ЕЛЕНА ПРАВДИНА

ГОВОРЯТ, В IT МНОГО ПЛАТЯТ

КАК ПОСТРОИТЬ УСПЕШНУЮ КАРЬЕРУ РАЗРАБОТЧИКА, ОСТАВАТЬСЯ ВОСТРЕБОВАННЫМ И НЕ ВЫГОРАТЬ

БОМБОРА
ИЗДАТЕЛЬСТВО
Москва 2023

УДК 004:159.92
ББК 32.973+88.4
П68

Правдина, Елена Владиславовна.

П68 Говорят, в IT много платят : как построить успешную карьеру разработчика, оставаться востребованным и не выгорать / Елена Правдина. — Москва : Эксмо, 2023. — 384 с. — (Книги-драйверы).

ISBN 978-5-04-119275-4

Льюис Кэрролл как-то сказал: «Здесь приходится бежать со всех ног, чтобы только оставаться на месте, а чтобы куда-то попасть, надо бежать как минимум вдвое быстрее».

ЭТО МИР IT. МИР, ГДЕ ТЕХНОЛОГИИ РАЗВИВАЮТСЯ БЫСТРЕЕ, ЧЕМ ВЫРАСТАЮТ НОВЫЕ ПОКОЛЕНИЯ.

Елена Правдина — профессиональный разработчик с опытом работы более 10 лет, включая порталы Videomore.ru, СТС, Wifire TV Lite, video.khl.ru, создатель HTML5-плеера для КХЛ, ОТР, СТС, а также Smart TV-приложений, фронтенд-разработчик Яндекс. В своей книге она с юмором и мудростью, приобретенной за годы руководства в компании «Инвентос», рассказывает:

— Когда и с чего начинают свой путь разработчики.
— Какие типажи успешны в отрасли и какие черты стоит в себе воспитать.
— Как находить и восполнять пробелы в технических знаниях.
— Что лучше: фриланс или жизнь на зарплату, офис или удаленная работа.
— Как выстраивать отношения в личной жизни и в рамках проекта.
— Как стать тимлидом, эффективно руководить, «побеждать» на собеседованиях, бороться с выгораниями и развиваться в мире, где, как говорят, много платят.

УДК 004:159.92
ББК 32.973+88.4

ISBN 978-5-04-119275-4

СОДЕРЖАНИЕ

ВСТУПЛЕНИЕ

«Дверь не закрыта!»

Расслабленный тон демонстрации многообещающего приложения по управлению умным домом моментально испарился. Обновляя экран телефона с только что выкаченной версией и вглядываясь в статус своего жилища, глава разработки за столиком в баре напряженно пытался осознать — дело в его забывчивости или же в просто-напросто севшей батарейке домашнего дверного датчика.

Знание внутреннего устройства приложения позволило тогда спокойно продолжить вечер. Квартира была надежно заперта. И если вы хотите сэкономить деньги на такси или время на общение со службой поддержки из-за очередного странного поведения вашего заказа — порой не лишним будет понимать, существует ли проблема в действительности или это всего лишь очередной баг.

Сейчас IT все теснее переплетается с повседневным миром вокруг. Либо вы уже пишете код, задаваясь вопросами своей карьеры, выгорания и счастливой пенсии,

либо же знания о том, как устроена сфера программирования, как работают приложения и создаются магические строки, понадобятся вам в обозримом будущем. Тоньше настроить под себя умный дом. Отдать ребенка в «правильный» IT-кружок. Вовремя сменить бесперспективную или наскучившую профессию.

Эта книга — сборник ответов, рекомендаций и забавных историй. Она призвана помочь и направить по верному пути желающих начать, а для тех, кто уже начал — сохранить любовь к работе и профессионализм на долгие годы, отбросить сомнения и дать подсказки по всем основным этапам карьеры и жизни в IT. Грамотно расти. Быть ближе к изменениям, что проникают в наш быт. Создавать, оставаясь увлеченным разработчиком. Или хотя бы понимать. Больше, чем те, кто никогда не бывал за кулисами.

Я не известный человек — я обычный, самый настоящий разработчик, не испорченный излишним медийным шумом.

