. Robiłem dużo. Byłem

zadowolony z efektów. Działam swoje, świat się kręci. Byle kawy i energetyków nigdy nie zabrakło!

Było super – do czasu, gdy wyczerpały się bateryjki. To nie stało się z dnia na dzień, ale trwało latami. Siedzenie po nocach oznacza minimalizowanie ilości snu. Pompowanie się wspomagaczami, podtrzymywanie powiek wykałaczkami, byle „jeszcze chociaż godzinkę" zanim głowa opadnie na klawiaturę.

Brzmi znajomo? Uwaga, bo na horyzoncie wyłania się wierzchołek góry lodowej!

Kompletnie nie wiesz o czym mowa? Uff, całe szczęście.

> **Na dłuższą metę tak się nie da.**
> **Tego się prędzej czy później pożałuje. Zawsze. To nie groźba, to fakt.**

Po latach takiego postępowania pojawiły się wahania nastrojów, problemy zdrowotne. Nieustanna irytacja, brak satysfakcji z czegokolwiek. Zaczął dopadać mnie prawdziwy strach: co się dzieje? Powinienem być szczęśliwy, a tymczasem wszystko jest bez sensu!

Regeneracja

Przyczyna okazała się banalna: człowiek nie może funkcjonować w taki sposób. Sen – niezależnie jak bardzo chcielibyśmy go uniknąć – jest konieczny, i nadmierna „optymalizacja snu" poprzez ciągłe skracanie poświęcanego nań czasu może mieć tragiczne skutki.

Nadszedł kres nocowania przed komputerem i spania po trzy, cztery godziny dziennie. Zacząłem poświęcać swoje „ulubione nocki poza łóżkiem" i po prostu się wysypiać. Sześć, siedem, osiem godzin – regularnie, codziennie. Wróciła radość, wrócił uśmiech. Ahaaa... Więc to przez to!

Może wydać Ci się to banalne czy nawet śmieszne, ale wielu programistów wymaga uświadomienia. Często bierzemy udział w takim głupim *pissing contest.* Kto to mniej spał, kto jest bardziej wyczerpany, kto dłużej dziubał w klawisze. Uwaga na to! To nasza branżowa choroba! Zboczenie!

Lepiej mieć 16 świetnych godzin w ciągu doby niż 20 beznadziejnych.

Oczywiste? Teraz tak. Ale jeszcze nie tak dawno musiałem poświęcić kilka bardzo smutnych tygodni na dojście do tego wniosku.

Nowy początek...

...dnia. Ja nie mam już sił, zdrowia ani chęci, by regularnie siedzieć po nocach. Widzę po znajomych i wielu rozmówcach, że ten etap po prostu przechodzi z wiekiem. W końcu dochodzi się do słusznego wniosku, że noc służy do spania. Tak było od wieków i dopiero niedawno człowiek zaczął majstrować przy naturalnym cyklu doby.

Rozwiązanie problemu braku czasu nasuwa się samo: w takim razie musi chodzić o poranek! I w istocie tak jest.

Mnogość zalet, jakie daje przejęcie kontroli nad początkiem dnia, aż ciężko jest opisać. Wstaję wtedy, kiedy chcę wstać. A nie w ostatniej chwili, najpóźniej jak się da. I robię to, co chcę robić, a nie to, co muszę. Bo nic mnie nie goni. Od samego początku to jest mój dzień. Mój!

Moje ranne wstawanie ewoluowało przez lata, i ciągle się zmienia. Obecnie staram się być na nogach przed piątą. Czasami jest to czwarta czterdzieści, czasami czwarta pięćdziesiąt pięć. Niezależnie od tego, czy aktualnie miałem pracę, czy też nie – początek dnia był niezależny od jakichkolwiek zobowiązań zawodowych. Robię to, co chcę. Robię to, co powinienem. Robię to, co pcha moje życie do przodu.

Po co?

Wstajesz z łóżka. Dom jeszcze śpi. Sąsiedzi śpią. Na social mediach pusto. Cisza, spokój. Zróbmy coś fajnego!

Motywacja towarzysząca takiemu nawykowi jest nieporównywalna z czymkolwiek innym. Dzień się jeszcze nie zaczął, a jestem gotów do działania! Na własny rachunek. Nieważne, co mnie czeka w ciągu dnia. Ten czas jest mój.

Konsekwentnie poświęcana sobie – i tylko sobie – nawet jedna godzina dziennie pozwoli na naukę kompletnie niedostępnych wcześniej umiejętności. Zrealizowanie najdzikszych projektów. Wymyślenie i przeprowadzenie przedsięwzięć, o jakich wcześniej nawet się nie śniło.

To bardzo niewiele i nieskończenie wiele jednocześnie. Liczy się konsekwencja, wytrwałość i zaangażowanie. Nie do przecenienia jest też pozytywny wpływ, jaki taka praktyka ma na całą resztę dnia. Idziesz do pracy z poczuciem spełnienia. Dzień się dopiero zaczyna, a Ty już masz na swoim koncie jakiś sukces!

Spróbuj. Daj szansę. Zobaczysz; zamiana godziny wieczornego relaksu na godzinę porannego działania może być najlepszą inwestycją. A tego wieczornego relaksu jeszcze trochę przecież zostanie, prawda?

Chęć kontynuowania spowoduje, że zaczniesz coraz bardziej zastanawiać się nad dalszymi perspektywami. Nad planami dalekosiężnymi, które małymi kroczkami, godzina za godziną, możesz wprowadzić w życie. Zaczniesz poszukiwać czynności wyciągającymi Cię z łóżka skoro świt.

Tylko nie marnuj tej magicznej godziny na sprawy związane z pracą! Nie otwieraj maila ani kalendarza. To jest Twój prezent dla Ciebie. Niech świat zewnętrzny Ci tego nie zepsuje.

Ta książka powstała w dużej mierze między piątą a siódmą rano.

Mój blog, prowadzony od dekady, również.

Wiele pobocznych hobbystycznych projektów, mających bardzo pozytywny wpływ na moje życie – także.

Bo w dzień po prostu… nie ma na takie rzeczy czasu!

Jak zostać skowronkiem?

Wiem, rozumiem: teraz może Ci się to wydawać nierealne. Jak to, wstać tak wcześnie? Niemożliwe!

Dlatego nie skacz od razu na głęboką wodę. Zacznij przesuwać swój dzień powoli, ale systematycznie. O kwadrans albo pół godziny tygodniowo. Tak, żeby się przyzwyczajać do tego procesu, by uniknąć szoku i zniechęcenia.

W tym procesie najważniejsze jest kilka sekund. Pierwsze kilka sekund po otwarciu oczu. Wtedy nie jesteś sobą. Wtedy włada Tobą tajemnicza siła, sterująca ciałem. Pragnąca tylko jednego: SPAAAAĆ! Twoim zadaniem jest wygranie w tym kluczowym momencie. Dojście do głosu, przejęcie kontroli. Wstanie z łóżka i wyjście z sypialni – z jakiegokolwiek powodu – oznacza zwycięstwo. Bo powrót do łóżka będzie już w takiej sytuacji świadomą, odpowiedzialną decyzją.

Podzielę się z Tobą dwoma sprawdzonymi sposobami, które prawie zawsze działają w tej nierównej walce. Pamiętaj: naszym celem jest wstanie z łóżka i opuszczenie pomieszczenia.

Pierwsza praktyka jest tyleż oczywista, co brutalna. Przed samym snem wypij trochę wody. Ile? Musisz się nauczyć swojego ciała. Jedną szklankę? Dwie szklanki? Poeksperymentuj. Celem tego zabiegu jest zmuszenie ciała do natychmiastowego zerwania się z posłania w momencie otwarcia oczu. Czasem nawet w trybie ekspresowym, *if you know what I mean*. Ufam, że wyjaśnienia zagłębiające się bardziej w ludzką fizjologię są zbędne.

Drugi trik dotyczy sposobu budzenia. Budzik dzwoniący koło ucha jest w porządku, jednak ma bardzo zdradliwą funkcję – *snooze*. Kto nigdy nie spędził godzin w łóżku, klikając *snooze* co osiem minut, niech pierwszy rzuci budzikiem!

A wystarczy położyć budzik poza zasięgiem ręki. Tak, by jego wyłączenie wymagało ruchu. Działa bardzo dobrze. Szczególnie, jeśli w pokoju obok śpi małe dziecko, tylko czekające na pretekst do porannego koncertu. Istnieje jednak ryzyko, że nie zdążymy zareagować na czas i obudzimy innych domowników.

Alternatywnie polecam zapoznanie się z aplikacją Sleep Cycle. Korzystałem z niej na Androidzie oraz iOS. Początkowo nie wierzyłem w jej skuteczność, ale ona… naprawdę działa! Darmowa wersja w zupełności wystarcza jako inteligentny budzik.

Leżący przy poduszce telefon bada ruchy i dźwięki wydawane podczas snu. W okolicy żądanej godziny wykrywa czas naszej największej aktywności

– a więc najpłytszego snu – i wtedy wybudza cichutkim, delikatnym dźwiękiem. Otwieramy oczy, uśmiechamy się i wstajemy z dobrą energią.

To z czasem staje się coraz prostsze. Gdy wyrobisz sobie taki nawyk, gdy wejdziesz w tryb realizowania w tym czasie swoich pasji, planów i marzeń, przestaniesz zauważać jakikolwiek problem.

Najważniejsza porada: miej po co wstawać. Ba, ja niekiedy budzę się o wyznaczonej porze – przed piątą rano – bez żadnego budzika! Bo po prostu chcę jak najszybciej wstać, by robić to, co kocham.

Porażka? Undefined

Oczywiście nie zawsze uda się wstać o zamierzonej porze. Zdarzą się dni, a nawet tygodnie, kiedy po prostu nie dasz rady. To normalne. Nie ma co się nad tym umartwiać i gnębić wyrzutami sumienia. Sam regularnie przegapiam te cudne wczesne godziny. Raz w tygodniu, kilka razy w miesiącu, a czasami nawet przez kilka tygodni z rzędu. Życie.

Jeśli w takiej sytuacji poczujesz, że brak Ci tych poranków, to bardzo dobrze! Po prostu wrócisz do tej praktyki, gdy znowu nadejdzie pora.

Dojrzałość

Przyznaję, to jest trudne. Ale nie ze względu na chęć spania. To jest trudne, ponieważ wymaga odpowiedzi na pytanie: „no dobra, wstaję rano... i co ja mam teraz robić?". To zmusza do refleksji. Zachęcam do poświęcenia kilku pierwszych poranków – tych krótszych, bo przesuniętych najpierw jedynie o piętnaście do trzydziestu minut – na zastanowienie się nad tą kwestią.

Może wypełnianie całych dni pracą, a wieczorów seansami, jest po prostu ucieczką? Ucieczką dokładnie przed tym pytaniem. Unikaniem myślenia o tym, co chcesz ze sobą zrobić. Czy jesteś tu, gdzie być chcesz? Czy zmierzasz w kierunku, który Cię satysfakcjonuje?

Na co poświęcisz dodatkową godzinę dziennie, którą daruje Ci los?

Challenge accepted?

Uwaga!

Jeszcze raz podkreślę z całą stanowczością: nie chodzi o wydłużenie dnia, ale o przesunięcie swoich „aktywnych" godzin.

Po prostu skracam wieczór. Pooglądam lub pogram przez godzinę, a nie dwie, trzy godziny dziennie. W zamian za to otrzymam tony satysfakcji, nowe pokłady energii, chęć do wstawania co rano z uśmiechem i kładzenie się spać z wyczekiwaniem na świetne jutro.

Jeśli chcę wstać przed piątą, to najpóźniej o dwudziestej drugiej dwadzieścia jestem już w łóżku, gotowy do snu. Jak dzidzia.

Przeczytaj więcej tutaj: http://zawodprogramista.pl/rano.

I poliuwaga!

Tak, znam pojęcie *polyphasic sleep*. Kiedyś nawet byłem na tyle nierozsądny, by autentycznie rozważać tę praktykę. Jakie to musi być super! Spać często (osiem razy dziennie), ale bardzo krótko (po piętnaście, dwadzieścia minut).

Na szczęście się na to nie zdecydowałem. Dziękuję, postoję. Albo poleżę.

Zwolennikom takich eksperymentów wystarczy powiedzieć, że nie ma żadnych badań potwierdzających pozytywny wpływ takiego cyklu na nasze zdrowie. Wręcz przeciwnie: okazuje się, że wszystkie fazy (ciągłego) snu są nam niezbędne.

Dobranoc.

⏎

Co odkryjesz dzięki *pet project*?

Jaki jest najfajniejszy programistyczny projekt? Oczywiście: nowy! Ale kto ma przyjemność tworzyć od zera nowe, świeże projekty – tak zwane greenfieldy – w codziennej pracy? Prawie nikt.

Jak jest?

Gros programistycznej pracy to utrzymanie istniejących systemów. Pół biedy, jeśli jest to system sensownie napisany. Ale często przecież siedzimy w bagnie po pachy, modląc się o to, żeby nie wpaść jeszcze pół metra głębiej.

Co oznacza wieloletnia praca w takich okolicznościach? Czy ktoś babrzący się w czymś takim przez dziesięć lat może mówić o dziesięciu latach doświadczenia? Ha, już wiesz, bo wielokrotnie o tym wspominaliśmy! On raczej będzie miał jeden rok powtórzony dziesięciokrotnie! Albo nawet jeden kwartał. W kółko i w kółko, klepanie tego samego.

Co zrobić?

Czy można zrobić coś, żeby wyrwać się z tej spirali? Można zmienić pracę. Ale istnieje duża szansa, że trafimy do takiego samego kołchozu.

Można usiąść wieczorkiem (albo rankiem!), odpalić IDE, nacisnąć *New project* – i jazda! Zacząć realizować jeden z pomysłów, które obijają się między uszami. Na pewno takie pomysły są. Są?

Wyzywam Cię: zrób to! Nie bój się, że zajmie Ci to tygodnie czy miesiące. Poświęć na ten eksperyment dosłownie trzydzieści minut, a być może dowiesz się o sobie jako o programiście… bardzo wiele.

Odkrycie

Założę się, że w większości przypadków będzie to wyglądało tak:

Pierwsze pięć minut: wybór tego, co zacząć pisać. Jeżeli naprawdę nie masz kompletnie żadnego pomysłu, to wiele propozycji znajdziesz w jednym z pierwszych rozdziałów tej książki.

Kolejne dziesięć minut: czekanie na uruchomienie się IDE, kliknięcie *New project* i zastanowienie się, jak nazwać projekt oraz gdzie go zapisać na dysku.

Ostatnie piętnaście minut: „hmm… co teraz?".

Co się okazuje? Po wyciągnięciu z tego swojego wielkiego pracowego projektu jesteś jak dziecko we mgle. Jesteś programistą „rozwiązującym problemy"

w jednym konkretnym kontekście. Nie wiesz, gdzie zacząć, co zrobić z problemem wymyślonym we własnej głowie.

Nie chodzi mi o konkretne rozwiązania! Chodzi mi o obsługę IDE i po prostu – jakikolwiek start! Przyznaj się przed sobą: nie umiesz. Jedyne, co robisz dobrze, to klikanie na służbowej maszynie.

Co dalej?

Nie ma co płakać. Nie przerażaj się! Nie obwiniaj! Też tak miałem.

Nie jest to dobry stan, ale nie jest nienormalny. Im wcześniej to sobie uświadomisz i to zaakceptujesz, tym lepiej. Bo tym wcześniej można zacząć działać!

Wystarczy bardzo niewiele, naprawdę. Nie bój się, że nagle zabraknie Ci czasu dla rodziny czy na wylegiwanie się przed TV. Dosłownie kilka godzin rozłożonych nawet na miesiąc pozwoli pokonać tę początkową bezsilność.

Po prostu zacznij to robić. Usiądź i koduj. Jeśli coś zakodujesz źle – to nic! Ważne, żeby napisać cokolwiek. Możesz odczuwać niemoc, totalną pustkę. Ale to też nic! Nie tylko Ty tak masz. Mają tak inni programiści. Ba, mają tak na przykład pisarze. Po prostu musisz się… rozpisać. Rozkodować. Rozkręcić.

Robisz to dla siebie. Zaczniesz realizować coś, co chcesz – a nie musisz! – zrobić. I Cię to wciągnie. Choć trochę.

Może przypomnisz sobie zapomniane już uczucie z czasów studenckich? Jak to było fajnie, gdy programowanie nie stanowiło wyłącznie środka do rzucenia chleba na stół?

Jeszcze dalej…?

Kolejne kroki są równie ważne, jednak o wiele łatwiejsze.

Uruchom to, co wyszło spod Twoich palców. Nawet jeśli nie jest skończone. Podziel się tym z innymi. Niech to będzie proste, brzydkie, ale niech działa.

Wrzuć na dowolny serwer testowy. Zadbaj, by można to było zobaczyć poza Twoim komputerem. Gdziekolwiek, cokolwiek; niech to będzie jakiekolwiek. Nie przejmuj się niczym.

A potem... ciesz się, że jesteś lepszym fachowcem niż jeszcze miesiąc wcześniej. W miesiąc udało Ci się pokonać autorozwojowy dystans – Twój rozwój jest większy niż w ciągu ostatniego roku. W końcu robisz coś poza ciągłym dziubaniem w kółko tego samego.

Wszystko to piszę z własnego doświadczenia. Zrobiłem i wdrożyłem stronkę msmvp.pl i portal parentingowe.pl (a po drodze kilka innych rzeczy); te dwa projekty idealnie nadają się do demonstracji, ponieważ są – albo przynajmniej były na początku – banalnie proste! A mimo to dały mi bardzo wiele. Nie wiem, czy one jeszcze działają, bo między moim a Twoim „dziś" mogło upłynąć wiele czasu. Ale nieważne.

Ważne, że przejdziesz przez cały proces. Przypomnisz sobie, że sam kod to nie wszystko. Trzeba kupić domenę. Trzeba to jakoś „ubrać", w jakikolwiek wygląd. Trzeba to wdrożyć... jakoś. Trzeba skonfigurować, żeby działało.

Daj się poznać!

Wejdź na http://dajsiepoznac.pl. To konkurs programistyczny, który kilkukrotnie organizowałem. Za jego sprawą ponad tysiąc (tak!) polskich programistów i programistek odważyło się rozpocząć swój projekt *open source*. Ponad tysiąc polskich specjalistów założyło blogi programistyczne.

Osoby te postawiły sobie wyzwanie. Wybrały technologię, której nie znają i... wystartowały. Skierowały swe kroki w nieznane. Po drugiej stronie wyszły lepsze. Nawet jeśli po drodze wiele poszło nie tak i projekt nie został finalnie ukończony. To nie ma większego znaczenia, bo nie o wyprodukowanie idealnego systemu tu chodzi.

Serio!

Gorąco do tego wszystkiego zachęcam. Pisz, twórz, rób!

Bez wypruwania sobie żył, bez odmawiania sobie relaksu, bez spiny. Ale... ot tak, na luzie. Dla własnej frajdy, rozwoju.

Wreszcie: dla satysfakcji! Spróbuj i zobacz. Pojawi się od razu.

Kiedyś wierzyłem, że programista musi robić jeden etat w pracy, a drugi później, dla siebie. To mi przeszło. Ale nie trzeba drugiego etatu, aby wcale niemało osiągnąć. To jest niewielka inwestycja, która się zwróci.

Kodem, jakkolwiek średni by nie był, podziel się na GitHubie.

A potem niech Cię rozpiera duma.

Rozwój, który nie pójdzie na marne

Dzisiejszy świat programistyczny rozwija się bardzo szybko i nie sposób za nim nadążyć. Ciągle nowe wersje wszystkiego, nowe biblioteki, nowe frameworki, nowe narzędzia. A nawet języki.

Można próbować być ze wszystkim na bieżąco, ale... no cóż, nie da się. Sam poddałem się stosunkowo wcześnie. Nawet będąc zapalonym dotnetowcem, kompletnie zignorowałem wschodzącą gwiazdę internetu od Microsoftu, czyli Silverlight. Pogromcę Flasha.

Dobrze na tym wyszedłem, bo nie zmarnowałem czasu na naukę technologii, której znajomość kilka lat później była kompletnie nieprzydatna. Zamiast tego bardzo mocno zgłębiałem temat testów jednostkowych. To procentuje do dzisiaj.

Jak wybrać obszary warte uwagi? W co zainwestować czas, by go nie zmarnować? To może wydawać się trudne, ale nie do końca takie jest. Trzeba po prostu odróżnić chwilową modę od długotrwałych trendów.

Oczywiście każda wiedza jest w jakiś sposób przydatna. Poszerza horyzonty, daje perspektywę. Ale chodzi o jak najlepsze wykorzystanie naszych cennych godzin.

Testy? Były obecne od lat i są obecne nadal. Mówi się, że przestają być potrzebne, ale głównie dlatego, że mało kto umie je porządnie napisać.

Wzorce projektowe? Kto to dzisiaj stosuje? Otóż używają ich wszyscy, ale nie zawsze świadomie. Bo wzorce są ukryte w bibliotekach i frameworkach. Fajnie jest je zauważać i być świadomym ich istnienia.

Programowanie obiektowe? No przecież jest wszędzie, prawda? Niby tak, ale ile osób zapoznało się z teorią w tym temacie, zgłębiło faktycznie znaczenie pojęcia „klasa"? Albo zastanowiło się nad (bez)sensem dziedziczenia?

Do tego warto dorzucić SOLID. W jednym ze swoich szkoleń zrobiłem wstawkę – kilka godzin „powtórki" z tych świętych zasad obiektowości. Okazuje się, że wielu zawodowych programistów nawet o nich nie słyszało!

Uniwersalną wiedzą jest Domain Driven Design. Polecam lekturę książek z tym związanych, by lepiej zrozumieć naturę naszej profesji i możliwe do niej podejścia.

Ciekawe spojrzenie na projekt da Ci CQRS i Event Sourcing. Nawet nie musisz tego stosować, ale gdy zaczniesz identyfikować zachodzące w systemie procesy jako zbiór komend i zdarzeń – rozumiejąc, dlaczego tak się dzieje – jakość Twojej pracy może znacząco wzrosnąć.

Mikroserwisy już wszystkim wychodzą bokiem. Ale warto zauważyć pewne analogie, przeanalizować przypadki udanych i nieudanych prób ich implementacji. Wreszcie: zorientować się, że to przecież nic nowego, bo wspomniane DDD i SOA (kolejny pasjonujący skrót) zdefiniowano już dość dawno temu.