За первые 10 лет работы успела пройтись по всем тропам и поджидавших на них граблям карьеры фронтендера. Окончила с отличием профильную специальность. И да, правда, это никому не нужно. Начинала с завораживающего причастностью тогда и ставшего немодным теперь: работала в инженерной поддержке далеких от IT сотрудников библиотеки, собирала из коробок и настраивала десятки рабочих станций за день.

Писала запросы на невообразимом сейчас французском к стандартной специализированной БД для библиотечных систем. Верстала под легендарный Internet Explorer 6. Писала на порицаемом ныне PHP. Грешила фрилансом: немецкое модельное агентство еще держит мое имя в IT-команде, а лучшие скутеры и мотоциклы и по сей день продаются в моем родном городе сквозь написанные в студенчестве строки, с дизайном, достойным места в музее IT.

Трудилась в лучшей компании онлайн-интернет-вещания. Разрабатывала ТВ-приложения под Samsung/LG/Philips, когда Smart TV только зарождался на рынке. Писала видеоплееры для всем известных и высоконагруженных проектов. Реализовывала стандарты видеорекламы — подводные части ежедневно наблюдаемых вами десятков баннеров, пре-, мид- и пост-роллов. Делала простые сайты и сложные порталы. Создавала адаптивные — для каждого браузера, любого устройства — интерфейсы, борясь за каждый пиксель по макетам. Проклинала зоопарки напичканных багами мобильных.

Изведала радости и муки руководства фронтендерским отделом. Собеседовала, взращивала кадры, как могла, скатывалась в психологию, принимала уходы как личные потери — и вырастала, чтобы не принимать.

Попала в одну из самых желанных IT-компаний нашей страны.

Да, поработать за навеваемой модой границей не успела. Хотя порывалась. И даже ступала одной ногой. Просто поняла, что мне это не нужно.

Зачем я написала книгу? Затем, чтобы помочь всем причастным избежать ошибок на своем пути, идти к намеченным целям без поворотов на лишние тропинки и, конечно же, стать профессиональнее и счастливее.

И еще, чтобы показать ежедневную рутину, жизнь, какой ее видят профессионалы IT, вставая утром и ложась вечером. А чаще — ночью. Мир, которому отдано XX часов в сутки, отдано всё, что есть у нас: молодость и энтузиазм. Где мы существуем в светлое время суток. Что видим вокруг. Что думаем о мире. Как общаемся. Влюбляемся и выгораем. Находим смыслы и идем дальше.

Мне писали. Студенты профильных специальностей и опытные разработчики. Полицейские, экономисты и даже сборщики корпусной мебели. Все они находили в книге мотивацию продолжать. Подтверждение своим мыслям, способы выхода из сложных ситуаций. Понимание, что ждет их в IT. Всего один опубликованный на Хабре[1] пост привел к тысячам скачиваний. Искренне надеюсь — вы из тех, кто найдет для себя в книге что-то важное и полезное.

[1] Хабрахабр, habr.ru — новостной сайт с системой блогов, наиболее популярный в русскоязычном сегменте IT-сообщества.

Для кого эта книга

Для тех, кто хочет связать свою жизнь с IT, для начинающих и думающих, быть или не быть. Информация, очищенная от моды, онлайн-курсов и инфобизнеса.

Для студентов профильных IT-направлений, для студентов, желающих перевестись в IT из других специальностей, разочаровавшихся в текущем выборе. Понять свои первые шаги и узнать, что ждет вас на заре карьеры, а что лет так через двадцать.

Для тех, кто измучен рутинной работой, наскучившей или потерявшей актуальность в стремительно меняющихся реалиях. Кто, желая, но терзаясь внутри сомнениями и, быть может, не веря в себя, хочет попробовать в IT. Начать, сделать поворот рабочего кресла к настоящему программированию, тестированию, управлению — любым граням серьезной IT-разработки.

Для джуниор-, мидл-разработчиков и опытных профессионалов — как сборник советов по каждому этапу карьеры и очередной биографический нон-фикшн: посмеяться над историями, увидеть себя, задуматься, признаться себе, не согласиться со взглядами.