W XXI wieku wypada zapoznać się z ofertą dostawców chmury: co to w ogóle jest i co można na tym zrobić? Przy okazji można liznąć temat Continuous Integration i Continuous Delivery, które dzięki temu stają się dużo prostsze i popularne.

A tak modne IoT? Zaprogramuj sobie lodówkę i uciesz się jak dziecko!

Masz zapędy naukowo-matematyczno-statystyczne? Świat Machine Learning, Big Data, sztucznej inteligencji czeka na Ciebie!

Wreszcie: nudzi Cię praca, klepanie ciągle tych CRUD-ów? Zerknij na programowanie funkcyjne! Możesz w ten sposób wywrócić swój programistyczny świat do góry nogami.

To raptem kilka propozycji na teraz – na rok 2017. Trochę klasyki, trochę współczesności. W niedalekiej przyszłości do tej listy dojdą zapewne druk 3D i wirtualna rzeczywistość. A jeśli czytasz tę książkę dużo później (masz ją wgraną w specjalny chip wczepiony do mózgu – prywatną bibliotekę?), to znaczy, że doczekaliśmy naprawdę ciekawych czasów.

I proszę bardzo – masz zapełnione najbliższe wieczory na następną dekadę! Nie da się umieć wszystkiego. Musisz wybierać. Wybieraj mądrze. Ale też:

Nie bój się zmienić zdania w trakcie.

Czy masz teraz *déjà vu?* Jeżeli czytasz tę książkę od początku, powinno się ono pojawić. Tak, wiele z tych tematów polecałem do nauki junior developerom! Tylko inaczej je argumentowałem.

Spójność FTW.

Pogoń za nowościami z chłodnej perspektywy

Pamiętam czasy, gdy w 2008 roku rodziły się: .NET 3.5. LINQ, *var,* wyrażenia lambda, *extension methods*... Rewolucja! Jakież to było wspaniałe. Tyle nowych możliwości!

I co?

Przychodzę do roboty z płonącym z wrażenia licem – i od razu kubeł zimnej wody: ty chyba śnisz! LINQ użyjesz u nas najwcześniej za dwa lata. *What???*

Zmieniłem pracę. Poszedłem do firmy, która używała LINQ2SQL, jeszcze w fazie alfa. *Bleeding edge,* nie baliśmy się tam niczego. Działało średnio. Firmy już nie ma na rynku.

Hmm...

Rozwój: 4 fun & profit

Z upływem lat zmieniają się priorytety, zmienia się postrzeganie technologii. Zmienia się wreszcie pojęcie wartości czasu.

W innych częściach tej książki poczytasz o *pet project* i dniu pracy własnej. O różnych innych ciekawych inicjatywach. Jedno i drugie rozwiązanie ma służyć „ostrzeniu programistycznej piły", czyli szlifowaniu umiejętności, podnoszeniu kwalifikacji, kolekcjonowaniu doświadczeń.

Uwielbiasz JavaScript? Co tydzień masz rewolucję! Programujesz obiektowo? To na pewno warto zerknąć w stronę programowania funkcyjnego. A do tego sprawdzić, co to jest R i dlaczego tak głośno ostatnio o *big data;* jak ugryźć ten temat?

Poszerzanie horyzontów jest potrzebne – to oliwa dla naszych programistycznych trybów. Bawiąc, uczy. Łakocie i witaminy.

Więc na co czekać? Lecimy z tym na serwery?

Hold your horses

Tutaj pojawia się problem. Jeden zapaleniec zapali drugiego zapaleńca i wrzucą w projekt coś, co nie do końca rozumieją. Co działa tylko w teorii. Jakąś bibliotekę w fazie beta, jakiś produkcik wypuszczony dopiero co. Jazda, moc!!!

Można się tym jarać, oczywiście. Sam bym się jarał – kiedyś, dawno temu. Teraz chybabym łby tym zapaleńcom pourywał.

Poszerzanie horyzontów jest fajne, dopóki można je rozpatrywać w kategoriach zabawy. Bez zobowiązań.

> **W momencie pojawienia się autentycznego ryzyka kończy się zabawa, a zaczyna podejmowanie nieodpowiedzialnych decyzji.**

Co się stanie, jeśli grupka rozwijająca projekt nagle przestanie go rozwijać? Albo sprzeda i trzeba będzie za jego używanie słono płacić? A co, jeśli jeden i drugi zapaleniec, w ferworze technicznych dyskusji, wjadą furą w drzewo? Albo przeskoczą do innej firmy, która pozwala na jeszcze więcej swobody?

Kto utrzyma projekt?

To są realne problemy. Bardzo przyjemnie jest ich nie zauważać. Ignorancja nie sprawia jednak, że problemy znikają. Wręcz przeciwnie: mogą się boleśnie ujawnić w najmniej odpowiednim momencie.

Królik doświadczalny

Ryzyko ryzykiem, ale to niejedyny wymiar problemów w „pogoni za nowym". Jest jeszcze... czas. Znowu!

Co rusz pojawiają się nowe propozycje dla programistów wszelakich technologii. Ja już lata temu postanowiłem: nie ruszam, póki gorące. Nie instaluję, kijem nie tykam, póki jeszcze mocno rośnie. Na pewnym etapie to już nie świetna zabawa, ale marnowanie życia.

Dlaczego? Bo w naszej branży tak mierzy się zaangażowanie i pasję. Czasami zdarza się usłyszeć z konferencyjnej sceny zapalenca (może mnie jeszcze kilka lat temu?). Grzmi: to jest właśnie nasz zawód, nasza powinność, nasza odpowiedzialność!

Moim zdaniem: no właśnie nie. Sorry, ja podziękuję.

Z mojej perspektywy to fucha darmowego testera. Za stary na to jestem. Zmiany co tydzień, beta zachowująca się jak niestabilna alfa. Kompletny chaos i brak poszanowania dla ludzi inwestujących czas w tę grę w kotka i myszkę. Przestało mnie to bawić. Przestałem się jarać byciem królikiem doświadczalnym dla nieodpowiedzialnych twórców oprogramowania.

Olać to?

Więc: co? Rdzewieć, zgrzybieć, olać, dusić się ciągle w tym samym sosie? Absolutnie nie! Chodzi o to, by postępować z głową. Z umiarem. Gnanie po falach kolejnych niewypałów niczym Bohun po stepie na pewno może dać radość. Nie jest jednak darmowe. Ani bezpieczne.

Warto trzymać rękę na pulsie. Orientować się, jakie są trendy, co się ogólnie dzieje dookoła. Ale czy inwestować czas w każdą nowinkę tylko dlatego,

że „chcę mieć ostrą piłę"? Polemizowałbym. Nierealne jest bycie ze wszystkim na bieżąco.

Jeśli Cię to kręci i hobbystycznie uwielbiasz alfy, kochasz bety – tak trzymaj! Jednak gdy spotkasz się z odmiennym podejściem, to niekoniecznie należy myśleć: „ale dziad! na emeryturę!". Bo dla jednego jest to radością. A dla innego, kto utrzymuje taki system: koniecznością szukania przyczyny, gdy wszystko się wali, nie wiadomo dlaczego. Zbyt wiele nocy nad tym spędziłem.

Za doświadczenie się płaci. Często płaci się właśnie tą ikrą. Tą pasją zgłębiania nowych, świeżych, gorących rozwiązań. Niejednokrotnie totalnie bezsensownych i... niedziałających.

A zarówno jedno, jak i drugie ma swoje miejsce i zastosowanie. Jedno w połączeniu z drugim pcha naszą branżę do przodu i trzyma w ryzach.

Rozwój: dalsze rekomendacje

Pierwotnie zamierzałem zamieścić w tej książce listę polecanych materiałów i internetowych zakamarków, w które warto udać się po dawkę inspiracji, wiedzy i wskazówek przydatnych do rozwoju. Bardzo sztucznie napompowałoby to jednak objętość książki, a nie chcemy przecież, by pękł Ci Kindle! (Czy tam kość promieniowa, jeśli czytasz – mniam! – wersję papierową). Dodatkowo internet ma to do siebie, że miejsca bardzo szybko pojawiają się i znikają.

Zamiast tego przygotowałem zatem dla Ciebie kilka stron WWW uzupełniających te książkę:

- http://zawodprogramista.pl/podcasty – listę polecanych podcastów,
- http://zawodprogramista.pl/konferencje – spis wartych uwagi konferencji,
- http://zawodprogramista.pl/kursy – wykaz źródeł do nauki online,

- http://zawodprogramista.pl/spolecznosci – listę społeczności programistycznych,
- http://zawodprogramista.pl/ksiazki – spis ciekawych książek,
- http://zawodprogramista.pl/ludzie – listę interesujących sylwetek z programistycznego światka,
- http://zawodprogramista.pl/blogi – wykaz blogów wartych śledzenia.

W miarę możliwości postaram się je aktualizować.

Tego wszystkiego jest... dużo. Bardzo dużo.

Jak ułatwić sobie życie? Oczywiście – odpowiednimi aplikacjami.

Czytanie

Do zbierania ciekawych treści „do przeczytania później" polecam aplikację Pocket. Dodajesz do niej interesujący link – z komputera, tabletu czy telefonu – i czytasz na dowolnym urządzeniu w wolnej chwili. Dodatkowo aplikacja prezentuje Ci treść w przyjaznym dla oka formacie, pozbywając się drażniących niekiedy styli i reklam.

Nie mogę również nie wspomnieć o aplikacji Kindle – uzupełnieniu genialnego czytnika od Amazonu. Aplikację uruchamiasz na dowolnym urządzeniu i czytasz. Używam jej sporadycznie na telefonie, gdy zapomnę wziąć ze sobą fizycznego Kindle'a (mam taki starutki, podrapany, pięcioletni – i cały czas sobie chwalę).

Słuchanie

Do podcastów nie mogłem się przekonać przez bardzo długi czas. Nawet nagrywając swój własny, nie słuchałem innych. Zidentyfikowałem kilka powodów: nudne treści, brak czasu i niewygodne słuchanie przez przeglądarkę.

Okazało się, że na wszystkie te zmory istnieją lekarstwa i że podcasty to naprawdę świetna sprawa! Moje dłuższe wywody na ten temat znajdziesz na stronie z listą podcastów, więc tutaj tylko w skrócie.

Nudne treści wyeliminowałem, znajdując interesujące audycje. Geniusz!

Brak czasu okazał się nietrafionym argumentem, bo czasu mamy bardzo dużo. Samochód lub komunikacja publiczna, ćwiczenia, bieganie, odkurzanie, gotowanie – wszędzie tam można słuchać podcastów. Oczywiście warto się czasami odłączyć i po prostu podumać albo włączyć muzyczkę, ale czas jest.

Trzeci argument to niewygodne słuchanie. I faktycznie, przez przeglądarkę nikomu nie radzę. Na szczęście są do tego odpowiednie aplikacje! Instalujemy jedną z nich na telefonie i nie dość, że mamy informacje o nowych odcinkach, to jeszcze możemy słuchać ich „na raty". U mnie sprawdza się to doskonale, dodatkowo osłabiając wymówkę „brak czasu".

Na Androida polecam Podcast Addict – świetna aplikacja!

Na iOS natomiast radzę zainstalować Overcast albo PocketCasts. Standardowo zainstalowana w iPhone aplikacja Podcasts jest… słaba.

Życie

Aplikacje pomagają, ale treści nadal pochłaniają masę czasu. Dlatego zawsze staram się pamiętać o najważniejszym: nigdy nie przeczytam i nie przesłucham wszystkiego.

Kiedyś wpadłem już w pułapkę nadproduktywności, gdy każdą chwilę życia wypełniałem kolejnymi bodźcami, źródłami informacji. Wymęczyło mnie to i nigdy do takiego trybu nie wrócę.

W życiu trzeba też czasem… żyć. Dla siebie. Wyłączyć wszystko i cieszyć się swoim własnym towarzystwem.

Ale zrobiło się zbyt refleksyjnie. Za wcześnie. Lecimy dalej!

6 | Zawód: Team Leader

Obowiązek lidera

Eksperyment z prowadzeniem zespołu może okazać się wielce satysfakcjonujący i wzbogacający. Możesz odkryć swoje nowe talenty i urozmaicić codzienne wyzwania.

Wpływ lidera

To zadanie, któremu nie wszyscy podołają. W końcu nie do tego się latami przygotowujemy. Ale trzeba się postarać. Zastanowić: jaki jest Twój obowiązek jako team leadera? I co tak naprawdę masz osiągnąć?

Ja poświęciłem temu zagadnieniu wiele godzin, gdy musiałem się odnaleźć w takiej sytuacji. Z uzyskanych wniosków byłem bardzo zadowolony! Nie tylko zresztą ja. Wiem, że część pracowników chciała być w moim zespole niezależnie od realizowanego w danym momencie projektu.

Przyszedł czas na refleksje i zrobiłem podsumowanie wszystkich moich dotychczasowych miejsc pracy, wszystkich moich szefów i liderów. Kto się na czym skupiał? Jaki był efekt dla firmy i klientów? Jak to wpływało na morale i atmosferę w zespole?

Czy moi liderzy mieli wpływ na moje odejścia z kolejnych miejsc pracy?

Obowiązek

Firma organizuje programistom warunki pracy. Środowisko. Sprzęt. Oprogramowanie. Kawkę, herbatkę – cokolwiek. Nawet te przeklęte piłkarzyki.

I najważniejsze: firma płaci.

A Ty, jako team leader, co możesz zrobić? Trzeba zorientować się w strukturze organizacji, priorytetach, układach... A potem obrać kierunek.

Moim zdaniem

> **najlepszego team leadera można poznać po tym, że ludzie zostaną w zespole, nawet jeśli konkurencja zaoferuje wyższą pensję.**

A co z projektem? Kto jest odpowiedzialny za jego powodzenie? Terminy, budżety... Odpowiadam: każdy członek zespołu, a w szczególności project manager.

Co z klientem? Kto jest odpowiedzialny za jego zadowolenie? Odpowiadam: każdy członek zespołu, a w szczególności „customer relations manager" (czy jakkolwiek nazwiemy osobę z takim zakresem odpowiedzialności).

Rolą moją – jako lidera – i Twoją – jako lidera – są zadowolenie i motywacja zespołu. W końcu teraz masz „team" w nazwie stanowiska!

Jak?

Walczyć z nudą! Nie ma mocnych. Zawsze wpadnie się w końcu w kołowrót rutyny, gdy trzeba powtarzalnie implementować podobne funkcje według ustalonych reguł, w znajomej architekturze, gdzie niewiele zostało już do wymyślenia.

Znam to bardzo dobrze. Długotrwałe wystawienie programisty na działanie tak miażdżących okoliczności potrafi zabić wszelką inicjatywę, zainteresowanie pracą i chęć rozwoju. Przychodzi do roboty, robi to samo co wczoraj, to samo co tydzień temu, to samo co miesiąc temu, a w perspektywie ma kolejne tygodnie/miesiące klepania identycznych linii kodu. W końcu wiele osób się poddaje. Gaśnie iskierka, znika zapał, i mamy zastępy robotów przed komputerami.

Zadaniem team leadera jest zadbanie, by w każdych okolicznościach dało się odnaleźć chociaż odrobinę świeżości. Aby choć raz na kilka tygodni programista mógł poznać coś nowego, wykazać się inicjatywą, oderwać na chwilę od codzienności.

Wiem doskonale, jak przeraźliwie nudna może być praca developera. Nawet jeśli w tymże developerze drzemią pokłady energii i chęci rozwoju – trzeba stworzyć odpowiednie warunki. Wykorzystać potencjał.

Nieraz doświadczyłem osobiście sytuacji, gdzie wszystko w projekcie było już wymyślone i postanowione przez kogoś innego, gdzie decyzje zostały podjęte, gdzie największy *fun* się zakończył, gdzie zostało po prostu tępe klepanie... i dopiero wtedy wkracza do pracy programista. Bierze w ręce kilof, zaciska zęby, nakłada na kark chomąto, pada na kolana i sunie przez to smutne życie

z poczuciem, że „coś jest nie tak, ale widocznie musi tak być, i pewnie wszędzie tak jest". Znasz to? Jeśli nie, to albo masz wielkie szczęście, albo jeszcze poznasz.

A jako team leader możesz postanowić sobie: „w moim zespole tak nie będzie!".

Ale serio: jak?

Na początku ostrzeżenie: musisz się przygotować na trudności. Możliwe, że już teraz zastaniesz zespół smutnych ludków z kilofami, rąbiących tę swoją nudną codzienność. To nie ich wina. Tak się po prostu dzieje. Trzeba im pokazać, że można inaczej.

Mogą nie chcieć. Bo zapomnieli, jakie to może być fajne, albo nawet nigdy tego nie doświadczyli. Przede wszystkim:

> **Traktuj programistów i programistki jak równoprawnych technicznych partnerów.**

Nie jesteś „tym jedynym". Nie jest tak, że wszystko, co trudne – czyli fajne – wymyślasz Ty, a zespołowi wrzucasz rzeźbienie w zerach i jedynkach, żeby działało. Pracujemy wspólnie, razem. Trudne funkcje dziel w miarę równo. Jedna osoba się dzięki temu rozwinie, inna – nabierze pokory. Wzajemna pomoc, dyskusje i – najważniejsze – poczucie satysfakcji.

> **Niech każdy dostanie zadania, z których może być dumny.**

Niech wytęży mózgownicę, pokombinuje, poeksperymentuje. Dzień może zakończyć się choćby i bez jednej dodatkowej linii kodu – nic nie szkodzi! Czasy mierzenia wydajności programisty w KLOC-ach (*kilo-lines of code*), już dawno odeszły do lamusa.

Ważna jest świadomość odpowiedzialności. „To właśnie ja mam zrobić coś całkiem skomplikowanego i ważnego! Mam to zrobić po swojemu, najlepiej jak potrafię! Nikt mi się nie wcina! Ufają mi! I ja im ufam! Ja to zrobię! JA!".

Nierealne? Nieprawda.

Proxy

Ale „góra" będzie krzyczeć!

To nie jest proces, który wdrożysz z dnia na dzień. To wszystko trwa. Musisz balansować między powinnością względem firmy i klienta a dobrem Twojego zespołu.

Ktoś krzyczy? Niech krzyczy, ale na Ciebie. Od Twoich ludzi wara. Sprawę postaw jasno. Jak zwykle, szczerość w komunikacji zdecydowanie pomaga.

Twoja wizja jest „nie do przyjęcia"? To może w tym miejscu, w tym czasie, nie jest to rola dla Ciebie? Nie szkodzi.

Przychodzą pretensje? Bierz je na siebie.

Przychodzą pochwały? Podkreśl rolę pracy zespołowej. Nie przypisuj sobie wszystkich zasług. Jeśli zespół się o tym dowie, bardzo ich to zaboli. Pozytywny feedback niech zawsze trafia bezpośrednio do autora czy autorki. Wiem, jak ważne jest usłyszenie czasem kilku słów pochwały za kawał dobrze odwalonej roboty.

Ale terminy gonią!

Jak tu się rozwijać, bawić, eksperymentować? Jak tu – nie przymierzając – pisać testy? Przecież ma być zrobione na już-teraz-dziś-albo-najlepiej-wczoraj!

No ale hej! Terminy zawsze gonią i będą gonić. Wymagania zawsze się zmieniają. Klient zawsze się do czegoś przyczepi. Nigdy nie będzie idealnie, bez względu na to, czy zespół poświęci kilka chwil na prawdziwą programistyczną frajdę, czy też nie.

W wielu firmach widziałem podejście: „chcielibyśmy zmienić u siebie X, nauczyć się Y, wprowadzić Z; wiemy, że w dalszej perspektywie byśmy na tym skorzystali; ale nie mamy na to czasu, bo musimy przecież programować!". I taki okres nieustannego kodowania nie trwa tydzień czy miesiąc, ale trwa latami.

Im więcej razy to słyszę, tym mniej jestem zaskoczony i tym bardziej mi ręce opadają. Dlatego też podczas rozmów o swoim przejściu na stanowisko team leadera stawiałem sprawę jasno. Nie będę kolesiem trzaskającym batem nad bandą wyrobników. Będziemy... Pamiętasz, jak to szło?

> **Robić dużo. Robić szybko. Robić dobrze.**

Ale na dłuższą metę będzie to możliwe tylko wtedy, gdy jako zespół będziemy czuć spełnienie w potrzebnej pracy.

⏎

Moje przygody z bycia liderem

Termin terminem, ale dzień w tę czy we w tę niczego nie zmieni. Im wcześniej Twoje otoczenie zaakceptuje tę wizję, tym lepiej. I to wcale nie jest nieodpowiedzialne, szczególnie przy projektach trwających miesiącami czy latami.

Może to natomiast mieć nieocenione efekty, jeśli chodzi o: stosunek programistów do pracy, zadowolenie z wykonywanych zadań, zaangażowanie w projekt, przepływ programistów do innych firm – a raczej jego powstrzymanie!

Podzielę się z Tobą kilkoma eksperymentami, które jako lider wprowadziłem w życie. Ku inspiracji.

Git jest git

Na jednym z cotygodniowych spotkań okazało się, że – hurra, hurra! – jesteśmy nieco do przodu względem harmonogramu. Co wtedy robimy? Nie, nie dokładamy zadań!

> **Co to za nagroda, jeśli za dobrze wykonaną pracę dostajesz…**
> **więcej pracy?**

Ale też nie idziemy do domu. Zadaniowy czas pracy to fajne hasło, ale w końcu w naszych realiach prawie każda umowa jest zawierana na określoną liczbę godzin. Niezależnie od sensowności takiej praktyki – tak właśnie wygląda rzeczywistość. Bonusowe godziny warto więc zainwestować, a nie zmarnować.

Umówiłem się z zespołem, że jeśli wyrobimy się z pracą przed terminem, spróbujemy czegoś nowego. Czegoś fajnego. Jak się okazało, zapał i chęci do spróbowania czegoś nowego i fajnego w ramach normalnej pracy spowodowały ukończenie wszystkich zadań tak, jak zakładaliśmy – przed czasem.

W ten wypracowany „bonusowy" dzień spotkaliśmy się w sali konferencyjnej i przez pół dnia opowiadałem zespołowi o nowym narzędziu. Akurat wtedy chodziło o przedstawienie systemu kontroli wersji Git. Przedstawiłem Gita zespołowi, który do tej pory używał TFS-a. Ludzie otworzyli szeroko oczy i usta i... tak zostali. Pokazywałem, co oferuje mój ukochany system. Pokazywałem, jak go używać. I wreszcie oznajmiłem, że w naszym zespole możemy się nim posługiwać właśnie w normalnej, codziennej pracy! Przez pozostałą część dnia wszyscy dostali proste zadania przy prawdziwym projekcie, aby zobaczyć, jak to się sprawdza w praktyce.

Jeden krótki, mały, nic nieznaczący dzień. A miał olbrzymi wpływ na całą naszą pracę w kolejnych miesiącach.

A szefostwo? Wiedzieli, że prowadzę zespół po swojemu. Ufali, że będzie dobrze. I było.

Nowe jest fajne

Kiedy indziej w projekcie pojawiła się konieczność zastosowania dodatkowego źródła danych. Ot, jakaś tam poboczna minibazka z jedną tabelą do trzymania nieważne-czego.

Śmiem mniemać, że standardowe podejście kosztowałoby jeden dzień nudnej pracy, o której programista chciałby jak najszybciej zapomnieć. Bo nie było tam absolutnie nic ciekawego. Ale

> **w małych nudnych zadaniach kryje się wielki potencjał!**

Zamiast standardowego podejścia do problemu zasugerowałem programiście coś innego: wykorzystanie biblioteki, której jeszcze nie znał. Sam miałem z nią styczność i wiedziałem, jaką frajdę sprawia jej używanie. Przednia zabawa!

Trzeba jednak pamiętać, że jednym z najciekawszych aspektów pracy programisty jest samodzielne poznawanie i rozgryzanie nowości. Czy byłoby fajnie, gdybym od razu napisał swoje rekomendacje, *best practices*, *guidelines*, podesłał kawałki przykładowego kodu – i tak dalej? Raczej średnia przyjemność. Skończyłoby się na pracy odtwórczej, a dokładnie tego chciałem uniknąć.

Ograniczyłem się więc do podania linku do strony tej biblioteki. Poszukaj, poszperaj, poczytaj, poeksperymentuj... Nie spiesz się. A najlepiej: swoje doświadczenia spisuj na bieżąco. Może dodatkowo powstanie z tego post na firmowego bloga? Kilka godzin więcej w skali wielomiesięcznego projektu było całkowicie niezauważalne. A satysfakcja, doświadczenie, poczucie rozwoju w czasie pracy: pozostało. Nie wspominając już o umiejętności wykorzystania biblioteki, która przydała się potem nieraz i długofalowo zaoszczędziła nam sporo czasu.

Z moich obserwacji wynika, że taki skok w bok – zejście z wydeptanej ścieżki, programowanie przez eksplorację – zawsze jest na dłuższą metę opłacalny.

Wskakuj na wyższy level!

Innym razem jeden z programistów nadspodziewanie sprawnie skończył zaplanowaną robotę. Tak, nawet – a może szczególnie! – w przypadku takiego trybu pracy to się zdarza.

Miałem dwa wyjścia: albo wysypać na niego nowy wór zadań, albo jakoś ciekawie wykorzystać ten nadrobiony czas. Teraz już wiesz, którą drogę wybieram w takiej sytuacji.

Moim konikiem od prawie zawsze są testy jednostkowe. Hobby, pasja, ulubiony techniczny temat.

W omawianej sytuacji zdecydowałem, że to świetna okazja do poćwiczenia TDD – Test-Driven Development. To nieskomplikowana w teorii, ale trudna w praktyce metodyka pisania kodu. Każdemu programiście przyda się trening!

Dorzuciłem mu jedno proste zadanko. Takie banalne. I zaplanowałem sporo więcej czasu, niż to wymagane. „Masz dużo czasu, a zadanie proste. Poćwicz przy nim TDD i opisz swoje doświadczenia".

Wspominałem już, że proste zadania są świetną okazją do podnoszenia kwalifikacji i poszerzania horyzontów, prawda? Teraz znowu doskonale to widać.

> **Rozwój w pracy? To moja odpowiedzialność jako lidera!**
> **Rozwój w domu? Nie mam prawa tego wymagać, od nikogo!**

Nic nie może się zepsuć. W najgorszym razie mamy zmarnowany jeden dzień, który zresztą nie będzie zmarnowany niezależnie od efektów, ponieważ doświadczenie – nawet w przypadku niewykonania zadania – pozostaje na długi czas!

Kilka artykułów, kilka postów, kilka testów później... i programista o wiele lepiej „czuje" testy. W tym projekcie to się być może nie przyda, ale w niedalekiej przyszłości – na pewno tak.

A do tego jaka frajda!

Umiar, ale...

Oczywiście nie można dbać tylko i wyłącznie o dobrą zabawę! Nie zapominajmy, że płacą nam za dostarczanie wymaganego oprogramowania. Ale w takich przypadkach cudowne jest, że tego nie trzeba wiele. Wystarczy troszkę, raz na jakiś czas. Wystarczy pokazać, że – jako lider – masz z tyłu głowy, by pracownicy się rozwijali. Że zależy Ci na tym aspekcie pracy i w miarę możliwości starasz się dostarczać go zespołowi.

Czyż takie spożytkowanie ekstra-czasu nie jest bardziej efektywne, pożyteczne i motywujące niż kolejna partia w Fifę na Xboxie?

⇨ *Warszawskie korpo słynące z bardzo przyjaznego podejścia do pracownika. Stolik do piłkarzyków oblegany przez programistów. Nagle od komputera dobiega wołanie: „Ej, krytyczny bug poleciał na produkcji!". Na co jeden z graczy odkrzykuje: „potem zobaczę, zajęty jestem!".*

Płakać mi się chce, gdy słyszę takie historie. Coś gdzieś kiedyś poszło bardzo nie tak.

Jeśli zapomnimy o tym, co jest w zawodzie programisty cudowne, i skupimy się na całej tej HR-owej otoczce, odwalając swoją robotę byle jak – przegramy.

Team leader nie może pozwolić na przegraną swojego zespołu.

Sam przez lata szukałem takiego miejsca pracy. I gdy dostałem szansę, tworzyłem takie miejsce dla innych. Z powodzeniem.

A jeśli mimo wszystko „góra nie pozwala", to... podaruj im tę książkę.

A może jesteś tą „górą" i książkę dał Ci w prezencie zespół? Weź się, *goddamnit*, ogarnij! Programista z rozbudzoną ambicją, chęcią działania, głodny bycia świetnym koderem, to skarb. A nie kolejny IT-niewolnik, przykuty na osiem godzin do klawiatury, od rana myślący tylko o obiedzie, a od obiadu – o wyjściu z pracy.

No dobra, ale skąd...?

Skąd Ty, jako team leader, masz brać pomysły na takie zadania? Cóż, to Twoje stadko. Ty bierzesz za nie odpowiedzialność. Musisz się postarać. Za to Ci płacą.

Zostaw w swoim planie pracy choć kilka chwil tygodniowo na research, na własny rozwój. Na sprawdzenie, „co w trawie piszczy". A jeśli naprawdę nie masz na to nawet godziny, to... zrób z tego zadanie dla kogoś w zespole! Raz w tygodniu niech ktoś z Twoich ludzi przygotuje prasówkę. Taką, jaką się robiło na WOS-ie w liceum. „Co się wydarzyło w tym tygodniu w świecie IT?".

A dodatkowa korzyść? Masz gotowy post na firmowego bloga. To wszystko pięknie się zazębia!

Podziękowania za świetną pracę możesz dostać nawet po latach.

Jak firma może dbać o rozwój i motywację pracowników?

Gdy los przygniata nudnymi projektami, gdy gorąca pasja zastyga w posąg zimnej obojętności, pora zacząć działać. Coś zmienić. Na szczęście i my, i nasze firmy – mamy do dyspozycji wiele możliwości walki z tym zjawiskiem.

Nie bój się: pokaż ten rozdział swojemu managerowi czy managerce. Może okaże się, że w Twoim otoczeniu jest miejsce na którąś z przedstawionych inicjatyw? One naprawdę się sprawdzają i mogą zdziałać cuda, jeśli chodzi o motywację pracowników oraz *team spirit*. Niezależnie od środowiska.

Dzień pracy własnej

Po raz pierwszy o takiej inicjatywie usłyszałem w kontekście Google'a i pracy tamtejszych programistów. Później temat ten wyskakiwał przy różnych innych okolicznościach. A na czym to polega? Otóż: raz na jakiś czas każdy z programistów ma możliwość przerwania pracy projektowej, dla klientów, i zajęcia się czymś swoim. Czymś innym. Czymś nowym.

Głównym celem może być to, żeby ludzie się nie wypalili. A dodatkowo: żeby po każdym takim dniu byli lepsi niż przed nim. Jeśli ktoś pracuje nad jednym projektem non stop, w kółko, przez X czasu, babrając się cały czas w tym samym, to może w końcu mieć dość tej stagnacji. I tak oto, dostając od pracodawcy dzień „gratis", może sobie wziąć coś nowego, pobawić się, poznać alternatywy, zobaczyć, co w trawie piszczy.

Tylko co z tym dniem zrobić? Można puścić wszystkich całkiem wolno, samopas, i powiedzieć: „róbta co chceta". Moim zdaniem warto jednak przykazać dodatkowo: „ale najpierw pomyślta". Nie sztuka usiąść do komputera i, nawet opierając się pokusie spędzenia całego dnia w rozrywkowych internetach, czas ten kompletnie zmarnować. A byłoby szkoda. Jeśli ktoś faktycznie ma tylko jeden dzień na parę tygodni, aby się w którymś kierunku rozwinąć, to warto wykorzystać go w stu procentach.

Ja bym taki dzień w jakiś sposób rozliczał. Przydałoby się, aby każdy zrobił małe podsumowanie nowych doświadczeń. Może krótka, nawet piętnastominutowa, prezentacja przed zespołem? Żeby wiedza w firmie się rozprzestrzeniała.

Fajnie byłoby te dni sensownie zaplanować, by układały się w spójną ścieżkę. Skakanie każdego takiego dnia na całkowicie różne tematy raczej nie przyniesie wielkich korzyści, bo tego czasu jest zwyczajnie za mało.

Wątpię, aby taki dzień był przydzielany raz w tygodniu – to w końcu aż dwadzieścia procent czasu pracy. Ale nawet jeden dzień w miesiącu to sporo.

Widziałbym to tak:

Unikamy sytuacji, w której każdy osobno coś sam sobie skrobie, w oderwaniu od reszty. Jeśli kilka osób będzie pracowało wspólnie, zmniejszamy pokusę spędzenia paru godzin na Fejsie czy Twitterze. Taki wewnętrzny, firmowy hackaton!

Ale też raczej szkoda byłoby narzucać z góry zespołowi, co ma robić, bo wtedy to już nie będzie praca „własna". Uczestnicy powinni zdecydować, co chcą poznać danego dnia. Dojść do jakiegoś konsensusu. Najlepiej pod warunkiem: używamy jak najmniej narzędzi, z którymi mamy już doświadczenie.

Przykładowe zadanie: napiszcie w ciągu jednego dnia szkielet czata, którego będziemy używali w firmie do wewnętrznej komunikacji. Podczas kolejnych miesięcy będzie on doszlifowywany, aż w końcu wyjdzie z tego skończony projekt zrealizowany w nowych technologiach. Fajne! I zdecydowanie do zrobienia w ciągu jednego dnia.

Jeśli każdego miesiąca działałoby się z takim rozmachem, to raptem po paru miesiącach zabrakłoby narzędzi do poznawania! I wtedy można każde z nich dodatkowo pogłębiać, dodając ficzery do rozpoczętych projektów. Co za tym idzie? Wzrost kompetencji firmy jako organizacji. Teraz programiści byliby w stanie świadomie wybrać odpowiednie narzędzie pod dany projekt, bez trzymania się kurczowo tego czy tamtego. Tego, co „już znamy".

Satysfakcja z pracy również powinna wystrzelić w górę – wspólne oczekiwanie na „dni własne" byłoby czuć dużo wcześniej. Dyskusje w kuchni nabrałyby

rumieńców. Wreszcie każdy może pogadać o czymś nowym, a nie ciągle o klepaniu tego samego.

Nawet więcej: czyż nie byłoby świetnym pomysłem spisywanie takich przygód na firmowym blogu? Albo jeszcze lepiej: wszystkie te eksperymenty opublikować na GitHubie i linkować do nich z bloga! Po prostu wypas.

Wszystko miło, wszystko fajnie, ale zostaje jeszcze jedna dość przykra kwestia: a co z terminami, klientami, prawdziwymi projektami? Wprowadzanie takiego dnia na zabawy z technologią ma sens, jeśli faktycznie ów dzień będzie się odbywał regularnie. Znam przypadki, gdzie takie coś przetrwało parę miesięcy, a potem padło. Bo przecież trzeba się zająć normalną robotą; wdrożenie czeka, klient się niecierpliwi, a my tu się bawimy. Z tego powodu inicjatywa ta, choć jak najbardziej godna pochwały, może być nieco kłopotliwa w realizacji. Jak bowiem spędzić cały dzień na dev-igraszkach, skoro robota się nawarstwia? Nie można przecież oczekiwać, że z powodu tego jednego dnia wszyscy będą harować w weekend – wtedy już lepiej zorganizować takie coś w sobotę i liczyć na ochotników. Wtedy warto wrócić do obserwacji, że jeden dzień w skali projektu to nie tak dużo...

Blog firmowy

Blog firmowy to inicjatywa, którą kilkukrotnie spotkałem w polskim internecie. Niestety rzadko udaje się go prowadzić dłużej niż kilka tygodni. A szkoda, bo ma prawie same zalety!

Skupiamy się na rozwoju zespołu: w tym przypadku odrywamy ludzi od kodu, chociaż na chwilę. Dobrze, to zdrowo. Napisanie szkicu posta nie powinno zająć dłużej niż dwie, trzy godziny. W rozbiciu na kilka dni w tygodniu nie wychodzi wiele. A programista dzięki temu zaznajomi się z zupełnie obcymi umiejętnościami, jak trudna sztuki komunikacji. Maile w zespole czy do klientów zaczną wyglądać inaczej. Lepiej. Bo praktyka czyni mistrza.

Wspólne tworzenie takiego miejsca w sieci podniesie morale zespołu, da poczucie jedności i robienia razem czegoś fajnego. Feedback do tekstów,

ich promocja w sieci, obserwowanie wzajemnego wzrostu „pisarskich" umiejętności... Tego nie można przecenić. To działa. Dla wielu osób może to być także pierwszy – ogromny – krok w kierunku społeczności. Takie przedsięwzięcie daje wielkiego powera, rozwija, uczy i zapewnia tony satysfakcji.

A o czym pisać? Jeżeli połączymy to z opisanym wcześniej dniem pracy własnej, nie ma problemu – właśnie o tym! Po zakończeniu takiego dnia każdy pisze kilka akapitów podsumowania. Posty na cały miesiąc gotowe! Ale nie jest to wymóg; przecież programistyczna praca to codzienna walka z maszynami. Walkę tę można opisać i znajdą się chętni na zapoznanie się z nią.

Wystarczy kilka osób, aby bardzo niewielkim nakładem pracy ruszyć z fajnym miejscem w sieci. Jedna osoba to za mało – wtedy nie będzie to blog „firmowy".

Dodatkowo, jako bonus: chyba nie trzeba przekonywać, że firmowy blog techniczny to doskonały wręcz sposób na promocję firmy? Zarówno wśród potencjalnych klientów, jak i – nawet bardziej – przyszłych pracowników.

To rozwiązanie nie ma wad.

Budżet szkoleniowy

Pracowałem kiedyś w firmie, w której wspólnie uzgodniliśmy, co nas – programistów – mega zadowoli. Wszelkie Xboxy czy piwka w piątek w biurze odrzucono, na szczęście, od razu. Bo to takie banalne i najprostsze.

Co wymyśliliśmy? O tym zaraz. Najpierw zobaczmy dwa przykłady tego, jak można przedstawić firmę w ogłoszeniu.

Przykład 1:

W naszym biurze masz hamaki, zjeżdżalnię i chill room z najnowszymi grami! Przychodzisz, kiedy chcesz; nikt nie kontroluje odbijania karty! Zapraszamy wszystkich ninja!

No to aplikuję, idę do roboty – a tam... Trzeba pracować! Programować! Poszaleli?! Ja chcę się bujać na hamaku! W nienormowanych godzinach pracy!!

Przykład 2:

Dbając o rozwój pracowników, umożliwiamy wyjazdy na konferencje i szkolenia. Sponsorujemy uczestnictwo w najważniejszych wydarzeniach branżowych i cieszymy się z nowych możliwości otwierających się przed naszymi programistami.

Wszystko jasne – idę do pracy, a nie na plac zabaw.

Jeżeli nie będę musiał brać urlopu, żeby pojechać na konferencję, a dodatkowo firma opłaci mi bilet, transport i hotel, to będę wnie-bo-wzię-ty!

Jeśli firma wyśle mnie na mega-szkolenie, gdzie podniosę swoje umiejętności i będę potem lepszym programistą, to niewiele więcej mi trzeba do szczęścia.

Sam od lat prowadzę szkolenia. Kilka razy ogłaszałem nabór na swoje szkolenia otwarte. Dostawałem masę maili w stylu: „gdyby tylko firma mi to zasponsorowała!". Doskonale tych ludzi rozumiem!

Chcesz zrobić swoim ludziom dobrze? W konkurencyjny, przydatny sposób? Wyślij ich na super-szkolenie! Wyślij nawet jedną osobę! Niech wróci i podzieli się pozytywną energią z całym zespołem. To nie kosztuje dużo (w porównaniu do innych dziwnych atrakcji), a motywacji zapewni na dłuuuuugi czas.

Oczywiście, nie na wszystkich to podziała. Ale jakich programistów potrzeba w firmie? Dev-speców jarających się podnoszeniem swoich kwalifikacji czy mistrzów w rzutkach?

Dlaczego reklamuje się biura jak Disneyland, skoro dziewięćdziesiąt pięć procent czasu spędzamy tam pracując? Jak to w biurze!

Jako absolutne minimum – bardziej konieczne niż MultiSport – firma powinna zapewnić swoim programistom i programistkom dostęp do jakiejś platformy szkoleniowej z kursami online.

Listę przykładowych szkoleń znajdziesz na http://zawodprogramista.pl/kursy. Nawet te płatne nie kosztują majątku, a jest to bardzo miły dodatek. I ładnie wygląda w ofertach pracy!

Internal meetup

Jednym z wielkich punktów zwrotnych w mojej karierze była decyzja: „zostanę prelegentem". Programista na scenie jest często jak ryba wyciągnięta z wody. Tak było w moim przypadku. To trudna i długa droga, ale jednocześnie rozwijająca i dająca bardzo wiele wartości oraz – co istotne – satysfakcji!

Od lat staram się angażować jak największe rzesze programistów i wyciągać ich na scenę. Bo to, co prawda, może boleć, ale warto. Robię to przez własne inicjatywy oraz współpracując z konferencjami i firmami.

Danie możliwości podszkolenia się z prowadzenia prezentacji i podzielenia się wiedzą w biurze, z koleżankami i kolegami, w czasie pracy, to coś rzadko spotykanego. Wystarczy nawet jedna trzydziestominutowa prezentacja tygodniowo, by odnieść wiele korzyści. Wspólny lunch w piątek w salce konferencyjnej? Z prelegentem opowiadającym o czymś ciekawym? O tak!

W odpowiedniej atmosferze powstają dyskusje. Można zaobserwować przepływ wiedzy – nie tylko na temat obecnych projektów, ale także w szerszym zakresie. A o czym mówić podczas prezentacji? Wracamy do tematu „dnia pracy własnej"!

Zachęceni programiści mogą spróbować swoich sił na pozafirmowych scenach. Lokalne meetupy, a później konferencje. Dołączymy do tego możliwość zaproszenia do biura ludzi spoza firmy, stream online czy późniejsze wrzucanie prezentacji na firmowy kanał na YouTube – i mamy miks iście wybuchowy.

English Fridays

Język angielski jest najważniejszym językiem programisty. Tym tematem zajęliśmy się na początku książki. Ale jak się go nauczyć, a potem szlifować? Oczywiście – praktykując!

Wiele firm organizuje pracownikom zajęcia z angielskiego. I świetnie, bardzo się to chwali. Ale można pójść o krok dalej.

Proponowanej tu inicjatywy nie widziałem jeszcze wdrożonej w praktyce. A szkoda, bo potencjalne korzyści – dla każdej ze stron – mogą być ogromne.

Na czym polega? Przez jeden dzień w tygodniu zapominamy w biurze o języku polskim. Dozwolony jest wyłącznie angielski. Nawet w rozmowach przy kawie, na czacie, albo podczas stand-upu. Tylko tyle i aż tyle.

Czy będzie to trudne? Tak! Niekomfortowe? Sam na początku czułbym się pewnie dziwnie. Ale w kupie siła! I ile niesie ze sobą wartości! Ile barier pozwoli przełamać!

Zachęcam do spróbowania.

Inne...?

Lista się tutaj nie kończy. Bardzo zachęcam do wymyślania kolejnych pomysłów i prób ich realizacji w środowisku firmowym. Spróbować nie zaszkodzi, a możliwości można mnożyć w nieskończoność.

⏎

Mój teamleadowy plan tygodnia

Rozpoczynasz „wyzwanie: team leader" w całkowicie nowym świecie. Los rzuca Cię tam, gdzie nie sięgają Twoja edukacja i przygotowanie. Trzeba sobie jakoś radzić, improwizując. Od zera, w nowej dziedzinie.

To bardzo trudna sytuacja. Ja czułem się totalnie zagubiony. Przez pierwsze dwa tygodnie nie udało mi się zrobić praktycznie nic. Nie wiedziałem, czym mam się zająć, kiedy się tym zająć, jak ogarnąć nowe obowiązki.

„Na czuja" bym daleko nie zajechał. Musiałem sobie – jak zwykle – wszystko spisać, uporządkować i usystematyzować.

> **Potrzeba planu. Pomaga prawie zawsze.**

Poświęciłem kilka godzin na utworzenie schematu tygodnia. A potem konsekwentne się go trzymałem.

Założenia

Słyszałem wielokrotnie, że w momencie zmiany stanu z „developera" na „leadera" trzeba zapomnieć o kodowaniu. No, może czasem jakiś kawałek architektury, albo *code review*. Ale normalne programowanie? Podobno „się nie da". Taki scenariusz był dla mnie najzwyczajniej w świecie nie do przyjęcia – z kilku powodów.

Po pierwsze: kodowanie nie było tylko moją pracą. Było moim hobby, moją ulubioną czynnością. Nie mogłem z tego zrezygnować.