Структура книги

Мы все проходим определенные этапы на своем жизненном пути, равно как и на пути профессиональном. Трудно отделить их — влияние второго на первый в судьбе

достойного разработчика колоссально. На каждом из таких этапов свои цели и задачи, свои советы и опыт уже прошедших ранее по этому пути. Поделиться последними и вдохновить на новые свершения и призвана эта книга. Хочется верить, ее структура сформирует некий Advice Driven Behaviour — поведение, управляемое советами, по аналогии с TDD — Test Driven Development — известной всем разработчикам методологией, основанной на написании успешно проходящих тестов, а затем уже самого программного кода.

Книга поделена на части, согласно этапам IT-карьеры и жизни: от обучения и начала до профессионализма и руководства, выгорания и переосмысления. Название каждой части базируется на аналогии с основными стадиями создания и работы программного обеспечения. Части содержат разделы, посвященные важнейшим вопросам на данном этапе.

DESIGN — проектирование. Часть о периоде с момента зарождения цели о вступлении на путь IT до формирования четкой картины о шагах по ее достижению.

«Когда начать», *«С чего начать»*, *«Как выбирать информацию и отсеивать ненужное»* — в этих разделах разобраны проблемы места и времени старта, обучения, правильных источников. Так ли важен возраст и профильное образование? Какие книги помогут, а какие курсы стоит обходить стороной? На что обратить внимание, как выстроить мышление и почувствовать себя на шаг

впереди остальных. А это зачастую отличная мотивация, чтобы продолжать захватывать мир. По крайней мере, для нас, амбициозных программистов.

Мы знаем, что заказчик всегда меняет требования. Учесть все пути, по которым пойдет развитие продукта, — задача практически невыполнимая даже для разработчика с бородой, длиною во всю историю развития IT. Так что и здесь — о шагах самых первых, пока жизнь и ваши цели не внесут свои разумные коррективы.

WRITE — написание кода. Часть посвящена основному этапу создания своих навыков, активного изучения теоретических основ, периоду начала программирования на популярных и не очень языках, средах и фреймворках. В случае более-менее классического сценария развития событий — данный этап посвящен обучению в вузе.

Как найти вдохновение и проверить, есть ли оно у вас, насколько важны усердие и глубина, какие типажи успешны в IT и как использовать конкуренцию — раздел первый *«Базовый класс: основы успеха»* — о ключевых столпах достижений.

Факультативы, научная деятельность, спортивное программирование и увлечения — раздел второй *«Расширяем роли: смежные активности»* — о вариациях IT-пути и дополнительных занятиях, дающих необходимое сейчас в IT разностороннее развитие.

Личные отношения и сила их воздействия на ваш рост как человека и профессионала — раздел третий *«Первая*

весна Естественного интеллекта» — о судьбоносных формах влияния на начальных этапах.

COMPILE — сборка и запуск. Часть о начале работы, первых шагах по применению полученных навыков в промышленных масштабах. О разрыве обучения с реальностью и приведении их в гармонию.

Как поддержать страсть к разработке и не навредить будущей карьере — раздел первый *«Выбор сборщика»* — о выборе первого места работы.

За какие задачи браться, как не распыляться, что станет плюсом, а что скорее помешает дальнейшему росту — раздел второй *«Примеры бинарников»* — о выборе между мелкими и крупными компаниями, опенсорсом, фрилансом и заказами для знакомых.

Как вести себя в команде и к какому микроклимату стремиться — раздел третий *«Оптимизация поведения»,* об отношениях в коллективах.

RUN — основной этап пути. Часть обо всем, что способно помочь идти летящей походкой, с высоко поднятой головой и бокалом в руке. Вернее, ехать на вполне себе автомобиле. Про самую классическую программистскую жизнь.

На что обратить внимание в поисках лучшей работы — раздел первый *«Выбор движка»* — о серьезных проектах и компаниях.

Как управлять разработчиками — раздел второй *«Между 0 и 1: карьера и руководство»* — о становлении признанными лидерами и уважаемыми сеньорами.

Как выстроить крепкую и позитивную профессиональную атмосферу — раздел третий *«Личные взаимоотношения»* — о поведении с коллегами.

Что сделать для подготовки и как вести себя в ходе рекрутинга — раздел четвертый *«Собеседования»* — об извечных и порой диссонансных IT-интервью.

Сколько получают IT-шники — раздел пятый *«Цена себе: зарплата и справедливость»* — о критериях формирования зарплаты и путях её повышения.