Po drugie: nie rozumiem, jak mogę być liderem zespołu, skoro nie robię tego, co zespół? Skoro nie czuję ich codziennych bolączek, nie doświadczam problemów? Tylko przez realizację takich samych zadań jestem w stanie odpowiedzialnie stwierdzić, czy architektura jest dobra, czy kod pisze się wygodnie, czy nie ma przeszkadzajek dających się łatwo usunąć, czy każdy scenariusz został odpowiednio przemyślany, czy struktura testów jest właściwa i zachęca do ich poprawnego tworzenia, czy narzędzia/biblioteki na pewno pomagają...

Jasne, zespół może podobne problemy zgłosić albo samodzielnie naprawić, ale jako osoba bezpośrednio odpowiedzialna za projekt i kod powinienem znać te sprawy z autopsji.

Oczywiście inne obowiązki nie pozwolą mi kodować tyle, ile pozostałym członkom zespołu. Ale poziom „zero" jest nieakceptowalny.

Te dwa powody nie są jedyne, ale wystarczające.

Dlatego pierwsze i najważniejsze założenie: muszę programować. Kodowanie przez plus minus czterdzieści procent czasu uważam za godny cel.

Założenie drugie: muszę wiedzieć, co i kiedy robić oraz co i kiedy mogę sobie darować. Wszystkiego naraz nie dam rady pociągnąć.

Założenie trzecie: osoby zainteresowane muszą wiedzieć, co i kiedy robię. Muszą na mnie polegać i darzyć mnie zaufaniem. Bez ciągłego zaczepiania i pytania, „jak idzie".

Mój plan nie może być na tyle trudny do wykonania, żebym w jakiejkolwiek sytuacji nie dał rady się z czymś wyrobić. Ba, w scenariuszu idealnym (to znaczy

gdy w projekcie nie ma „pożaru") wykonanie mojego planu powinno zostawić sporo czasu na dodatkowe fajne czynności (czyli pewnie programowanie albo przygotowywanie rozwijających atrakcji dla zespołu).

Organizacja

Najpierw wypisałem zadania, które muszę wykonywać. Nie wystarczy pomyśleć przez kwadrans i spisać wszystko, co się pojawi w głowie. Najlepiej jest ten proces budować stopniowo, po prostu konsekwentnie logując każdą zawodową czynność. Po tygodniu lub dwóch pojawi się kompletny zestaw Twoich obowiązków.

Potem wystarczy porozrzucać je między dniami!

Praca w biurze generuje sporo nieplanowanych przerwań. Tym tematem zajmujemy się w innym rozdziale, ale nawet przy perfekcyjnym zastosowaniu się do wymienionych tam rad może zdarzyć się coś niespodziewanego.

Jednocześnie jestem zdania, że nie da się być dobrym „leadem" w modelu zdalnym, jeśli cały zespół pracuje w biurze. Dlatego postarałem się pogodzić jedno z drugim i ustaliłem model mieszany: przed dwa dni w tygodniu pracowałem z domu, zdalnie. Te dni najczęściej faktycznie przynosiły o wiele mniej przeszkadzajek niż dni pozostałe. Dzięki temu mogłem skupić się na tych zadaniach, które najbardziej wymagały skupienia, czyli na przykład na pracach czysto programistycznych.

Mój ówczesny plan tygodnia wyglądał następująco:

Poniedziałek

- weryfikacja zadań z piątku i weekendu (gdyby ktoś nadgorliwie postanowił ponadrabiać coś w domu)
- przegląd nowych (zaktualizowanych) zadań (problemy z założeniami lub wymaganiami – do rozwiązania przez klienta, project managera bądź innych członków zespołu)
- przegląd i ewentualne rozdzielenie w zespole nowych (zaktualizowanych) błędów znalezionych przez testerów lub klienta
- programowanie (jeśli zostanie czas, co mało prawdopodobne)

Wtorek (praca w domu)

- weryfikacja postępów w pracach
- programowanie (większość dnia)
- przegląd zadań

Środa (spotkanie i rozmowy)

- przygotowanie do spotkania: podsumowanie postępów zrobionych przez zespół w ciągu ostatniego tygodnia
- cotygodniowe spotkanie z zespołem
- rozmowy w cztery oczy z członkami zespołu w razie potrzeby
- przegląd zadań i błędów
- programowanie (jeśli zostanie czas, co mało prawdopodobne)

Czwartek (praca w domu)

- programowanie (większość dnia)
- przegląd błędów

Piątek (dzień organizacyjny)

- oznaczenie zakończonych *user stories* (potrzebne dla testerów do planowania testów na kolejny tydzień)
- *user stories* z nieskończonymi zadaniami: ustawić szacowany tydzień zakończenia
- plany na kolejny tydzień – podział zadań między członków zespołu
- przegląd zadań
- przegląd bugów

Plan ten sprawdzał się dobrze; zachęcam do jego analizy i zaaplikowania w formie dostosowanej do Twojej sytuacji.

Poza nim w organizacji pracy pomagały mi także: „lista błędów do poprawy", „lista maili do napisania", „lista kawałków kodu wymagających refactoringu" i „lista rozmów do przeprowadzenia".

Efekt

W ten oto sposób koło się zamyka. Taki rozkład jazdy ma kilka ciekawych cech.

Po pierwsze: zwykle nikt nie czeka na mój udział w rozwiązywaniu zadań (sam czekać nienawidzę).

Po drugie: zwykle weryfikuję zadania na bieżąco, akceptując je albo zwracając z feedbackiem. Raczej nie wiszą „w kolejce" dłużej niż jeden dzień (sam czekać nienawidzę – po raz drugi).

Po trzecie: zwykle każdy z członków zespołu wie, co ma robić na przynajmniej tydzień do przodu.

Po czwarte: testerzy mają dane do planowania testów na kolejny tydzień wtedy, kiedy są im potrzebne (sam czekać nienawidzę – po raz trzeci). Ja natomiast nie muszę codziennie aktualizować stanu, tylko zostawiam to najbardziej czasochłonne zadanie na wesoły przedweekendowy piątek.

I wreszcie: mnie jest dobrze, bo okazuje się, że da się „leadować" i programować jednocześnie! Trzeba tylko znaleźć odpowiednią strukturę, żeby niczego nie zaniedbać. Tak zwany algorytm.

Aha, wart uwagi jest fakt, że do pracy przychodziłem przed moim zespołem. Od lat jestem rannym ptaszkiem i zwykle byłem w biurze o siódmej rano. Miałem przynajmniej godzinę na spokojne „wejście w dzień", dzięki czemu od samego początku byłem już rozgrzany, gotowy do ewentualnego wspólnego rozwiązywania problemów.

Jak można wywnioskować z pozostałych rozdziałów, oczywiście opuszczałem biuro także przed programistami. Zawsze.

⏎

Jak prowadzić rozmowę kwalifikacyjną?

Skąd bierze się nowych programistów w firmie? Z rekrutacji oczywiście! I ktoś musi ją prowadzić. Co, jeśli jesteś to Ty?

Hurra!

Z pierwszym scenariuszem uporamy się bardzo szybko. Zakłada on, że skaczesz z radości. Nie możesz się doczekać tych wszystkich kandydatów i kandydatek, których trzeba ocenić i zweryfikować. W takim przypadku: gratuluję! I możesz spokojnie przejść do kolejnych punktów.

O nie!

Drugi scenariusz jest nieco bardziej skomplikowany. Zakłada on raczej nastawienie: „Gdybym chciał rekrutować, tobym został rekruterem! A chcę programować, więc zostałem programistą!". Ale przecież Ciebie też ktoś zrekrutował do firmy, ktoś prowadził cały proces. I – miejmy nadzieję – nie skończyło się na pytaniach o pogodę, umiejętność gry w rzutki i „plany na kolejne 5 lat". Odbyła się rozmowa techniczna, co nie?

Działy HR i rekruterzy nie są w stanie zrobić wszystkiego. Nie mają odpowiednich kwalifikacji do sprawdzenia technicznej wiedzy kandydata. Tak jak Ty nie masz odpowiednich kwalifikacji, by robić to, co oni robią na co dzień.

Nawet jeśli taka perspektywa nie wydaje się interesująca, to trzeba zrozumieć, że ktoś musi się tym zająć. Jeżeli padło na Ciebie, może to być okazja do zadowolenia. Widocznie najbardziej nadajesz się do tej misji! Masz umiejętności wykraczające poza klepanie w klawiaturę. To duży plus.

Ta osoba po drugiej stronie stołu nie jest niczemu winna, nie wyżywaj się na niej. Ona i tak jest zestresowana. To nie „Idol" czy inne sadystyczne igrzyska, a Ty nie siedzisz na krzesełku jurora odpowiedzialnego za zgnojenie biedaka. Choć wiem – uwierz, wiem – że czasami poziom kandydatów aż zachęca do złośliwości i zabawienia się w *roast*.

Ale pamiętaj, że to nie jest konkurencja, rywalizacja. W taki sposób nie można podchodzić do rekrutacji. Bo – o czym niektórzy zapominają – tylko jedna strona przygotowywała się do rozmowy, znając pytania. Ty piszesz scenariusz, Ty możesz zastawiać pułapki i wyjść na krynicę wiedzy wszelkiej, bo Ty decydujesz o tematach. Masz fory. Grasz z przewagą. Tak zwany *handicap*.

Sprawdzaj takie umiejętności, jakich faktycznie będziecie od danego człowieka potrzebowali. Nie wymagaj, by programista napisał kod na kartce. Chyba że na co dzień piszecie kod na kartce! Zlituj się, nie pytaj o parametry metod wzięte prosto z dokumentacji. Od tego są przecież IDE i internet.

Niektórzy czasami zapominają, jaki jest cel rekrutacji.

> **Twoim zadaniem nie jest udowodnienie drugiej stronie,**
> **że do niczego się nie nadaje.**

Twoje zadanie to zweryfikowanie, czy nada się do Waszego projektu! A to dwie zupełnie różne rzeczy. Reprezentujesz firmę, a Twoje podejście to wizytówka miejsca pracy.

Prowadź rekrutację tak, jakby to Twoje miejsce było po przeciwnej stronie stołu. To może być Twój przyszły kolega, koleżanka z zespołu. Z tym człowiekiem będziesz spędzać jedną trzecią swojego życia.

Jeśli spojrzysz w ten sposób, być może zmieni się Twoje nastawienie.

⏎

Jak pracować z juniorem?

Oto stało się: ktoś został Ci przydzielony pod opiekę. Hurra! Albo: *oh no!* Co teraz?

Przede wszystkim: gratulacje! Czy chcesz tego, czy nie, przed Tobą bardzo odpowiedzialne zadanie. Niezależnie od tego, czy było to zaplanowane, czy też

to tylko „wpadka", masz swoje programistyczne dziecko. I w dużej mierze Ty odpowiadasz za to, co z niego wyrośnie.

Odpowiedzialność

Może się wydawać, że to Cię przerasta. Ale spoko, tak jak z rodzicielstwem: nikt nie jest na to naprawdę w stu procentach gotowy.

Wydaje Ci się, że przesadzam? Takie coś do rodzicielstwa porównywać? Oczywiście, że przesadzam! Ale to celowy zabieg. Lepiej przechylić się w tę stronę niż w drugą.

Wielokrotnie spotykałem na swojej drodze programistów – nawet bardzo doświadczonych! – wspominających z rozrzewnieniem pierwszą pracę, pierwszy projekt i pierwszego mentora. Niezmienne twierdzą oni, że właśnie pierwszy lider ukształtował ich jako dojrzałych programistów. Dał im więcej niż całe studia.

Miło, prawda? Uwierz mi: chcesz być takim mentorem, taką mentorką! A nie zarozumiałym ważniakiem. Ale do tego zaraz dojdziemy.

Zasady

Poświęć chwilę na zastanowienie się, w jakim trybie chcesz wykonywać nową rolę. Na pewno musisz nieco swojego czasu przeznaczyć na pomoc innym, wprowadzanie w projekt, a nawet po prostu nauczanie programowania. Najlepiej od samego początku stanowczo określić ramy tej współpracy.

Zaakceptuj, że czasy wesołego kodowania non stop się już skończyły. To normalna kolej rzeczy. Nie walcz z tym, nie irytuj się – to nic nie da. Trzeba po prostu znaleźć sposób na pogodzenie programowania z nowymi obowiązkami.

Bardzo często team liderzy (tym teraz jesteś! nawet jeśli „team" to tylko jedna osoba poza Tobą) skarżą się na rozpraszanie i przerywanie ich pracy. „Oni włażą mi na głowę!", „Ciągle ktoś czegoś ode mnie chce!", „Nie mogę się na niczym skupić, bo mi non stop przerywają!". Dokładnie te słowa słyszałem od wielu programistów, na których spadły takie nowe obowiązki. To efekt braku jasno ustalonych reguł i nieumiejętnego zarządzania czasem.

Przywilejem „tych nowych" jest niewiedza. I to właśnie Ty masz im pomóc. To Twoje zadanie, równie ważne jak dotychczasowe programowanie. Musisz znaleźć na to czas, bo inaczej... za rok te osoby nadal nie będą wiedzieć tego, co potrzeba. I to już będzie Twoja wina.

Nie oznacza to bynajmniej, że masz zostać niańką i być do dyspozycji przez cały czas! To jest właśnie częsty problem: świeżo upieczeni liderzy albo całkowicie wypierają się swoich nowych obowiązków, albo pozwalają im kompletnie przejąć cały swój dzień. Ani jedno, ani drugie nie działa.

Zatem jakie praktyki mogą zaoszczędzić Ci irytacji oraz konfliktów w zespole?

Pomidor!

W światku produktywności istnieje metoda nazwana *pomodoro*. Polega ona na celebrowaniu naszej ludzkiej jednowątkowości. Niewielkie fragmenty czasu poświęcamy na sto procent pracy nad jednym zagadnieniem, bez żadnych przeszkadzaczy. Potem następuje krótka przerwa. Następnie praca – i znowu przerwa. I tak w kółko. Co czwarta przerwa jest dłuższa.

Przykładowy powtarzalny fragment dnia zaplanowany według reguł *pomodoro* może wyglądać tak: 25 minut pracy – 5 minut przerwy – 25 minut pracy – 5 minut przerwy – 25 minut pracy – 5 minut przerwy – 25 minut pracy – 35 minut przerwy.

Trzy takie bloki dają siedem i pół godziny. Pozostałe pół godziny to planowanie i organizacja pracy.

Ale dlaczego o tym piszę? W omawianej sytuacji taka praktyka może się rewelacyjnie sprawdzić! Niektóre z fragmentów czasu są Twoje i tylko Twoje. A pozostałe to „niesienia pomocy" zespołowi. Konkretny schemat wyjdzie w praktyce. Zobaczysz, co sprawdza się w Waszym przypadku.

Z mojego doświadczenia wynika, że dobrze jest poświęcić zespołowi blok czasu przed obiadem. Z tego prostego powodu, że podczas obiadu można kontynuować dyskusje. Nigdy natomiast nie przeznaczałbym zespołowi ostatniego bloku w ciągu dnia. Dlaczego? Bo takie bloki sporadycznie mogą się przedłużyć,

a ja zawsze wychodzę z pracy punktualnie. Dodatkowo przed zakończeniem pracy lubię przygotować sobie plan na następny dzień.

Jak wygląda Twój czas dla innych? Odpowiadasz na ich pytania, rozwiązujesz problemy. Programujecie w parze, poznajecie system. Przedyskutowujecie możliwe drogi. Wkładasz mentorską czapkę i tworzysz lepszą przyszłość dla swoich ludzi.

Ile takich bloków poświęcać zespołowi? Nie ma jednej właściwej odpowiedzi, ale wypracowanie tego nie jest trudne. Dodatkowo: to jest zmienne w czasie. Na samym początku na pewno junior potrzebuje więcej Twojej uwagi, a później ten wymiar możecie redukować. Ważne, by plan dnia zawsze wyglądał tak samo i abyście zmiany w harmonogramie wprowadzali za obopólną zgodą. Możesz nawet wydrukować swój plan godzinowy na dany tydzień i powiesić przy biurku. To pomaga.

A co z czasem Twoim? To proste: robisz swoje. Zamykasz się w pokoju, nakładasz słuchawki, odcinasz od świata i do dzieła!

Konsekwencja

Nikt nie ma prawa Ci przeszkodzić. Nie ma: „ale ja tylko na sekundkę", „to tylko ten jeden raz", „w drodze wyjątku". Nie i koniec! Siedzę w słuchawkach i teraz robię swoje. Wróć, gdy nadejdzie mój czas dla Ciebie, czyli o godzinie, którą znajdziesz na rozpisce. Do tej pory zrób coś pożytecznego, zastanów się nad problemem na własną rękę. Doceń i wykorzystaj samodzielność, jaką otrzymujesz.

Nie bez przyczyny na początku tego rozdziału nawiązałem do rodzicielstwa. Tam w szczególności zasady ustala się po to, by się ich trzymać. W przeciwnym razie wywołujemy tylko niepotrzebne zamieszanie.

Na początku Twoja konsekwencja może spotkać się z oporem i zdziwieniem. Z próbą sił. Bo spójrzmy prawdzie w oczy: często przecież, szczególnie w zespołach IT, na spotkaniu ustala się jedno, a potem i tak każdy robi, jak uważa. Ale nie w tym przypadku! Z czasem – w przeciągu kilku dni – współpracownicy docenią jasne postawienie sprawy. Zauważą brak kłótni, irytacji, zniecierpliwienia.

Kluczem jest stanowczość i zdecydowanie – zarówno w stosunku do innych, jak i wobec siebie. Nauczysz się pracować w określonych blokach czasowych. Nie będziesz poza nie wychodzić, odzyskasz poczucie kontroli, nie staniesz się niewolnikiem swojego zespołu i nowych obowiązków.

I... będzie Ci z tym wspaniale!

Don'ts

Wszystkie powyższe porady usprawnią proces, uczynią go łatwiejszym i dla Ciebie, i dla Twoich podopiecznych. Ale to nie wszystko. Mam też kilka rekomendacji, czego nie robić! Proszę o jedno: poświęć temu chwilę refleksji. Sam popełniałem te błędy, albo popełniano je w stosunku do mnie.

Przede wszystkim: nie rób z siebie wyroczni. Owszem, masz więcej doświadczenia. Masz szerszą perspektywę. Wiesz więcej. Może nawet jesteś mądrzejszym człowiekiem. Ale to nie oznacza, że Twoi juniorzy mają tępo akceptować Twoje rekomendacje i ślepo wykonywać rozkazy! Nie programujesz robotów, tylko szkolisz specjalistów. A specjalista myśli!

Moim mottem od czasów liceum jest:

> **Think for yourself. Question authority.**
> ***Timothy Leary***

Wypisałem to sobie nawet na ramkach pod tablicami rejestracyjnymi – tak bardzo uwielbiam ten tekst! Myślenie, kwestionowanie, poszukiwanie lepszych rozwiązań to cnota. Warta pielęgnowania bardziej niż cokolwiek innego.

Oczywiście czasami gonią terminy albo po prostu dyskusje trwają zbyt długo i należy je uciąć. Ale postęp rodzi się z kwestionowania *status quo*. Tacy ludzie w zespole to skarb!

Wiadomo też, że więcej umiesz. W przeciwnym wypadku role byłyby odwrócone. Nie musisz podkreślać tego na każdym kroku. Przekazuj swoje doświadczenie, ale nie wyszydzaj braku wiedzy. To kompletnie do niczego nie prowadzi.

Pracujesz na tym, co masz. Skupiaj się na tym, co możliwe w przyszłości, a nie na przeszłych zaniedbaniach.

Czasami będzie kusić Cię podkreślenie swojej pozycji poprzez ciągłe kontrolowanie podopiecznych. Trzymanie ich na tak zwanej krótkiej smyczy.

⇨ *Kiedyś w jednej z firm jadłem przy biurku kanapkę i czytałem jakiś programistyczny blog. Wydawałoby się: normalna sprawa. Nagle czuję pukanie w ramię i cichą reprymendę: „Maciek, czy ty nie powinieneś w pracy programować?". To był mój lider.*

Od tamtej pory tygodniami czułem się jak w Big Brotherze, *zestresowany, ciągle obserwowany i monitorowany. Produktywność i motywacja spadły straszliwie. Nie wytrzymałem w tej firmie długo.*

Twoja rola to bardzo trudne i odpowiedzialne zadanie. Życzę powodzenia! Oby Tobie poszło lepiej niż mnie lata temu.

⏎

Jak zabić inicjatywę i pasję?

Teraz zobaczymy co można zrobić, by wyssać z ludzi energię. By zrobić z nich wyrobników, nudziarzy, chłopów pańszczyźnianych. Działa w stu procentach przypadków! Satysfakcja gwarantowana!

A raczej jej brak.

Wielokrotnie zdarzało mi się, że budziłem się rano i na samą myśl o kolejnym dniu w pracy robiło mi się niedobrze. Straszne uczucie. Mimo całej świetności naszego zawodu, ogromnej gamy wyzwań i oczekujących na rozwiązanie pasjonujących problemów – bycie programistą może czasami dać się nieźle we znaki.

Przeczytaj ten rozdział, zapłacz gorzko i nigdy nie bądź takim liderem!

Jeśli jesteś po drugiej stronie i masz takiego lidera, daj mu ten rozdział do przeczytania.

Przedstawione poniżej sytuacje mają cechę wspólną: wszystkie są – o zgrozo! – autentyczne.

Płacą? To bierz i ani mruknij!

Niedługo po studiach pracowałem w firmie, w której nie musiałem robić nic. Autentycznie! Przychodziłem do pracy i przez osiem godzin czytałem książki, oglądałem filmy, układałem pasjansa. Wszystko to w oczekiwaniu na cokolwiek do roboty. Jakiekolwiek zadanie.

Wydawać się może: praca marzenie? Nic bardziej mylnego! Mogę z czystym sumieniem powiedzieć, że:

> **nicnierobienie to była najbardziej męcząca praca w moim życiu.**

(A robiłem wiele rzeczy, w tym także malowałem balkony w USA jako „robol ze Wschodu").

Nie dość, że najbardziej męcząca, to jeszcze najbardziej irytująca. Chodziłem cały czas wściekły i smutny. Czy po to studiowałem? Czy po to uczyłem się na własną rękę? Czy po to się certyfikowałem, zarywałem noce, rezygnowałem z uciech studenckiego życia?

Oczywiście na początku nie było źle! Ile blogów się wtedy naczytałem, ile teorii wchłonąłem, ile rozwiązań mogłem „testowo" zaimplementować! Tylko ciągle pojawiała się refleksja: po co? Z czasem przyszło zwykłe otępienie. Jaki jest cel tego wszystkiego? Kiedy mi się to przyda? Nigdy??

Wysłany czasami na bezsensowne spotkanie, o którego temacie nie miałem zielonego pojęcia, czy przydzielony do głupiego zadania odpowiedniego dla licealisty z klasy humanistycznej – jedyne co, mogłem zrobić, to zżymać się i nieustannie narzekać: „co ja tu robię, po co mnie zatrudniliście?".

Czas odmierzałem lunchem i kolejnymi papierosami. Chyba nigdy nie paliłem więcej niż wtedy. Szedłem do pracy, inkasowałem pensję, układałem

pasjansa, czekałem na koniec dnia, szedłem do pracy, układałem pasjansa... Straszne.

Tyle czasu zmarnowałem na tego durnego pasjansa! Ale nie miałem sił na cokolwiek innego. To zdecydowanie najgorszy czas w moim zawodowym życiu.

Skończyło się oczywiście na rzuceniu papierami i zmianie pracy, ale tych zmarnotrawionych godzin już nikt mi nie zwróci. A i raczkująca dopiero wówczas pewność siebie jakby podupadła.

Chcesz zmasakrować psychikę swojego człowieka? Spraw, że będzie totalnie nieprzydatny.

Tu jest opis zadania. Wyłącz mózg i do dzieła!

Każde zadanie można przydzielić na bardzo wiele sposobów. Wykonujący je programista czy programistka może poczuć się albo jak niezbędny tryb w pracach zespołu, albo jak tępe, w pełni zastępowalne narzędzie.

„Zrób to dokładnie tak!". Albo nawet: „masz tu kod, który kiedyś napisałem, i dopasuj go do aktualnego projektu". Za każdym razem, gdy miałem do czynienia z czymś podobnym, wyła we mnie z upokorzenia „istota rozumna". Czy ja nie potrafię tego zrobić? Czy jestem tu po to, by kopiować rozwiązania? Ja chcę tworzyć! Myśleć! Daj mi szansę!

Potencjalnie interesujące wyzwania mogą zostać w ten sposób całkowicie zmarnowane. Bezmyślne postępowanie według ułożonego wcześniej algorytmu skończy się źle. Stanowisko: programista maszynopisarz. *Consider it done!* – i już.

Oczywiście, nie można wynajdować koła od nowa po raz enty. Trzeba dzielić się wiedzą i doświadczeniem. Ale z umiarem! I szacunkiem. Niech każdy ma prawo do chwili satysfakcji i zrobienia czegoś fajnego.

Wiem, że ja sam byłem kiedyś takim liderem. Przydzielałem zadania w taki sposób, że wystarczyło tylko doklepać kod, w ogóle się nad tym nie zastanawiając. „Za wszystkie serdecznie żałuję".

Chcesz mieć stado bezmyślnych małpoludów zamiast kreatywnych programistów? Spraw, by ich mózgi były w pracy zbędne. Nieużywany narząd zanika.

Masz pomysł? Słuchamy!
A potem i tak zawsze zrobimy po mojemu

Lider jest między innymi od tego, by nadawać kierunek zespołowi. Ma decydujące zdanie, obmyśla, kombinuje... Ale dobry lider także słucha. I dyskutuje.

Programista z zapałem zabiera się do wymyślenia ciekawego rozwiązania. Potencjalnie super, świetny pomysł. Z dumą prezentuje efekt pracy – często wykonanej resztkami motywacji po godzinach! – i czeka na odrobinę uwagi. Może uznania? Albo chociaż dyskusji, a w scenariuszu minimalnym – feedbacku! Wytknięcia błędów w myśleniu! Zauważenia włożonego wysiłku!

A widzi tylko kiwnięcie głową, sekundę refleksji i usta układające się w magiczne: „oooookkkeeeej, a teraz zrobimy po mojemu". Krew człowieka zalewa.

⇨ *W jednej z firm zaproponowałem zrezygnowanie z pisania wszystkiego w procedurach składowanych. To zabiera masę czasu, jest nudne, łatwo o błędy. Nic fajnego. Zamiast tego rzuciłem pomysł, by wykorzystać ORM – chociaż do najprostszych scenariuszy.*

Sam nie wiem, na co liczyłem, ale na pewno nie na: „posprawdzałem, poczytałem, potestowałem i twoja propozycja jest bez sensu – zostajemy przy tym, co mamy". Powiedziane zaledwie kwadrans później! W tyle czasu to ciężko nawet porządnie zastanowić się nad propozycją.

Chcesz mieć w zespole wrogość, obojętność i całkowity brak zaangażowania? Wyeliminuj jakikolwiek wpływ swoich ludzi na tworzony projekt. Ignoruj pomysły i refleksje.

Płacimy ci za kodowanie. Więc „po prostu koduj"!

Nieważne, co programista robi; ważne, aby miał IDE odpalone na monitorze. Wtedy jest dobrze.

Czy to, ile czasu na ekranie widać kod, jest naprawdę dobrym miernikiem wydajności programisty?

Byłbym wniebowzięty, gdyby mój zespół interesował się czymś więcej niż tylko zalepieniem zgłoszonych bugów czy dodaniem do rozwijanego projektu nowych funkcji. Gdyby ludzie z własnej woli chcieli rozwijać swoje umiejętności, z dnia na dzień uczyć się czegoś nowego, stawać się coraz lepszymi w swoim fachu.

Chcesz mieć bierność i marazm w zespole? Stosuj mikrozarządzanie, mikrokontrolę i liderowanie batem.

> **Grunt Programmer – is only allowed to write code.**
> **It is not allowed to think about design or user requirements.**
> **That's someone else's job.**

Ku refleksji.

7 | Freelancing

Umowa o pracę czy działalność gospodarcza?

Czasami „freelancing" jest mylony z „działalnością gospodarczą" i po prostu „wystawianiem faktur". Jako programiści możemy wybrać (oczywiście w porozumieniu z pracodawcą) różne formy zatrudnienia.

Warto wiedzieć, że jedno (przejście na DG) może niechcący doprowadzić do drugiego (freelancing), i że to może być całkiem fajne zrządzenie losu.

Ważna konieczna uwaga!

Nie jestem prawnikiem ani księgowym. Ty pewnie także nie. Dlatego też w tym rozdziale jedynie wskazuję kierunki dalszego zdobywania wiedzy we własnym zakresie.

Ta treść powstaje w połowie roku 2017, a prawo ciągle się zmienia. Interpretacja prawa może być różna. Przeczytaj, przeanalizuj, a następnie zweryfikuj poprawność i aktualność tych porad dla Twojego przypadku.

Piszę ten rozdział z doświadczenia i w dobrej wierze, ale z powyższych przyczyn nie mogę wziąć za niego odpowiedzialności.

Dwie opcje

Podstawową i standardową formą jest umowa o pracę. Z punktu widzenia pracownika jest to najkorzystniejsza relacja z firmą. Nie przejmujemy się żadnymi składkami, podatkami, ustawami: po prostu wykonujemy swoją robotę, a kasa wpływa co miesiąc na konto. Komfort, wygoda, spokój.

Coraz częściej spotyka się jednak relację B2B, czyli *Business to Business*. Jako programiści wystawiamy wówczas pracodawcy faktury. Prowadzimy firmę, działalność gospodarczą.

Ja osobiście z „etatu" (umowy o pracę) przeszedłem na B2B dziesięć lat temu i korzystam ze swobody oferowanej przez tę formę zatrudnienia. Mam wielu klientów, podejmuję się różnorakich zleceń. Moja oferta obejmuje całe spektrum

usług: od programowania, przez konsultacje, prowadzenie szkoleń, marketing, aż po publikację reklam. A teraz jeszcze wydawnictwo. Firma – jak nic!

Ale jako standardowy programista także preferowałem fakturowanie klientów niż „bycie pracownikiem". Dla mnie osobiście ważne było nawet samo słownictwo. „Usługodawca" *vs* „pracownik". Pod tym względem jednak jestem nieco skrzywiony.

Zalety B2B

Dla jednych B2B to zamiennik zwykłej umowy o pracę. Dla innych sposób na optymalizacje podatkowe i oszczędzanie.

Ja z tego kroku byłem niesamowicie zadowolony głównie z jednego powodu.

Swoboda i otwarcie umysłu

Jako „przedsiębiorca", a nie „pracownik", możesz doświadczyć ciekawego zjawiska. W głowie zaczną kłębić się pomysły: co mogę teraz osiągnąć, skoro już mam własną firmę?

Tutaj poszukuj największych korzyści.

Możesz szukać innych klientów na Twoje usługi. Możesz kombinować z rozszerzeniem zakresu swojej oferty. Wreszcie możesz zastanowić się: co tak naprawdę chcę robić, mając tyle nowych możliwości?

Nie u wszystkich spowoduje to taki efekt. Ale u mnie rozpoczęcie działalności uprościło pozyskanie dodatkowych klientów. Zrozumiałem, że nie chcę do końca życia „być na etacie".

I co ważniejsze: że wcale nie muszę.

„Wrzucam w koszty"

Druga zaleta prowadzenia działalności to oddawanie mniejszej ilości pieniędzy skarbowi państwa. Przede wszystkim: na DG zapłacisz mniejszy ZUS. Dodatkowo sporo wydatków zarejestrujesz jako koszty prowadzenia działalności. Efektem będą niższe podatki.

Za bardzo nie skupiałem się na tej kwestii. Nie oszukujmy się: rachunki za telekomunikację, internet czy benzynę pewnie nie sprawią znaczącej różnicy w domowym budżecie. Ale już zakup samochodu, komputera albo telefonu to tysiące złotych oszczędności! I korzystam z tego przywileju z wielkim uśmiechem na ustach.

Dodatkowo, umowa o pracę jest kosztownym rozwiązaniem z perspektywy pracodawcy. Przechodząc na własną działalność, zwalniasz firmę z konieczności opłacania za Ciebie składek. Wystawiasz fakturę – i koniec. Nie będziemy zajmować się teraz dokładnymi wyliczeniami. Istotne jest to, że B2B generuje niższe koszty dla pracodawcy, a zaoszczędzonymi pieniędzmi można się po prostu podzielić.

Dla samych benefitów w tym zakresie ja bym jednak nie zakładał działalności. Pewnie nawet nie umiałbym wyliczyć, czy faktycznie *summa summarum* wyjdę na plus. No i nie będą to i tak żadne kokosy.

B2B – na co uważać?

To wszystko nie jest takie proste i cudowne, jak mogłoby się wydawać na pierwszy rzut oka. Koszty i obowiązki nie znikają całkowicie. One są przenoszone na Ciebie.

Koszty

W przypadku umowy o pracę firma odprowadza na Twoje konto całkiem spore składki emerytalne do ZUS-u. Wierzyć w przetrwanie ZUS-u czy nie? To temat na inną książkę. Fakt pozostaje faktem: przy własnej działalności musisz bardziej skupić się na zabezpieczeniu swojej przyszłości finansowej. Otrzymaną nadwyżkę najczęściej się „przejada", a warto ulokować ją na konto emerytalne. Nikt tego za Ciebie nie zrobi. Na B2B zapomnij o emeryturze z ZUS.

Prowadzenie firmy kosztuje. Niezależnie od tego, czy zarabiasz, czy nie, musisz zapłacić za siebie minimalną składkę ZUS. Zasady w roku 2017 wyglądają tak, że przez pierwsze dwa lata będzie to ZUS preferencyjny – niecałe pięćset

złotych miesięcznie. Później zapłacisz ponad dwa razy tyle (na dzień dzisiejszy stawki wynoszą ponad 1100 złotych i co roku są podwyższane).

Zawsze. Co miesiąc. Nie ma zmiłuj.

Należy pamiętać także o kosztach prowadzenia księgowości. Nigdy nie robiłem tego sam. Płacę księgowej 150 złotych miesięcznie. A dodatkowo: kilkanaście złotych miesięcznie za system do wystawiania faktur.

Przywileje

Umowa o pracę daje pewne przywileje, o których nawet nie myślimy. Bierzemy je za pewnik. A gdy znikają, bardzo ich brakuje. Przekonałem się o tym na własnej skórze.

Przechodząc na własną działalność, podpiszesz nową umowę z pracodawcą. Zadbaj o to, by w nowych warunkach nie zabrakło zapisu o płatnym urlopie! Moja pierwsza umowa B2B nie zawierała tego zapisu i to był ogromny błąd. Przekonałem się, jakie to uczucie, gdy przerwa w pracy podczas wakacyjnego wyjazdu kosztuje więcej niż sam wyjazd. Może to nieco zepsuć radość i relaks. Szczególnie, jeśli nie zrezygnowało się z tego przywileju w pełni świadomie.

To samo: święta. Słaba to frajda, gdy zamiast – jak wszyscy wokoło – cieszyć się „długim weekendem", wkurzasz się, że przepada Ci kilka dniówek. Albo że trzeba będzie to wszystko nadrobić. Jak temu zaradzić? Umówić się, że dni ustawowo wolne od pracy nie będą zmniejszać wynagrodzenia wypłacanego w ramach nowej umowy. Podobnie sprawa wygląda w przypadku „chorobowego". Jeden czy dwa dni przeziębienia to nic wielkiego. Ale co, jeśli złamiesz obie ręce? Albo gdy złośliwa grypa położy Cię w łóżku na pół miesiąca? Nie licz na ZUS. Przecież obniżenie jego kosztów jest jednym z celów B2B, a to ma swoje konsekwencje! Nici z wsparcia. Na umowie o pracę przez pierwsze miesiące zwolnienia dostajesz osiemdziesiąt procent pensji. A teraz? „Radź sobie samodzielnie". Albo zadbaj o odpowiedni zapis w nowej umowie.

I jeszcze jedno: okres wypowiedzenia. Nasza branża jest o tyle wyjątkowa, że długi okres wypowiedzenia służy bardziej pracodawcy niż programiście. Możesz

skorzystać z okazji i odpowiednio go skrócić. Zalecam jednak pewną ostrożność i sam nigdy nie chciałbym mieć okresu wypowiedzenia krótszego niż jeden miesiąc. W jednym szczególnym przypadku, o którym już wcześniej opowiedziałem, zgodziłem się na jego wydłużenie aż do siedmiu miesięcy.

Zagrożenia

Zanim podejmiesz decyzję o przejściu na własną działalność, upewnij się, że z prawnego punktu widzenia możesz to zrobić. Ale „jak to, czy mogę?". Umowa o pracę jest korzystna nie tylko dla pracowników, ale także dla budżetu państwa!

Pamiętasz „zaoszczędzone koszty", którymi firma może podzielić się ze świeżo upieczonym właścicielem „Usługi Programistyczne Company Imię Nazwisko"? Te pieniądze skądś się biorą. Po prostu mniej ich wpływa do państwowego skarbca.

Jakich klientów będzie mieć Twoja firma? Jak zmienią się Twoje obowiązki? Jeżeli w Twojej sytuacji zmieni się tylko i wyłącznie sposób rozliczania, stawiasz się na celowniku. Nie można tak robić. B2B jako prosta „ucieczka" od umowy o pracę to niebezpieczna zagrywka i może skończyć się kontrolami i (uzasadnionymi) pretensjami Miłościwie Nami Rządzących.

W tym przypadku nie zdawaj się na opinię pracodawcy. Musisz zadbać o odpowiednie konsultacje we własnym zakresie.

Przechytrzyć system?

Można próbować bawić się w rozwiązania „alternatywne". Istnieje opcja założenia firmy w innym kraju, lepiej traktującym przedsiębiorców, dającym więcej swobody. Bardziej z obowiązku wspominam o tej możliwości; nie robiłem tego, nie zamierzam i nie zachęcam.

Naiwne poczucie patriotyzmu nakazuje mi wspierać swoją działalnością mój kraj, tutaj płacić podatki i z pokorą przyjmować wszelkie pomysły rządzących – niezależnie od tego, kto tę władzę obecnie sprawuje.

A dodatkowo – a może nawet przede wszystkim! – nie zawsze jest to legalne.

Więc: umowa o pracę czy działalność gospodarcza?

Jeżeli teraz masz umowę o pracę, nie planujesz rozszerzania swojej działalności, cenisz spokój i komfort, to trzymaj się tej umowy.

Co istotne, działalność możesz założyć zawsze, a potem w razie potrzeby zawiesić i wrócić na etat. Ta decyzja – jak prawie każda w naszym życiu – jest odwracalna.

Więcej?

Jeśli czujesz niedosyt, udaj się na http://zawodprogramista.pl/prawo. Tam znajdziesz więcej polecanych materiałów, przygotowanych przez specjalistów, a nie programistę takiego jak Ty ;).

Korzystając z okazji polecę Ci wspaniałą książkę „Finansowy Ninja" autorstwa Michała Szafrańskiego. Otworzyła mi oczy na wiele spraw, a przy poruszanych tutaj dylematach będzie szczególnie pomocna.

Jak zostać freelancerem?

Och, gdybym za każdym razem, słysząc to pytanie, dostawał złotówkę, tobym miał... no, niemało złotówek!

Coś jest w tym legendarnym freelancingu, że tak bardzo ciągnie do niego programistów. Swoboda, wolność, wiatr we włosach!

To proste! Czyli nieskomplikowane

Procedura zostawania freelancerem nie jest skomplikowana, ani trochę. Przynajmniej, gdy patrzymy na to z nieco dalszej perspektywy. Postanawiasz: „chcę być freelancerem!". Potem znajdujesz pierwszego klienta. Potem realizujesz projekt, inkasujesz zapłatę. Powtarzasz proces.

I... to wszystko! Brawo, jesteś freelancerem!

Jednak:

Lepszy późniejszy start niż falstart.

Zwykle namawiam do działania, do podjęcia ryzyka. Twierdzę, że każda decyzja jest odwracalna. I to prawda! Ale odwracalność decyzji nie oznacza braku konsekwencji. A w tym konkretnym przypadku konsekwencje mogą być znaczne i należy się odpowiednio przygotować.

Jack of all trades

Tu jest pies pogrzebany. Okazuje się, że w momencie przejścia na freelancing przestajesz być programistą, a zaczynasz być sprzedawcą! Sprzedajesz usługi. Tak się składa, że własne. I marketingowcem. CEO, CTO i CFO w jednym. A do tego jeszcze zostajesz developerem dostarczającym sprzedawany produkt. Szef i pracownik. Biznesmen i „klepacz".

Zapotrzebowanie na te cechy rozkłada się różnie w zależności od indywidualnej sytuacji każdego freelancera. Jednak aby freelancer programista miał co robić (i co jeść), freelancer marketingowiec musi zainteresować potencjalnego klienta, a freelancer sprzedawca – „domknąć deal". To trudne i wymaga nauki, często pozyskiwanej dopiero z własnych błędów.

Praca bez bata nad karkiem

Zawód freelancera oznacza pracę zdalną. Ten temat był już poruszony w części czwartej, pamiętasz? Może warto zerknąć tam teraz ponownie.

Praca z domu, kawiarni, biura coworkingowego... Czy masz pewność, że takie warunki są dla Ciebie odpowiednie? Nie testuj samolotu już po wzbiciu się w powietrze: sprawdź to wcześniej!

Sam byłem przekonany praktycznie w stu procentach, że praca w domu jest dla mnie idealna, a mimo to podjęcie ostatecznej decyzji zajęło mi aż osiem miesięcy.

Nawet jeśli z łezką w oku wspominasz czasy uczelniane, kiedy to w domowym zaciszu realizowało się kolejne projekty na zaliczenie – to nieważne. To było. Prawdopodobnie w innych warunkach, w innych okolicznościach. To się nie liczy.

Musisz wiedzieć, jak to jest teraz.

Jak to sprawdzić? Bardzo prosto! Wymyśl projekt do zrealizowania, weź dwa tygodnie urlopu i mimo „wolnego" wstawaj regularnie, co rano. I pracuj tyle, ile trzeba.

Po takim eksperymencie nie będzie już potrzeby zgadywania – po prostu dowiesz się, czy masz odpowiednie predyspozycje. Niczego nie rób na siłę. Przygotuj się na to, że Ci się odechce. Normalna sprawa.

Wiele osób najzwyczajniej w świecie nie potrafi pracować w domu. Pokusa dłuższego pospania, pogapienia się w okno, pogrania w grę czy poklikania na Fejsie jest czasami zbyt duża. Można z tym walczyć na różne sposoby. Ja na przykład jadę do galerii handlowej, kawiarni czy parku, wkładam słuchawki do uszu, narzucam bluzę z kapturem i siedzę jak Mr. Robot. To mi pomaga.

Konieczność pilnowania się nie jest jedynym problemem. Jeszcze gorsza może być praca w pojedynkę, bez pokoju pełnego innych programistów... Akurat to mi pasuje; zawsze podczas pracy najbardziej lubiłem po prostu muzykę i święty spokój. Ale dla Ciebie brak „towarzystwa", do którego można otworzyć paszczę, może być nie do wytrzymania.

Udowodnij więc sobie, że samodzielna efektywna praca jest w Twoim wypadku wykonalna. A nawet więcej – przyjemna!

Wspomniane dwa tygodnie wystarczą do omawianego testu. Przed eksperymentem wypisz listę rzeczy, które chcesz w tym czasie osiągnąć. Po eksperymencie zobacz, co Ci się udało. Następnie odpowiedz sobie na pytanie: czy przy takiej skuteczności jesteś w stanie działać na dłuższą metę? Nie ma wymówek, że „dopiero zaczęłam", że „w kawiarni spotkałem kumpla i się zagadaliśmy", że „nie mogę wejść we *flow*". Problemy, które napotykasz teraz, będą Ci towarzyszyć także później.

Trudne pytania

Uczciwie odpowiedz na kilka pytań.

- Skąd potencjalny klient dowie się o Tobie? Wywiesisz plakat na mieście, wykupisz reklamę w radiu? Jak przekonasz go do siebie? Certyfikatami, doświadczeniem, referencjami, portfolio?
- Jaka będzie Twoja przewaga na rynku? Freelancerów i firm zajmujących się wytwarzaniem oprogramowania jest cała masa. Chcesz wyróżnić się ceną? Jakością? Specjalizacją? Czasem wykonania? Zakresem usług?
- No i... czy masz wystarczające doświadczenie, aby wziąć prawną odpowiedzialność za zrealizowanie całego projektu? Ewentualnie: czy masz finanse na wykupienie odpowiedniej polisy, gdyby jednak coś poszło nie tak? To jest często wymaganie projektowe.