Нужна ли вам релокация — раздел шестой *«Работа за границей»* — о немаловажных факторах при выборе между трудом на родине и за рубежом.

Из чего складывается профессионализм, как стать вызывающим уважение разработчиком — раздел седьмой *«Детали мастерства»* — о тонкостях успеха.

DEBUG — отладка. Часть о проблемах разработчиков со стажем. Многие сознательные и ответственные специалисты признаются, что уставали. Думали открыть свою мастерскую по изготовлению шкафов, или, скажем, автомойку. Посвятить себя фотографии. Да и, в конце концов, — выползти из шкуры «детей подземелья» на свежий воздух, под лучи солнца и ароматный ветер. Иными словами, бросить к черту всё и уехать в путешествие на накопленные остатки.

На что обратить внимание, когда кажется, что все известно и хочется перемен — раздел первый *«Поиск: проблемы дальнейшего развития»* — о частых ошибках профессионалов на пути к росту.

К чему приходят разработчики, с какими мыслями борются. Как выбраться, потеряв смысл, стряхнуть пепел и продолжить — раздел второй *«Профессиональное выгорание»* — о предупреждении выгорания, его признаках и борьбе с ним.

REFACT — приведение в порядок. Часть о построении более-менее гармоничных отношений накопленного опыта с внешним миром и полноценной жизнью.

Как найти и отсечь лишнее, установить границы — раздел первый *«Анализ»* — о поддержании баланса между жизнью и работой.

Какие проблемы нарастают в IT, и что ждет всех причастных в будущем — раздел второй *«Горизонт событий»* — о хобби, семье, отношениях, детях и жизненных векторах зрелых специалистов.

Куда уходят разработчики, дизайнеры и менеджеры — раздел третий *«Что кроме»* — об изменчивости мира и примерах успеха IT-специалистов в других профессиях.

Польза от прочтения

Вы узнаете, когда и с чего начинают свой путь в IT, как выбирать место работы и выстраивать отношения в жизни и на проектах. Что важно, а что лишь информационный шум, что даст вам фору, а что никогда не приведет к статусу высококлассного специалиста.

Поймете, какие типажи успешны в отрасли и какие черты стоит в себе воспитать. Как вырастают в раз-

работчики серьезных продуктов, и как живётся девушкам в IT.

Почувствуете, как стать тимлидом и руководить, «побеждать» на собеседованиях, бороться с выгораниями и развиваться. Где искать вдохновение, как попасть в компанию мечты и оставаться жизнерадостным разработчиком.

Найдете мысли, подсказки и, быть может, ответы по всем основным аспектам своего карьерного пути, чтобы продолжить существовать в IT профессионально и счастливо.

DESIGN

ОТ ЗАРОЖДЕНИЯ ИДЕИ ДОБИТЬСЯ УСПЕХА В IT-РАЗРАБОТКЕ ДО ПЕРВЫХ ДЕЙСТВИЙ

КОГДА НАЧАТЬ

Is there right time for?..

«Уже слишком поздно», «Смогу ли я?», «С моим экономическим факультетом?». Герои меняются: гуманитарный диплом, непрофильная профессия, цены ненужных курсов, возраст сомнений.

Одна напутственная фраза стартовых дней моего IT-обучения прочно врезалась в память:

«Ты никогда не сможешь соперничать с мужчинами — пока они будут писать код и совершенствовать свои навыки, ты будешь жарить котлеты».

Я вспоминаю ее и по сей день. То чувствуя на миг, что стою в шаге от принятия той безнадежности и смирения, то, и гораздо чаще, — упиваясь осознанием, что да, я смогла: избежать, добиться. И вы — никогда не допускайте первого.