To dopiero początek, ale już na tym etapie niektóre odpowiedzi mogą Cię zaboleć. I dobrze! Lepiej teraz niż później.

„Mierz siły na zamiary", „do odważnych świat należy" – i tak dalej. Ale to nie usprawiedliwia bezmyślności.

Znajdź swoje słabe punkty i postaraj się z nich wybronić. Postaw się w sytuacji potencjalnego klienta i zareklamuj sobie swoje usługi. Nie myśl, że „jakoś to będzie". Zidentyfikuj przeszkody stojące na Twojej drodze, a następnie odłóż na jakiś czas zamiar zostania freelancerem. Pokonaj je i wróć do tematu. Może potrzebujesz więcej doświadczenia zawodowego? Może brak Ci znajomości? A może przyda się dodatkowy certyfikat?

Poduszka finansowa

Masz firmę, pierwszy projekt zakończony – i cisza. Nowego zlecenia ani widu, ani słychu. Co wtedy? Brzmi nieprawdopodobnie i dziwacznie? Nie chce się o tym myśleć, bo na pewno nie przytrafi się to akurat Tobie?

Szczerze radzę, by przed wejściem w ten „biznes" najzwyczajniej w świecie napchać sobie kasy do wirtualnej skarpety. Kwota w przypadku każdej osoby

jest oczywiście inna, więc najrozsądniej będzie przedstawić ją jako „zdolność przeżycia przez kilka miesięcy". Kredyt, czynsz, ZUS, jedzenie, benzyna, rachunki, ubezpieczenia... Wszystko pomnożone. Przez jaką liczbę?

Ja startowałem we freelancingu z zapasami na trzy miesiące. A dlaczego akurat trzy? Wtedy nie miałem jeszcze kredytu, nie miałem dziecka. Widziałem to tak: dwumiesięczny okres „nieurodzaju" nie powinien kończyć tej jakże fajnej przygody. Przez tyle czasu mógłbym potencjalnie mówić sobie: „nic się nie dzieje, jeszcze będzie dobrze". Jeśli jednak po dwóch miesiącach nadal cisza... cóż, czas na szukanie normalnej pracy. Środków zostało na miesiąc, a to powinno wystarczyć na jej znalezienie.

Teraz wydaje mi się to nie do końca odpowiedzialne. Dzisiaj celowałbym w minimum sześciomiesięczne zabezpieczenie finansowe. Może z wiekiem i doświadczeniem człowiek potrzebuje coraz więcej asekuracji?

Zabezpieczenie finansowe jest też konieczne ze względu na ewentualne okresy choroby. Nie licz na to, że wyżyjesz z ochłapów rzuconych przez ZUS. Jesteś na swoim, to jesteś na swoim! Zapomnij o otrzymywaniu na chorobowym osiemdziesięciu procent pensji tak jak na etacie.

Na czym będziesz pracować?

Na etacie jest jasne: dostajesz komputer, masz *software* i zasuwasz. Nie przejmujesz się takimi drobnostkami jak system operacyjny, IDE, programy graficzne, subskrypcje, dodatkowe biblioteki.

Przed podjęciem decyzji o freelancingu rzuć okiem na cennik narzędzi potrzebnych Ci do pracy. W niektórych technologiach wszystko zrobisz na darmowych wersjach, ale w innych trzeba przygotować się na wydatki. I to czasami bardzo duże.

Zabezpiecz się

Skoro już jesteśmy przy finansach... Pamiętaj, że Ty realizujesz projekt i wystawiasz faktury. I co dalej? To oczywiste: klient płaci!

Niestety – nie zawsze. Może się to wydawać niewiarygodne, ale nieopłacone faktury są normą. Opóźnienia w płatnościach to także codzienność.

A Ty swoje koszty ponosisz cały czas. Ba, musisz także zapłacić podatki za wystawione faktury. Niezależnie od tego, czy pieniądze za wykonaną usługę faktycznie do Ciebie dotarły!

Doświadczyłem tego na własnej skórze. Nie dość, że klient nie opłacił moich faktur – więc przez dość długi czas pracowałem za darmo – to jeszcze na dobitkę sporą część tej kwoty musiałem dodatkowo oddać skarbówce. Już z własnej kieszeni. Podwójnie na minusie. Ta historyjka pojawiła się już we wcześniejszych rozdziałach. Bardzo przykre doświadczenie.

Jak się przed tym ustrzec? Mam dla Ciebie trzy porady:

Po pierwsze, rozbijaj płatność za projekt na etapy. Nigdy nie zgadzaj się na jedną, stuprocentową zapłatę dopiero po zakończeniu przedsięwzięcia i oddaniu projektu. Staraj się o zaliczkę i kolejne transze wpłacane na konto wraz z postępami prac.

Jeśli potencjalny usługobiorca nalega i absolutnie nie chce rozważyć alternatywnego harmonogramu rozliczeń, poszukaj innego klienta. Nigdy nie daj się oczarować milutkim „przecież musimy sobie ufać!". Bo zaufanie powinno działać w dwie strony.

Po drugie, zainteresuj się usługą faktoringu. W skrócie: między Tobą a kontrahentem pojawi się pośrednik. Od tej pory Twoje faktury będą zawsze opłacone w terminie – właśnie przez tego pośrednika. Niepłacący klient? Opóźnienia? Konieczność windykacji czy szukania sprawiedliwości w sądzie? To już nie Twoja sprawa – i życie staje się prostsze. Taka usługa oczywiście kosztuje! Ale czasami warto ją rozważyć dla spokoju ducha.

I po trzecie, faktura pro forma. To nie jest poradnik podatkowo-finansowy, więc na chwilę obecną wystarczy stwierdzenie, że od pro formy nie musisz płacić podatków.

A więcej materiałów na podobne tematy znajdziesz pod adresem: http://zawodprogramista.pl/prawo.

Chcesz być księgową? Idź na księgowość!

Jesteś programistą, więc doskonale wiesz, jak ważny jest odpowiedni podział zadań między najbardziej kompetentne osoby.

W IT jest jasne: jeśli bazodanowiec zabierze się za interfejs użytkownika, to raczej nikt nie będzie chciał tego używać. Jeśli frontendowiec zacznie implementować kod wielowątkowy na serwerze, system długo nie pociągnie.

Po prostu: każdy się na czymś zna. Nawet jeśli znasz się na wszystkim związanym z programowaniem – brawo, full-stacku! – to i tak Twoja głowa jest już pełna cennej wiedzy. Dorzuć do tego wszystko związane z prowadzeniem działalności – i wychodzi naprawdę pokaźny zbiór informacji.

Odciąż się, jak to tylko możliwe.

Nawet nie staraj się wnikać we wszystkie prawne kruczki, podatkowe wygibasy i proceduralne bzdury. Od tego jest biuro rachunkowe – jak najszybciej sobie takie znajdź. Niech dostarczą Twoje dokumenty w odpowiednie miejsca podczas zakładania firmy, niech doradzą termin rozpoczęcia działalności, niech podeślą potrzebne szablony umów.

Nie trać czasu na takie coś! To praca innych ludzi. Ważna i odpowiedzialna praca (której zaniedbanie może się dla firmy skończyć bardzo nieprzyjemnie), ale nie do wykonywania przez Ciebie.

Czy wiesz, jaki wpływ na twoje finanse ma zarejestrowanie firmy podczas trwania umowy o pracę? Czy wiesz, od kiedy podatek liniowy zaczyna być opłacalny albo kiedy należy Ci się „promocyjny ZUS" pozwalający oszczędzić sporo pieniędzy przez pierwsze lata działalności? Czy wiesz, co możesz odpisać od podatku oraz jakie dokumenty i gdzie musisz zanieść w przypadku choroby?

Nie? Naprawdę nie wiesz? I dobrze, ja też nie! Jeśli mam wątpliwość, pytam odpowiednią osobę. W tym przypadku: księgową. Otrzymuję odpowiedź, podejmuję działanie i zapominam o całej sprawie.

Niejednokrotnie słyszałem z tego powodu opinie, że jestem skrajnie nieodpowiedzialny, ale… Wolę skupić się na innych rzeczach. Nie da się robić wszystkiego.

Mnie taka przyjemność kosztuje 150 złotych miesięcznie. Niedużo, zważywszy na to, ile czasu dzięki temu oszczędzam.

⏎

Czy warto być freelancerem?

Jak zwykle, wszystko ma swoje dobre i złe strony. Ja parałem się freelancingiem przez prawie trzy lata. Różni klienci, różne projekty, wiele zleceń. Podobało mi się, dopóki... nie przestało. Jestem wdzięczny za te doświadczenia, ponieważ pozwoliły mi docenić inne formy pracy.

BOSS-less?

Ogromną zaletą freelancingu jest owo *free*! Nikt Ci nie mówi, co robić. Masz pełną swobodę. Robisz, co chcesz i kiedy chcesz. Niczego nie musisz. Po prostu: nie masz szefa! Cud-miód, żyć nie umierać?

Otóż niekoniecznie! Z jednej strony: faktycznie nie masz szefa jako takiego, ale coś robić trzeba. Samodzielnie pilnować wykonania zadań i samodzielnie się z nich rozliczać. W tym całym ambarasie zdecydowanie jest więc szef. To Ty!

Ale nie tylko! Może od strony formalnej nikt nie ma w umowie napisanego stanowiska nadrzędnego względem Ciebie, ale w praktyce: takich osób jest więcej. „Klient nasz pan!". To nawet gorzej niż szef. Bo od tych klientów, ich zadowolenia, rekomendacji i widzimisię zależysz. Kompletnie.

Nie daj się więc oczarować pięknej wizji „pracy bez szefa". To ściema, ułuda.

Techie

Irytuje Cię konieczność dłubania ciągle w tych samych frameworkach i bibliotekach? Ile można, co nie? Dobra wiadomość jest taka: jako freelancer to najczęściej TY decydujesz, jak stworzyć dany projekt. Możesz skakać między

technologiami, ile chcesz. Dzisiaj PHP, za miesiąc .NET, a potem Python. Tutaj Ruby, tam trochę Javy, a na koniec C++. Jeden moduł obiektowo, drugi funkcyjnie. Tutaj TDD, tam totalna partyzantka. Żegnaj nudo! Nigdy więcej monotonii! Brzmi pięknie, prawda? Tak, brzmi. I naprawdę możesz w ten sposób działać!

Ale prawdopodobnie nie będziesz. Wiesz, dlaczego? Bo nikt Ci za to nie zapłaci! Albo inaczej: bo teraz Ty za to płacisz, i to słono. Owszem, możesz zdobyć bardzo szerokie doświadczenie. Ale jak to przed sobą uzasadnisz? Może być ciężko!

Dzięki temu jednak zrozumiesz, dlaczego w pracy nie pozwalano na ciągłe eksperymenty. Każdy nowy element w Twoim programistycznym warsztacie to inwestycja. Może się zwrócić, jeśli spełni swoje zadanie. Ale też może wygenerować straty, wydłużając czas realizacji projektu.

Cienka linia

Prowadząc swój mikrobiznes (bo freelancing to biznes!), bardzo łatwo dać się porwać rwącemu nurtowi pracoholizmu. Bez umiaru, bez opamiętania. „Ja muszę! Jak nie ja, to kto? Jeszcze tylko ten jeden projekt!".

Widząc konkretne (i bezlitosne!) przełożenie godzin spędzonych na pracy na ilość kasy na koncie, czasami ciężko uzasadnić przed samym sobą chwile relaksu. Obijania się, regeneracji. Niby „nie samą pracą żyje człowiek", ale wiem, jak ciężko jest usprawiedliwić w tym przypadku „niepracowanie". Można wpaść w pułapkę i każdy moment spędzony poza pracą przeliczać na stracone zyski.

Ja potrzebowałem wielu miesięcy rekonwalescencji i przez długi czas musiałem uczyć się na nowo „sztuki relaksu". Wbrew pozorom, to nie jest takie proste.

„Biznesowy" koszmar

W miarę wzrostu popularności Twoich usług, zaczną spływać do Ciebie coraz to nowsze propozycje projektów. Jak odpowiesz na te kolejne zapytania? Wypadałoby zapoznać się z wymaganiami, specyfikacją. Zobaczyć, czy może to być ciekawa opcja. A może nawet wstępnie wycenić i oszacować czasochłonność?

Jasne! Tylko... kto to zrobi? No Ty. Kiedy? Pewnie wieczorami i nocami, bo w dzień przecież pracujesz nad projektami bieżącymi. A kto za to zapłaci? Zgadnij!

Nikt!

Analiza wymagań, tworzenie specyfikacji, a nawet sam proces wyceny to usługi, za które można pobierać wynagrodzenie. Ale nie oczekuj, że do pojedynczego, indywidualnego freelancera przyjdą klienci gotowi płacić za takie rzeczy.

Często otrzymasz maila z kilkoma stronami w PDF-ie, luźnymi myślami i paroma rysunkami. Oszacuj, wyceń. Wieczór masz z głowy, a i tak nie zrobisz tego rzetelnie – bo się nie da.

Oczywiście możesz od razu odrzucać takich klientów, jednak to zależy od Twojej sytuacji. Los może zmusić Cię do wybierania spośród właśnie takich zleceń. I klops.

To jest nagminne. Takie zapytania często nawet nie mają na celu faktycznego podpisania umowy, tylko zorientowanie się w zgrubnych kosztach. Albo poprawienie i uszczegółowienie specyfikacji. Za darmo.

Uważaj na to. Nieprowadzące do żadnych konkretów wiadomości to codzienność i z czasem nauczysz się je odróżniać od poważnych propozycji.

Nomad

Pracujesz, gdzie chcesz, kiedy chcesz i jak chcesz! Pięknie! Ja to uwielbiam. Ale uważaj: to może się skończyć NIEpracowaniem. To bardzo trudne.

Nawet pisząc tę książkę, udawałem się czasem na odludzie, gdzie ledwo dochodził zasięg sieci komórkowej. Wydanie niniejszej pozycji to jedno z moich największych życiowych osiągnięć i marzeń. A mimo to czasami nie potrafiłem znaleźć w sobie dyscypliny na tyle, by systematycznie i codziennie pracować nad książką. By skupić się w normalnym, domowym środowisku.

Przez lata, programując zdalnie, także borykałem się z tym problemem i poszukiwałem najróżniejszych rozwiązań. Było to szczególnie trudne w przypadku, gdy nie byłem rozliczany z godzin czy pomniejszych zadań, ale z całościowych

projektów. Chociaż jestem ekstremalnie wręcz zdyscyplinowany i zmotywowany do działania, nie zawsze przychodziło mi to z łatwością.

Czyli... warto czy nie?

Każde nowe doświadczenie nas wzbogaca. Mnie freelancerka nauczyła pokory w stosunku do „rynku". Zrozumiałem, że nie samym kodem żyje człowiek. Zacząłem też szanować pracę innych – pracę tak zwanego biznesu. Marketing, PR, sprzedaż, analiza czy wreszcie testowanie i wdrożenia – to wszystko jest bardzo potrzebne. Dopiero, gdy musiałem tym wszystkim zająć się sam, spadły mi klapki z oczu.

Ciężko jednoznacznie stwierdzić, czy próba takiej formy pracy będzie „bardziej na plus czy bardziej na minus". Ja jestem za eksperymentowaniem i badaniem różnych możliwości, nawet jeśli finalnie okaże się to klapą. Możliwe, że na długą metę się nie opłaci. Ale... może warto mimo to?

Wartość doświadczeń mierzy się inaczej niż ich opłacalność. Pewnie zarobiłbym więcej na etacie niż zarobiłem przez czas bycia freelancerem. I ten siwy włos na skroni może by się nie pojawił. Ale wtedy nie miałbym tak szerokiego spojrzenia na branżę IT. Być może nawet nie napisałbym tej książki? Kto wie.

Ciągnie Cię w tę stronę? Spróbuj! Najwyżej się wycofasz. W najgorszym wypadku po tej przygodzie Twój portfel będzie nieco chudszy niż przed nią. Ale nie traktuj tego jako stratę, tylko jako koszt lekcji. Inwestycję w przyszłość, zmianę perspektywy. Zrozumienie rynku, w którym się obracasz.

Pamiętaj tylko o moich poradach i rekomendacjach, żeby – w razie czego – nie było tragedii, tylko co najwyżej lekkie sparzenie.

Po raz kolejny warto zadać sobie pytanie: co najgorszego może się stać? I po raz kolejny okaże się, że przy odpowiednich przygotowaniach – nic.

To pytanie i ta odpowiedź zmieniają wszystko!

⏎

Skąd wziąć pierwszego klienta?

Masz kilka dróg.

Najbardziej oczywista z nich to: „z internetów"! Znajdziesz sporo portali z ogłoszeniami zamieszczonymi przez osoby poszukujące wykonawców pomniejszych zleceń. Możesz od razu zacząć od zagranicznych, takich jak upwork.com. Na początek może to być Twoje jedyne wyjście, i – godząc się na walkę z baaaardzo tanimi konkurentami – w ten sposób zdobędziesz bazę pierwszych klientów, zaczniesz zdobywać doświadczenie w biznesie. Zbudujesz sobie reputację, jakiś punkt startowy.

Dobrą sytuacją jest bycie poszukiwanym, a nie poszukującym. To moment, w którym klienci sami zaczynają do Ciebie przychodzić. U mnie rolę wizytówki i zaproszenia do nawiązania kontaktu pełnił (i pełni nadal, ale w innym zakresie) blog devstyle.pl. Sporo zleceń otrzymałem właśnie przez tamtejszy formularz kontaktowy. Rola marketingowca zaczyna pozycjonować Cię jako eksperta, budować wizerunek specjalisty. Tak, by z kolei sprzedawca – miał z kim pracować!

Zdobywanie doświadczenia – czyli realizacja projektów – to wspaniała okazja do jednoczesnego pokazywania się kolejnym potencjalnym klientom. Wystarczy dzielić się swoimi przygodami ze światem, przedstawiać realizowane pomysły. Tworzyć treści, w których zaprosisz publiczność do podążania za Twoim tokiem rozumowania. Przedstawianie swoich usług oraz umiejętności w takim kontekście działa cuda.

Pamiętaj, że w dzisiejszych czasach (prawie) każdy ma „jakieś" grono odbiorców. W tym gronie mogą znajdować się potencjalni klienci! Ilu masz znajomych na Facebooku czy innych portalach społecznościowych? Ile adresów e-mail jest zapisanych w swojej skrzynce? Daj światu znać, że posiadasz odpowiednie umiejętności i wystawiasz je na sprzedaż. Możesz się zdziwić, jak wiele zdziała jeden e-mail wysłany w odpowiednim momencie do odpowiedniej osoby.

U mnie to właśnie tak się zaczęło: od e-maila do znajomych. Akurat „ktoś znał kogoś szukającego programisty na zlecenie". Dobra wiadomość jest taka,

że pracę związaną z wysłaniem maila wykonujesz raz, a działa ona na Twoją korzyść przez długi czas. Zostajesz na radarze wielu osób jako „wolny programista do wzięcia" – i kto wie, co może z tego wyniknąć, nawet za kwartał czy rok?

Koniecznie znajdź klienta przed rezygnacją z dotychczasowej pracy. Ja zrobiłem dokładnie w ten sposób: miałem podpisaną umowę na duży projekt na kilka tygodni przed odejściem z etatu. I jest to najlepsza możliwa weryfikacja. Bo:

> **Jeśli nie potrafisz znaleźć klienta teraz,**
> **to w jaki sposób znajdziesz go za miesiąc?**

Gdy zmuszą Cię do tego okoliczności? No właśnie: nie znajdziesz. I dopiero będzie bieda!

Zmiana perspektywy

Jeżeli faktycznie na serio myślisz o tym, by zostać freelancerem, to bardzo polecam Ci książkę „Breaking time barrier" Mike'a McDermenta. To króciutka darmowa publikacja, potrafiąca zmienić perspektywę. Nie będę jej tutaj streszczał; po prostu znajdź, pobierz (jest darmowa) i przeczytaj koniecznie!

A więcej inspiracji znajdziesz na http://zawodprogramista.pl/ksiazki.

Umowa projektowa

Jeden z aspektów freelancerki jest tak bardzo istotny, że zasługuje na osobne omówienie. Pamiętaj:

> **Koniecznie spisuj umowę, zawsze!!!**

I to nie byle jaką! Nieważne, czy masz zrobić nowego Facebooka, czy postawić WordPressa dla kumpla z osiedla. Jeśli jakakolwiek suma pieniędzy przy

okazji zmienia właściciela, to musi zostać spisana umowa! I to nie ze względów prawnych. Nie ma: „dogadamy się w trakcie". Nie ma, że „jakoś to będzie". Nie będzie! Gdyby ludzkość nie uczyła się na własnych błędach, to jeździlibyśmy samochodami na kwadratowych kołach (ech, nieśmieszne). Naucz się na moich.

Poniżej znajdziesz listę zalet wynikających z umowy oraz całą masę zawieranych w niej informacji.

Bądź PRO!

Umowa to oznaka profesjonalizmu. A raczej odwrotnie: brak umowy to oznaka braku profesjonalizmu. Stworzenie umowy kosztuje nieco czasu, a czasami i pieniędzy. I bywa upierdliwe. Jednak to nie jest tylko zrzędzenie i przesadna formalność. To początek gwarancji udanej współpracy.

Spisanie umowy pokazuje, że poważnie podchodzisz do zlecenia. Nie tylko Tobie, ale też klientowi (kimkolwiek by nie był!) uświadamia, że to poważna sprawa. Że oto wchodzicie na ścieżkę biznesową. A to zupełnie co innego niż znajomość koleżeńska.

Umowa pozwala odseparować życie zawodowe od życia prywatnego, a to niezmiernie istotne. A dlaczego tak dużo piszę o tym życiu prywatnym, koleżeńskości i tym podobnych? Dlatego, że bardzo wielu freelancerów znajduje pierwszy projekt właśnie w gronie swoich znajomych. Ba, przecież sam do tego zachęcam.

Good times, bad times

W pełni rozumiem sceptyków twierdzących, że umowa to często niepotrzebny balast. Szczególnie w przypadku projektów zlecanych sobie po koleżeńsku. Też tak kiedyś myślałem... do momentu, w którym okazała się przydatna.

Owszem, w stuprocentowo udanym projekcie umowa może okazać się zbędna. Ale pamiętaj:

- **Umowę podpisuje się na złe czasy!**

W razie nieporozumień można do niej sięgnąć i szybko rozwiązać spory. Bez niej balansujesz na krawędzi.

Konkrety!

Czy wiesz, że ogromna większość projektów informatycznych przekracza czas, budżet lub kończy się porażką? Zabijają je niedopowiedzenia i konflikty zrodzone przez niejasności. Nie da się ich uniknąć w stu procentach, ale można je zminimalizować.

Kluczowe jest zrozumienie wzajemnych wymagań przed rozpoczęciem współpracy, a nie wypracowanie jednomyślności dopiero w trakcie. Wtedy jest już za późno.

Słowo pisane, ubrane w formalne punkty i paragrafy, wygląda inaczej niż luźny e-mail albo rozmowa telefoniczna. W umowie widać, o czym nie pomyśleliśmy. A czasami jest nawet jeszcze ciekawiej: widzimy, jak opacznie zrozumiała nas druga strona, interpretując nie do końca sprecyzowane wyrażenia niezgodnie z intencją.

Umowa to esencja wszystkich dotychczasowych ustaleń, trwających tygodniami czy nawet miesiącami. Ostateczna wersja negocjacji i wzajemnych oczekiwań. Spotkania, maile i telefony przekształcone w zrozumiałe, wyzute z emocji podpunkty.

Specyfikacja

Obowiązkowym załącznikiem do umowy będzie specyfikacja projektu. Specyfikacja zawiera – jakżeby inaczej – wyspecyfikowane cechy Twojego zlecenia. Samo określenie „coś jak Facebook, tylko czerwone" to za mało. A – uwierz mi – dostawałem propozycje zleceń tego typu.

Specyfikacja powinna zawierać dokładny opis każdej funkcji systemu. Makiety ekranów w interfejsie użytkownika, nawigację między poszczególnymi ekranami. Jest to efekt analizy wymagań klienta, a to zupełnie osobny etap projektu, odrębny od jego realizacji. Klient może się tego podjąć sam, możesz zrobić

to Ty, a może się tym zająć ktoś inny. Przed rozpoczęciem prac musisz jednak dokładnie wiedzieć, co jest do zrobienia.

No to Internet Explorer 6! Czy Android 4.0?

Tworząc aplikację webową, koniecznie określ, jakie przeglądarki i jakie ich wersje będziesz wspierać. Nie zgadzaj się na suport dla staroci, jak Internet Explorer 6, 7 czy 8. Nie zgadzaj się także na wspieranie „przyszłych wersji" przeglądarek. Nie wiesz, czy za rok nie wyjdzie jakiś kompletnie niekompatybilny potwór. W przypadku przeglądarek określ także, na jakim systemie operacyjnym będą one uruchamiane. Wiesz na przykład, że funkcja *zoom* w Chrome działa na Macu kompletnie inaczej niż na Windowsie?

Z kolei pisząc aplikację mobilną, wyspecyfikuj wspierane wersje systemu operacyjnego. To jest minimum Twojego zabezpieczenia. Idealnie byłoby mieć także listę wspieranych urządzeń, ale – przynajmniej w przypadku Androida – jest to raczej niemożliwe.

Podpisuj się pod wsparciem dla rozwiązań dostępnych dzisiaj, teraz. Tych, które faktycznie możesz przetestować przed zakończeniem prac.

Płatności

A jak z pieniędzmi? Ile rat? W jaki sposób opłacane? Pod jakimi warunkami wypłacane? Jaki termin płatności będzie widniał na fakturze?

Daty

W umowie nie można pominąć harmonogramu prac, wykraczającego poza „ma być gotowe za pół roku".

W harmonogramie wpisujemy konkretne terminy, zobowiązujące strony zobowiązują do wykonania określonych czynności. Którego dokładnie dnia zaprezentujesz pierwszą wersję? Kiedy wersja kolejna? Na ile dni przed „oficjalnym demo" udostępnisz (i w jaki sposób?) klientowi wersję „do poklikania"?

To wszystko trzeba przewidzieć, zaplanować, ustalić.

Ale daty obowiązują nie tylko Ciebie! Klient także pracuje nad systemem, prawda? Może się zdarzyć, że Twoja praca jest zależna od materiałów dostarczonych przez niego. Na przykład masz zrobić *back-end* i *front-end*, ale grafiki nie narysujesz. Zmieszczenie się w terminie wymaga dostarczenia grafik przez klienta kilka tygodni wcześniej. Tę datę także koniecznie umieszczacie w umowie!

⇨ *Kiedyś realizowany przeze mnie projekt wydłużył się, ponieważ klient dostarczył mi grafiki dwa dni przed ustaloną datą zakończenia prac nad systemem. Nie miał pojęcia, ile pracy wymaga dostosowanie roboczej wersji do ostatecznego wyglądu. A ja założyłem, że jest to oczywiste i nie dopilnowałem tego.*

Wina leżała po obu stronach i na szczęście wszystko sobie ugodowo wyjaśniliśmy. Lecz mogło się skończyć różnie. A wszystko przez głupie niedoprecyzowanie!

Każda data istotna z punktu widzenia projektu musi być zawarta w umowie. W przeciwnym razie – to tak, jakby jej nie było.

Support

Ile gwarancji dajesz na swoje rozwiązanie? A co dzieje się potem? Jakim kanałem mają być zgłaszane uwagi i błędy? Ile masz czasu na odpowiedź, a ile na naprawę? I co to znaczy „naprawa": czy tylko poprawiony kod w repozytorium, czy także wdrożona wersja na produkcji? Czy ten sam błąd pojawiający się kilka razy pociągnie za sobą jakieś dodatkowe kary? No i wreszcie: czym jest „błąd" i jak go odróżnić od żądania nowej funkcji w systemie, wymagającej dodatkowej opłaty?

Temat gwarancji jest na tyle istotny, że jeszcze go za chwilę rozwiniemy.

Hosting?

Projekt może śmigać na Twoim roboczym komputerze, ale klient na tym nie zarobi. Trzeba określić, co dzieje się po zakończeniu prac programistycznych. Kto

zajmuje się jego wdrożeniem i gdzie ono następuje? Kto ponosi koszty serwerów, certyfikatów, domen?

Klient może nie mieć świadomości, z ilu i jakich kroków składa się proces wytwarzania oprogramowania. Tak jak ja teraz odkrywam świat pisania i wydawania książek (co?? redakcja? skład? korekta? czyli nie wysyłam ot tak doca do drukarni?), tak samo Twój klient wcale nie musi wiedzieć, że „te kody trzeba gdzieś wdrożyć" i co to w ogóle znaczy.

Możesz wziąć to wszystko „na siebie". Zostać usługodawcą i utrzymaniowcem, zapewniając działanie systemu. Pamiętaj jednak, że wówczas to Ty odpowiadasz za awarie wybranego przez siebie hostingu. To na Ciebie polecą gromy w przypadku przerwy w działaniu.

Ja zawsze pomagałem przy wdrożeniach systemów, ale nigdy nie zgodziłem się na świadczenie usługi hostingowej. Wszak na awarie faktycznego wybranego przeze mnie hostingu nie mam żadnego wpływu! A te zdarzają się cały czas, nawet największym graczom.

Zespół

Nawet freelancer szuka czasami pomocy. Czy to zbyt szybko zbliżający się termin, czy specyficzne wymagania – nieraz w pojedynkę nie da się wszystkiego ogarnąć.

⇨ *Implementowałem kiedyś portal z osadzonym dość skomplikowanym komponentem do obróbki plików graficznych. Wówczas (dawne dzieje!) najsensowniejszym rozwiązaniem było użycie technologii Flash.*

Sam Flasha kompletnie nie znałem i absolutnie nie chciałem się go uczyć, więc zleciłem wykonanie tego małego zadania innej osobie. Mi realizacja tego widgetu zajęłaby pewnie około dwóch tygodni, a znajomy ekspert dostarczył mi rozwiązanie w ciągu trzech dni. Oczywiście musiałem za nie zapłacić, ale zaoszczędzony czas był zdecydowanie wart tej ceny!

Ciebie też to może czekać. Dlatego uważaj na zapisy w umowie wymagające realizacji całości projektu przez Ciebie osobiście. Najlepiej będzie od razu przygotować klienta na możliwość dalszego zlecenia części prac, żeby uniknąć nieporozumień.

Konsekwencje

Jakie są konsekwencje niewywiązania się z umowy? Czy przewidujecie kary, odsetki, inne reperkusje?

Każdy punkt umowy niech zawiera podpunkt: „w razie niewywiązania się z zadania…".

Wiadomo, że cała umowa daje podstawę, by pójść do sądu, ale to środek absolutnie ostateczny.

To jest piękno umowy: im więcej spraw ustalicie na początku, tym mniej powodów do kłótni będzie później. Opracujecie algorytm pełen ścieżek *if/else* i po prostu będziecie nim podążać. Bez pretensji. Wszystko wyjaśnione z góry, na zapas, na spokojnie. Zamiast później, w nerwach i gorącej atmosferze.

Więcej informacji…

Po więcej materiałów zapraszam Cię na stronę http://zawodprogramista.pl/prawo – znajdziesz tam dodatkowe treści na temat umowy, łącznie z wzorem umowy do pobrania.

Odpowiedzialność za oddany projekt

Data końca projektu jest dla freelancera bardzo ważna. Wtedy możesz zacząć czytać nową książkę, pograć na komputerze, pójść do kina. Ten dzień to Twoje święto.

Ale co dalej? Czy wiąże się to z możliwością wykasowania kodu, wycięcia z pamięci ostatnich tygodni, całkowitą wolnością i pełnym relaksem? Niekoniecznie.

Zmiany, zmiany, zmiany...

Mija kilka dni i przychodzi mail od klienta:

Ej, wiesz co, jednak po zastanowieniu chciałbym mieć przycisk OK w kolorze zielonym, a przycisk CANCEL – w czerwonym. To dla ciebie na pewno chwila, mogłabyś...?

albo:

Nie wiem, co się dzieje, ale gdy pokazywałeś system, wszystko działało tak szybko. Czemu teraz jest wolniej? Zupełnie inaczej się z tego korzysta. Zrób coś z tym.

albo:

Inaczej wyobrażaliśmy sobie raport pokazujący podsumowanie miesięcznej sprzedaży, trzeba to zmienić i dostosować do naszych oczekiwań.

albo:

Doszliśmy do wniosku, że SQL Server jest jednak dla nas za drogi. Czy możesz raz-dwa zmienić aplikację, aby korzystała z jakiejś darmowej bazy danych? Najlepiej takiej, która będzie działała na Linuxie?

Trochę to wszystko przerysowane, ale nie wyssane z palca.

Jak do tego doszło?

Reakcja na takie zachowanie może być różna. Jak zawsze – to zależy. Odpowiedź na podobną prośbę jest uwarunkowana wszystkimi okolicznościami, które do niej doprowadziły.

Czy podczas negocjacji i podpisywania umowy zgodziliście się, że po oddaniu projektu programista nadal będzie wprowadzał modyfikacje? Jeżeli tak, to w jakim zakresie?

Jak wyglądała współpraca na etapie tworzenia systemu? Czy ktoś go faktycznie testował w warunkach bojowych? Czy ostateczni użytkownicy mieli okazję się z nim zapoznać?

Obowiązkiem wykonawcy jest jasne zasygnalizowanie konieczności testowania systemu przez klienta podczas jego powstawania. Sytuacja, w której programista tworzy swoją radosną wizję programu, a weryfikacja zgodności efektu tych prac z oczekiwaniami zleceniodawcy odbywa się dopiero po zakończeniu prac i faktycznym rozpoczęciu produkcyjnego wykorzystania w codziennym życiu, jest trochę… chora. Klient musi wiedzieć, że system może działać inaczej u niego w firmie niż na komputerze programisty. Że powinien sprawdzić to działanie wcześniej niż dopiero po oddaniu projektu; że jedynie śledzenie na bieżąco procesu powstawania programu może prowadzić do całkowitej satysfakcji; że podejście: „masz tu moje luźne oczekiwania – zrób coś z tym i do zobaczenia za X tygodni" jest złe.

I najważniejsze: klient musi być świadom tego, że słowa „akceptuję" i „zatwierdzam" mają ogromne znaczenie. Ich luźne traktowanie, robienie z nich nic nieznaczących formułek na zakończenie spotkania czy mechanicznych odpowiedzi na maile z pytaniami lub sugestiami, najzwyczajniej w świecie dowodzi braku szacunku dla programisty.

Podejście: „przecież zawsze można zmienić zdanie" uwłacza osobie ślęczącej przed klawiaturą. Owszem, zdanie można zmienić, ale decyzje nie powinny być podejmowane i wdrażane w życie bez uprzedniego porządnego przemyślenia. I odpowiednich konsultacji, jeśli są konieczne.

I co wtedy?

Takie sytuacje ma regulować omawiana już umowa! Jeżeli współpraca przebiegała wzorowo, klient był świadom wagi wszystkich podjętych decyzji,

przeprowadzane przez niego testy faktycznie pomogły w podniesieniu jakości końcowego produktu, a mimo to chciałby wprowadzić drobną modyfikację, to... ja takie modyfikacje wprowadzałem. Nie był to priorytet na zasadzie „za godzinę dostaniesz nową wersję", ale zdarzało mi się poświęcić jeden wieczór na jakieś niewielkie zmiany. Po koleżeńsku, jako podziękowanie za udany biznes i potencjalną inwestycję w przyszłe kontakty.

Ale też nie można dopuścić do sytuacji, w której ten grzecznościowo użyczony czas jest marnowany; do tego, że dam palec, a wezmą całą rękę. Pomysły, zanim zostaną zaakceptowane przez klienta, powinny być przez niego przemyślane, uzgodnione ze wszystkimi zainteresowanymi. Wymagania nie mogą być wielokrotnie modyfikowane, a testowanie – traktowane jako obowiązek wyłącznie wykonawcy, czyli mój. W takim przypadku – przykro mi: twój czas minął! Koniec projektu to koniec projektu, ja swoje zrobiłem.

Jednoznaczna rekomendacja, co zrobić w takiej sytuacji, nie jest możliwa, bo każdy przypadek wygląda nieco inaczej i wymaga innego podejścia. Gdy brakuje regulacji w umowie, panuje całkowita dowolność. Dlatego warto zadbać o to wcześniej i takie scenariusze przewidzieć, zaplanować.

Do skutku!

Bardzo niebezpiecznym zjawiskiem jest przeświadczenie klientów, że zamówione rozwiązanie ma być dopieszczane przez wykonawcę aż do osiągnięcia kompletnej satysfakcji. Nie może być tak, że po terminie nadchodzi czas na „dostosowanie" projektu do faktycznych potrzeb. Podpisanie umowy na projekt nie czyni programisty „maszynką do kodowania" dostępną dla klienta na czas nieokreślony – aż do momentu, gdy program będzie w stu procentach odpowiadał jego wyobrażeniom.

Niestety, takie podejście spotyka się bardzo często, szczególnie przy projektach mniejszych, tanich, półamatorskich – a właśnie z takimi możesz mieć do czynienia na samym początku, gdy Twoje nowe zlecenia będą zależały głównie od rekomendacji.

Jasna, szczera komunikacja po raz enty! Przypieczętowana umową.

To fakt, wyobrażenia nieuchronnie ewoluują. Jest to całkowicie normalne zjawisko. Wymagania zawarte w specyfikacji dołączonej do umowy w ogromnej większości przypadków będą nieco (lub znacznie) odbiegać od tych stawianych w dniu zakończenia prac. Czy to jednak oznacza, że projekt uznajemy za „zaakceptowany" dopiero w momencie realizacji wszystkich nowych zachcianek? Nie! Nie zapominajmy, na co się umawialiśmy. Oczywiście specyfikacja początkowa nie jest żadną świętością i może podlegać modyfikacjom, ale trzeba mieć na uwadze, że to oryginalne założenia muszą być spełnione, a nie te wymyślone w trakcie.

Wszystko to sprowadza się do zrozumienia i zaakceptowania przez klienta prostego faktu: w dniu zakończenia projektu freelancer może zabrać się za następny.

Bo inaczej... jak planować pracę?

Błędy i utrzymanie

Co jednak w przypadku, gdy stworzone oprogramowanie najzwyczajniej w świecie nie działa do końca tak, jak powinno? Wracamy tu do punktu „testowanie".

Oprogramowanie tworzymy wspólnie, a nie jednostronnie. Do czego prowadziłaby praktyka zakładająca, że programista tworzy projekt, a następnie odpowiada za jego poprawne działanie do końca życia? Swojego lub projektu!

Coś takiego jak „oprogramowanie bez bugów" nie istnieje. Czy powinniśmy być gotowi, by na każde zawołanie poprawiać błędy w projektach oddanych pół roku temu? Rok temu? Trzy lata temu? Przecież doprowadziłoby to do paradoksalnej sytuacji, w której dla freelancera z każdym kolejnym zleceniem rośnie ryzyko darmowego, potencjalnie czasochłonnego powrotu do przeszłości.

Takie założenie to pomyłka. Bardzo ważne jest, aby obie strony w pełni to rozumiały i akceptowały konsekwencje rezygnacji z takiego podejścia.

Zamieńmy role i wyobraźmy sobie sytuację odwrotną. Wykonawca przychodzi do klienta kilka miesięcy po zakończonej współpracy i mówi: „Dobrze nam

się współpracowało, więc mam prośbę... Miałem nieprzewidzianą sytuację – zalało mi piwnicę – i wydałem na remont wszystkie pieniądze, które dałeś mi za ten projekt, co go dla ciebie kiedyś zrobiłem. Może mógłbyś mi dołożyć jeszcze trochę?"

Prawda, że brzmi absurdalnie? Dlaczego więc traktować mniej absurdalnie dokładnie taką samą sytuację, tyle że widzianą oczami programisty?

Gwarancja

Rozwiązaniem jest gwarancja, do której spisania zachęcałem w rozdziale dotyczącym umowy. Wtedy klient dokładnie wie, jakiej reakcji może się spodziewać po wykonawcy zlecenia i na co ma zwracać uwagę, obserwując proces powstawania systemu. Wreszcie: klient rozumie, jak ważne jest słowo „zatwierdzam"! Widzi, że często nie będzie od tego odwrotu.

Kiedyś do swoich umów freelancerskich dołączałem poniższą klauzulę. To nie jest oficjalna rekomendacja, ale pokazuje, jaką postać może przyjąć. Zawsze przed zastosowaniem „cudzych" porad zachęcam do konsultacji z prawnikiem. To rzekłszy:

Zleceniobiorca udziela trzymiesięcznej gwarancji na wytworzone oprogramowanie w zakresie:

– wydajności rozwiązania,

– poprawności logiki kierującej programem.

Przez problemy wydajnościowe rozumie się przynajmniej dwukrotnie wydłużenie średniego czasu reakcji programu na wybrane polecenia użytkownika w stosunku do pomiarów wykonanych przy oddaniu i akceptacji projektu przez Zleceniodawcę. Pomiar czasu reakcji oprogramowania na wybrane akcje użytkownika będzie wykonany na oprogramowaniu podłączonym do pełnej, aktualnie używanej przez Klienta bazy danych. Pomiary w momencie oddania projektu i zgłoszenia usterki odbędą się na takim samym komputerze, przy takim samym obciążeniu sprzętu.

Gwarancja nie obejmuje w szczególności:

- *wprowadzania udogodnień w interfejsie użytkownika aplikacji – interfejs użytkownika będzie modyfikowany zgodnie z sugestiami podczas implementacji systemu,*
- *modyfikacji logiki działania aplikacji, o ile zaimplementowane rozwiązanie jest zgodne z ustaleniami dołączonymi do umowy w postaci specyfikacji,*
- *modyfikacji utworzonych raportów, o ile zostaną one utworzone zgodnie z informacjami uzyskanymi podczas procesu analizy i implementacji; weryfikacja poprawności raportów powinna odbyć się przed zakończeniem prac nad systemem na podstawie danych historycznych zawartych w produkcyjnej bazie danych Klienta.*

Zleceniobiorca zobowiązuje się do wyeliminowania zgłoszonej usterki w ciągu 7 dni od stwierdzenia jej zasadności.

^D

Na koniec: Jak w pełni wykorzystać potencjał branży IT

Zbliżamy się do końca naszej wspólnej podróży. Co zrobić, byśmy rozstali się w możliwie najlepszych nastrojach? Najlepiej będzie chyba nakreślić zgrubny plan na przyszłość.

Niezależnie od tego, czy już jesteś w IT, czy dopiero się do nas wybierasz – to ogromna szansa! Programowanie programowaniem, ale wyjątkowa sytuacja, w której się znajdujemy, otwiera wspaniałe możliwości.

Dobra wiadomość: drogę do dowolnego celu można przedstawić w postaci powtarzalnego algorytmu.

Zła wiadomość: nie znam żadnego skrótu. To wymaga czasu, poświęceń i ciężkiej pracy. A także wyjścia ze swojej strefy komfortu.

Dodatkowo:

> **Niekoniecznie „się opłaca", lecz zdecydowanie warto.**

Rozwijaj się nieustannie, w różnych kierunkach

Bycie najlepszym programistą czy najlepszą programistką na świecie nie otworzy Ci wszystkich drzwi, do których zamarzysz sobie zapukać. Owszem, zdobycie czarnego dev-pasa to bardzo zacny kierunek! Poznawanie kolejnych języków, frameworków i narzędzi to jednak za mało.

Wiesz... Na pewnym poziomie wszystko i tak wygląda podobnie. Wymyślanie architektury, tworzenie kodu, optymalizacja. To prędzej czy później sprowadza się po prostu do walki – z wymaganiami, z klientami, z samym sobą. I wspomnianej wcześniej walki z maszynami.

Można spotkać się z rekomendacjami, aby nie uczyć się „wszystkiego" pobieżnie. Żeby zamiast tego zgłębiać jedną dziedzinę, aż dotrze się do jej sedna. Żeby stać się „ultraekspertem". Tak, to prawda. Warto być kimś takim.

Do tego wniosku doszliśmy zresztą w rozdziale dotyczącym programistycznej specjalizacji. Ale jak łatwo jest nałożyć klapki na oczy i zapomnieć o poniższej poradzie, która pojawiła się w tym samym rozdziale:

> **Know something about everything and everything about something.**
> ***Thomas Henry Huxley***

Tym razem przytaczam tę mądrość w nieco innym kontekście. Rozwój wszerz, a nie tylko w głąb, daje ogromne możliwości! I... chodzi nie tylko o kodowanie!

Prawie od początku swojej kariery prowadzę blog devstyle.pl. Ewoluuje, rozwija się. Jego prowadzenie przez tyle lat było dla mnie prawie drugim etatem, a jednocześnie – największym hobby.

Dodatkowo pozwoliło mi zaistnieć w świadomości ogółu polskiej społeczności programistycznej. Także w Twojej świadomości! Niby taka prosta rzecz: ot, blogasek – ale ile możliwości! A także – ile stresu, nerwów, kosztów, walki z samym sobą. Ale warto.

Dzięki temu zaistnieniu wskoczyłem w rolę prelegenta na wielu, wielu programistycznych konferencjach. Prowadziłem także na scenie wywiady. Kilkukrotnie wcieliłem się nawet w rolę konferansjera prowadzącego całą konferencję! Ba, specjalnie na tę okazję kupiłem trampki. Ha, ha.

Wymagało to masy cierpliwości i ogromnej determinacji. Zaledwie kilka lat wcześniej bałem się jakiejkolwiek komunikacji z ludźmi. Zdecydowałem jednak, że tak nie może być. Że to zmienię, choćby nie wiem co!

Występowanie na konferencjach to dokonanie, z którego jestem najbardziej dumny. Najwięcej stresu mnie ono kosztowało, było najtrudniejsze. Na scenach nauczyłem się mówić do dużych grup ludzi, a co najważniejsze – nauczyłem się improwizować.

Nigdy nie jestem w stu procentach przygotowany, nigdy nie mam wszystkich zdań wyrytych na pamięć. Po prostu: wskakuję na scenę i mówię. Kiedyś było to dla mnie nie do pomyślenia. Teraz przychodzi naturalnie. I bardzo, bardzo się przydało na następnych etapach.

Kolejna inicjatywa: podcast DevTalk.pl. Dzięki blogowi i konferencjom zbudowałem wielką sieć kontaktów, więc nigdy nie było problemu ze znalezieniem chętnych do nagrania rozmowy. Ale ta inicjatywa miała w tle inne zadanie. Dzięki prowadzeniu rozmowy z nieznajomymi podszkoliłem się w trudnej sztuce... rozmowy właśnie! Nigdy nie będę w tym mistrzem, ale potrafię porozmawiać choć trochę!

Wstyd się przyznać, ale kiedyś, gdy mnie ktoś zaczepił na korytarzu na konferencji, zdarzało mi się udawać konieczność natychmiastowej wizyty w kibelku. Żeby uniknąć „dziwnej" rozmowy! Od założenia podcasta nie musiałem tego robić ani razu. Choć nagranie pierwszych odcinków było bardzo, bardzo trudne.

Widzisz tutaj sens i ukryty cel?

Wymienione aktywności splotły się wreszcie w piękne podstawowe umiejętności, których tak bardzo mi wcześniej brakowało. Dzięki nim cieszę się prawdziwą wolnością. To nie jest nie wiadomo co i większość ludzi rodzi się z tymi zdolnościami. Ja musiałem je zdobyć, wyrwać losowi własnymi pazurami.

Przez kilka lat wykorzystywałem te nowo nabyte umiejętności do prowadzenia szkoleń. To niesamowicie opłacalne zajęcie, dzięki któremu mogłem zrezygnować z codziennej pracy, by kontynuować rozwój na wymarzonych frontach.

Blog dał mi pozycję eksperta; konferencje – umiejętność publicznego przemawiania prosto z głowy, improwizując, choćby przez cały dzień. A dzięki umiejętności „prowadzenia rozmowy", nabytej podczas nagrywania podcastu, udawało mi się nawiązać przyjazne stosunki z grupą szkoleniową.

Wilk syty i owca cała, a ja mogłem myśleć o zarabianiu na utrzymanie przez niecały tydzień roboczy w skali całego miesiąca. Resztę poświęcałem na przygotowania do dalszych działań i regenerację po dekadzie ciężkiej harówy.

Moja droga to tylko przykład. Pokazuje jednak, jak ważne jest otwarcie się na coś więcej niż tylko zera i jedynki. Twoja ścieżka może być zupełnie inna, ale wspólny mianownik da się wyciągnąć:

> **Boisz się czegoś? Zrób to!**
> **Nie umiesz czegoś? Zrób to!**
> **Wreszcie: chcesz coś zrobić? Zrób to!**

Nie mów, że nie masz czasu. Praca, dzieci, dom – kiedy to wszystko robić?!

Dopiero co zdradziłem Ci rozwiązanie tego problemu: kto powiedział, że masz spędzać w pracy czterdzieści godzin tygodniowo? Niech Twoim priorytetem

na najbliższe lata będzie redukcja etatu. Nie jest to proste, ale zyskasz tak wiele, że warto poświęcić się temu celowi.

On zmieni Twoje życie.

Ale to przecież oznacza mniej kasy, czyż nie? Niekoniecznie (pamiętasz rozdział o negocjowaniu podwyżki?). Ale nawet jeśli tak to... co z tego?

Debugguj życie. Stale.

Często pytamy swoje dzieci: „kim chcesz zostać, jak dorośniesz?". A gdyby tak zapytać osobę dorosłą: „kim chcesz być w chwili swojej śmierci"? Co będzie wyryte na Twoim grobie?

Za trzydzieści lat... za pięć lat... za miesiąc!... Czy chcesz robić to samo, co robisz dzisiaj?

Zadawałem sobie takie pytanie po kilku tygodniach w każdej pracy etatowej, dawno temu. W każdym wypadku odpowiedź brzmiała „nie". I za każdym razem właśnie wtedy podświadomie odchodziłem z tamtego miejsca. Wtedy nie zdawałem sobie z tego sprawy, ale teraz to rozumiem.

Zadałem sobie takie pytanie po kwartale siedzenia przy znienawidzonym systemie. Czy chcę na swoim nagrobku widzieć tabliczkę z napisem: „poległ, dłubiąc w Sharepoincie"? Nie chciałem.

Zadałem sobie takie pytanie po dwóch latach fajnej pracy, w której mogłem się rozwijać w różnych kierunkach. Ale okazało się, że niekoniecznie tych, które mnie w tamtej chwili nęciły najbardziej.

To ćwiczenie dla wszystkich!

Może jesteś team leaderem, eksprogramistą, kochasz programowanie, ale jedyną drogą awansu były: *dress code*, Excel i spotkania? Znamy to, prawda? Pisałem o tym we wcześniejszych rozdziałach.

Może masz własny biznes, prowadzony z sukcesem i poważaniem, ale Ci się po prostu znudziło? Albo zabija Cię harowanie w nim po piętnaście godzin na dobę? Freelancerka nie taka różowa, jak ją malują?

Albo jest jeszcze gorzej, i te piętnaście godzin dziennie wyrabiasz na cudzy rachunek?

Może właśnie ciągnie Cię do IT, do programowania?

Albo inaczej: zamiast X lat doświadczenia zdobywasz jeden rok doświadczenia X razy, i Cię to męczy?

A może po prostu jesteś młodym bogiem, jak ja!? I tak naprawdę możesz wywrócić w swoim życiu kompletnie wszystko do góry nogami. Bo, tak obiektywnie rzecz biorąc, naprawdę możesz wszystko.

Perspektywa zmienia się całkowicie, gdy uświadomisz sobie, że:

> **Każda decyzja jest odwracalna...**
> **...więc co najgorszego może się stać?**

Zadałem to pytanie w jednym z odcinków swojego vloga. Otrzymałem potem wprost lawinę korespondencji. Olbrzymia grupa ludzi poczuła się... odblokowana! Odblokuj się i Ty!

Przez całe życie słyszymy: „uważaj na to", „unikaj tamtego", „zachowuj się odpowiedzialnie", „życie takie jest", „nigdy nie będzie idealnie", „i tak nie masz najgorzej"... A co, jeśli ja chcę to idealne, wymarzone życie? Kiedy mam to robić, jeśli nie teraz? Kto ma tak żyć, jeśli nie ja?

Daj sobie chwilę na refleksje. Niech z tyłu głowy zacznie kręcić się trybik obmyślający Twój sposób na generowanie satysfakcji. Nie musi to być od razu plan przejęcia kontroli nad wszechświatem – ale chociaż nad małym jego kawałkiem. Twoim kawałkiem.

„Czy za trzydzieści lat nadal chcesz robić to, co robisz dzisiaj?". To nie jest żadna kołczerska zagrywka. To poważna kwestia. Niewiele osób się nad tym zastanawia.

Czy kurs Twojego życia jest taki, jaki faktycznie chcesz, aby był? Czy wszystko jest idealne? A jeżeli nie, to co możesz zrobić, aby to zmienić?

U mnie lata temu na całe moje przyszłe życie pozytywnie wpłynęło uświadomienie sobie tego:

> **Kupujemy rzeczy, których nie potrzebujemy,**
> **za pieniądze, których nie mamy,**
> **żeby zaimponować ludziom, których nie lubimy.**
> ***Dave Ramsey***

Wrócę do jakże odkrywczej prostej prawdy: kasa to nie wszystko! Coś dziwnego stało się w naszej branży. Bardzo często – szczególnie na początku drogi – to jedyny wyznacznik ludzkiej wartości. Jedyny cel. Miernik jakości życia i... jakości specjalisty!

Ale w tym momencie pozwolę sobie na jedyny wulgaryzm w niniejszej książce:

> **Z młodych gniewnych wyrosną starzy, wkurwieni.**
> ***Jonasz Kofta***

Wiem to. Znam to. Byłem tam.

Praca programisty nigdy nie uczyni z Ciebie prawdziwego bogacza. Nie skosisz milionów, naparzając w klawiaturę. Nigdy nie będzie dość! Ta gonitwa się nie kończy.

Jedyne wyjście to po prostu zatrzymanie się. I spacer do zupełnie innej mety.

Oszczędzaj

Tak, kasę! Ale banał, prawda? Niekoniecznie! Ten temat także już poruszaliśmy, ale to tak bardzo ważne!

Spotykałem młodych programistów, zarabiających trzykrotność średniej krajowej, a brakowało im na opłacenie ZUS-u! Co tu poszło nie tak? Odpowiedź jest naprawdę bardzo, bardzo prosta: żyj poniżej swoich możliwości finansowych.

Programista niekoniecznie zarabia kokosy, ale raczej biedy nie klepie. To po prostu trzeba wykorzystać!

> **Wydając całą pensję, wydajesz swoją niezależność.**

Oszczędzanie nie jest proste. Może być bolesne. Szczególnie, gdy obserwujesz, jak całe otoczenie spełnia swoje przyziemne marzenia.

Ja sam przez ponad dwa lata marzyłem, dzień w dzień, o jednym konkretnym samochodzie. Zamieniło się to prawie w obsesję. Codziennie oglądałem jego zdjęcia w internecie, czytałem recenzje, śnił mi się on! Miałem wystarczającą ilość pieniędzy na koncie, ale... samochód nie da długotrwałego szczęścia.

W końcu kupiłem go sobie, ale czekałem na ten moment latami. To była nagroda za wytrwałość, cierpliwość i powściągliwość.

„Arystokracja IT"? Co to za arystokracja, która nie przeżyje miesiąca bez wypłaty?

Ryzykuj

Chciałbym napisać tutaj o czymś innym, ale niestety kolejny krok jest także poniekąd związany z pieniędzmi. One często nakładają nam kajdany i powstrzymują przed szukaniem własnej ścieżki. Ale jest też druga strona medalu: odpowiednie zarządzanie kasą pozwala rozwinąć skrzydła!

Sam podejmowałem wiele ryzykownych decyzji. Całkowicie nieopłacalnych, z zewnątrz wyglądających na głupie. Ba, nawet obiektywnie głupich!

Ale żadnej nie żałuję. Przez lata starałem się dotrzeć do stanu, w którym chwilowo znajduję się teraz. Robię dokładnie to, co chcę robić. I zminimalizowałem robienie tego, czego robić nie chcę.

Ten swój wymarzony samochód przez bardzo długi czas, codziennie, zamieniałem na wolność i swobodę. Dodatkowo czułem, że co się odwlecze, to nie uciecze.

> **Rome ne s'est pas faite en un jour.**

(Wierz lub nie, ale właśnie w tej chwili, podczas pisania tych słów włączyła mi się piosenka *Rome wasn't built in a day* zespołu Morcheeba!).

Mogłem za to podejmować decyzje dziwne. Szalone. Nie do końca odpowiedzialne. Korzystałem z tego, że w końcu nadszedł mój czas. Wszystkim życzę

takiego poczucia. Takiej możliwości. Bez strachu, bez oglądania się za siebie. Jest takie rosyjskie przysłowie:

> **Kto nie ryzykuje, nie pije szampana.**

Szampanami opijałem się w przeszłości i nie to mnie już pociągało. Wyszukaj w internecie pojęcia „procentografia"; nie odważyłem się zawrzeć niektórych swoich opowieści w książce. Niezależnie od motywów: pięknym uczuciem jest możliwość robienia tego, co się chce robić.

Mam jedną radę: nie pytaj nikogo o opinię na temat swoich planów. Nic Ci to nie da. To Ty masz wiedzieć, co chcesz robić. Inni nie mają pojęcia, co dokładnie dzieje się w Twojej głowie, co tam od lat kiełkuje. Rób to, co uważasz za słuszne.

> **Proś o ocenę efektów, a nie zamiarów.**

Ryzyko to jedyny krok, z którym musisz trochę poczekać. Całą resztę zacznij wdrażać już dzisiaj. Dokończ książkę, weź głęboki oddech i… do dzieła!

Kontroluj zniszczenia

W naszym zawodzie bardzo łatwo jest stanąć nad przepaścią… a potem zrobić kilka kolejnych kroków. Ja wykonałem te dodatkowe kroki i roztrzaskałem się o skały. Wielu moich znajomych – także. Ty nie musisz! Zobacz to: http://zawodprogramista.pl/spowiedz.

Wieloletnie nieodpowiedzialne szastanie zdrowiem miało poważne konsekwencje. Miesiącami dochodziłem do siebie, byłem na etapie oceny zniszczeń i regeneracji. Teraz jest już całkowicie OK.

Jeszcze niedawno miałem jednak wyjątkową szansę na spokojne realizowanie wszystkiego, co od dawna chciałem zrealizować, a mimo to nie byłem w stanie stuprocentowo się tym cieszyć. Wywróciłem swoje życie do góry nogami w poszukiwaniu szczęścia – ono było już, w zasięgu wzroku, na wyciągnięcie ręki… ale jeszcze musiałem się trochę postarać. Jeszcze i jeszcze. Bo coś ciągle było nie tak.

W końcu odkryłem, o co chodzi. Poświęciłem temu tematowi tygodnie researchu, analiz i rozmyślań. Brzmi głęboko, i to naprawdę było głębokie! A okazało się... dość banalne. Obserwacja była o tyle zaskakująca, że właściwie oczywista. Powinienem sam to wiedzieć od dawna, ale byłem ślepy.

Przez te wszystkie lata zarzynałem się, harując jak wół. Ośmiogodzinny sen zdarzał się sporadycznie, może raz w miesiącu. Bo to przecież marnowanie czasu.

Marnowanie niezbędne. Poruszyłem już ten temat we wcześniejszych rozdziałach, ale muszę do niego wrócić także w tym miejscu. Poczytaj jeszcze więcej pod adresem: http://zawodprogramista.pl/rano.

Doceń chwilę obecną. Zachowaj umiar. Nie daj się zwariować.

Jeżeli zaciągasz u siebie dług, to rób to świadomie.

Ja zaciągnąłem dług ogromny, nie zdając sobie z tego sprawy. I spłacałem go z odsetkami przez blisko rok.

W tej książce nie daję rekomendacji, których nie jestem absolutnie pewien. Żeby nie narobić szkody. Dlatego też dodatkowo – na sam koniec – odsyłam Cię pod adres: http://zawodprogramista.pl/psycho. Tam znajdziesz coś, czego nie znajdziesz nigdzie indziej. Oby Ci się nie przydało. Ale jeśli jednak, to *I've got your back*!

Nigdy nie jest za późno...

Lata temu, po kilku pierwszych doświadczeniach w różnych firmach, doszedłem do smutnego wniosku: jestem skazany na programowanie. Nigdy nie będę robił niczego innego, nawet jeśli mi to zbrzydnie.

Na szczęście życie to zweryfikowało. Okazało się, że byłem po prostu młody i głupi. Myliłem się.

Mogę być Twoją inspiracją, jeżeli chcesz. Kto by nie chciał takiej muzy, co nie? Nie namawiam Cię jednak do rzucania papierami i szukania nowej pracy ot tak. Albo do zmiany branży bez refleksji czy do bezmyślnego zakładania kolejnego

durnego start-upu – fakapu. Ani do wysyłania do szefa maila o temacie: „dawaj więcej hajsu albo nara!".

Wszystko trzeba przemyśleć, przygotować. Warto poświęcić na to nawet lata. Ale w końcu nadchodzi czas na zadanie sobie pytania: co dalej? A stąd już prosta droga do odpowiedzi na kolejne: co teraz?

U mnie początkiem drogi było programowanie. Gdzie jest jej koniec? Nie potrafię przewidzieć. I ogromnie mnie to cieszy!

U Ciebie może być odwrotnie! Programowanie może być finałem, bo – odpowiednio zarządzane – jest wspaniałą przygodą.

Żyj i daj żyć innym.

Życzę Ci powodzenia na Twojej własnej, autorskiej, niepowtarzalnej ścieżce.

> **Celów nie wyznacza się, by cieszyć się z wyników.**
> **Cele wyznacza się, aby celebrować drogę do nich.**

Podziękowania

Proces powstawania tej książki uświadomił mi, jak dużo zawdzięczam wielu osobom. Jestem niezmiernie wdzięczny losowi, że rzucił nas w tę samą czasoprzestrzeń.

Po pierwsze, dziękuję mojej Żonce i mojej Córeczce. Za cierpliwość, zrozumienie i wsparcie. Niełatwo musi być ciągle wysłuchiwać coraz to nowszych pomysłów i recenzować wariackie inicjatywy. Zasypiać w samotności, bo nagranie podcasta, bo webinar, bo lajw. Chować się w pokoju obok, bo „ćśś, tatuś teraz nagrywa". Patrzeć, jak wspólna kasa ciągle leci do wora z naszywką „coś w końcu wypali!". Cóż... Długo nam to zajęło, ale już się nauczyliśmy: trzeba samemu być szczęśliwym, by uszczęśliwiać innych.

Dziękuję Jakubowi Gutkowskiemu. Gutek, razem dojrzewaliśmy jako programiści. W tym samym czasie stawialiśmy pierwsze kroki z blogami, występami scenicznymi, animacją społeczności. Obaj liżemy blizny po Sharepoincie, pamiętamy „budowanie samolotu w locie". Wiemy, co to nieprzespane miesiące i mordercza żonglerka priorytetami. Powodzenia na wszystkich frontach, *dev-mate*!

Dziękuję Basi Fusińskiej. Wzajemnie dodawaliśmy sobie powera, motywujące kopy i bezcenne rady na prelegenckiej ścieżce. Pokonywanie barier i walka z własnymi słabościami u boku takiej zawodniczki to zaszczyt! Ciężko mi zdecydować, którym Twoim wielkim planom mam kibicować. Trzymam kciuki za wszystkie, inspiruj nas dalej!

Dziękuję Michałowi „Szaffiemu" Szafrańskiemu (autorowi książki „Finansowy Ninja") i Mateuszowi „JavaDevMatt" Kupilasowi (autorowi książki „Junior Developer"). Panowie, Wasze doświadczenia z self-publishingiem umożliwiły mi truchtanie po wydeptanych już ścieżkach i unikanie zgubnych pułapek. Wasza transparentność, szczerość i uczciwość inspirują i motywują. Michał, Twoje *case studies* to kopalnia wiedzy, objawienie. Podbijajcie świat dalej!

Dziękuję Andrzejowi Krzywdzie za otwarcie mi oczu. Dzięki Tobie zrozumiałem, że „każdy jest sprzedawcą" i że to dobrze, a nie źle. Bo „kiszenie w sobie produktów" to działanie na szkodę – i swoją, i potencjalnych nabywców. Ile się

od Ciebie nauczyłem, to głowa mała! No i patrz, już nie będziesz miał powodu, by wypchnąć mnie na Mazurach z łódki przy Kalibrze albo Behemocie!

Dziękuję Mirkowi Burnejko za inspirację do działania. W dużej mierze dzięki Tobie wszedłem na jutuby, które najwidoczniej czekały na mnie od dawna! A bez tego… nie byłoby niniejszej książki. Planowałem ją od wielu lat i dopiero ten krok spowodował konkretne działania. *Done is better than perfect*! I do przodu!

Dziękuję Julii „Marcell" Górniewicz. Julio, pewnie nigdy tego nie przeczytasz, ale jedna linijka z Twojej piosenki rozpoczęła lawinę. Dekadę temu zrozumiałem, że *passion is something that likes to be conquered*. Tak niewiele trzeba, by czarne balony nabrały kolorów. Wdzięczność na zawsze.

Dziękuję moim hojnym Patronom, aktywnym Czytelnikom i Czytelniczkom, zagorzałym Słuchaczom i Słuchaczkom, inspirującym Widzom, ambitnym Uczestnikom i Uczestniczkom konkursu Daj Się Poznać… Całemu wspaniałemu *community* zgromadzonemu wokół „imperium devstyle". Wspólnie tworzymy bezpieczne, radosne i potrzebne miejsce w polskim internecie. Bez Was to wszystko nie miałoby sensu.

Dziękuję wszystkim, z którymi miałem okazję pracować. Wiem, że czasem byłem fiutem, mrukiem, PITA. Tym, który nie posiedzi w pracy kwadrans dłużej, nawet gdy „naprawdę trzeba". Dzięki naszym wspólnym przygodom i trudnym doświadczeniom jestem teraz tu, gdzie jestem. I dzisiaj, w tej chwili, nie zamieniłbym tego miejsca na żadne inne. Każdemu z Was życzę dotarcia do swojego celu.

I finalnie dziękuję… Tobie! Za wsparcie mnie przez przeczytanie tej książki. Za dotarcie aż do tego miejsca. Za feedback, którego nieuchronnie w ten czy inny sposób mi udzielisz. Być może zaliczasz się do jednej z powyższych grup. A jeśli nie, to… serdecznie zapraszam!

Kurde, wyszło jak list pożegnalny. Ale to nic. Przynajmniej jest szczerze.

Howgh, rzekłem!

Maciej „procent" Aniserowicz, devstyle.pl

Białystok, 14 września 2017 roku


devstyle.pl