Мой путь в IT начался классе в десятом, когда в сознании начал всё серьезнее вырисовываться вопрос поступления. Стоит отметить, что я выросла в семье, со-

стоящей по большей части из врачей, и по меньшей, но столь же значимой для меня, — преподавателей. С детства все разговоры, шутливые вопросы и представления о моем будущем витали вокруг профессий доктора и учителя. Да и я сама была искренне убеждена, что продолжу семейную традицию и стану хирургом или лором, как бабушка, акушером-гинекологом, как дедушка или, в самом близком и понятном мне случае, — терапевтом, как мама. Каждый из них отдавал всю свою жизнь профессии и становился одним из лучших в своем кругу, занимал руководящие должности: управлял отделением или был заведующим консультацией. Их путь хоть и был требующим массы сил и отдачи, но оставался прозрачным и понятным. Одна специализация, одно место работы, одни глубокие знания — и на всю жизнь. Каким любопытным и тяжелым для меня позднее станет осознание: в IT всё не так. Попытки проецировать профессиональный опыт другой сферы и иного поколения на представления о будущей жизни и работе добавили сложностей принятия на начальных шагах карьеры. В области, где за год твои навыки и знания могут устареть достаточно, чтобы перестать вызывать блеск в глазах рекрутеров, да и в глазах собственных, где появляются и уходят в прошлое фреймворки, мода на стили и пути решения одних и тех же ежедневных задач. Перемены происходят так стремительно, что спасают лишь гибкий ум, глубокое понимание базовых концептов и непрерывное, ежечасное, ежеминутное погружение, самообучение, чтение,

развитие — и практика, практика. Но, впрочем, обо всём по порядку.

Не помню точно, как и когда именно ко мне пришла мысль пойти на IT-шный факультет. Мой отец был причастен к той самой преподавательской части семьи. Кажется, однажды он принес домой списки всех специальностей единственного в городе технического вуза, где работал сам. И мой выбор пал на «Информационные системы». Тогда же началась подготовка.

Пожалуй, такое время серьезного старта — средние и старшие классы, с прицелом на вуз — одно из самых классических. Еще лет 15 назад, когда этот старт совершала я, то было время и вовсе раннее: мало кто знакомился с азами программирования сверх школьной программы информатики. Да и в общественном представлении «хайпа», как это принято говорить сейчас, вокруг IT — с тем кадровым голодом и космическими зарплатами — не предвиделось. Никаких фронтенд-разработчиков[1], IT-дизайна, понятий UX/UI и бригад тестировщиков программного обеспечения не существовало в массовом сознании отрасли. PhotoShop не предугадывал появления толп веб-дизайнеров, ринувшихся отрисовывать макеты динамических страниц в продукте, предназначенном для обработки статичных растровых изображений. ВКонтакте еще не был запу-

[1] Здесь и далее вы можете найти пояснения к терминам, названиям технологий и инструментов в Глоссарии в конце книги.

щен, Falcon только пыталась выбраться из фантазий инженеров в бескрайние слои атмосферы, а устройство повседневного средства связи и аппарат для создания снимков, само собой, ничего не знали о возможном единении.

Максимум, что представлял себе среднестатистический подросток, не считая отдельных гиков и олимпиадников, о стоящих профессиональных инвестициях своего ума — польза освоения ПК, который стал постепенно завоевывать место в быту и бизнесе. При том что компьютерная грамотность населения оставалась на крайне низком уровне. А значит, был шанс востребованности — для обслуживания рабочих станций и манипуляций в пакетных программах на нужды фирм.

Доступ в Интернет осуществлялся по телефонным модемам и заветным карточкам на ограниченное время, а лабораторные работы те счастливчики, у которых были более-менее честные уроки информатики, приносили на дискетках. Переформировать свое сознание на higt-tech-восприятие и включиться в бесконечную гонку технологии тогда требовало некоторых усилий.

Сейчас всё проще. Даже если вы решите начать одновременно с поступлением, и, тем более, если уже начали, — вы успели впитать очень многое. Незаметно для себя. IT теперь вокруг нас, с детства, каждый день. Пусть прогресс нещадно меняет формы — вектор развития уже в вашем сознании. Вы знаете множество терминов. Ежедневно вы стоите по ту сторону барьера для разра-

ботчиков — на темной, пользовательской, половине. Вы видите удобные приложения и тормозящие интерфейсы, увлекающие дизайны сайтов и неработающие ссылки. С большой вероятностью — знаете разрешение и название ОС вашего телефона, хотя бы раз натыкались на страницу 404 и пользовались HTML5-плеером. А значит, вы уже чувствуете, что есть «хорошо», а что стоит сделать, чтобы получить разгневанного пользователя. Да, детали изменятся, но сам принцип мышления и нужный угол взгляда на мир — уже с вами. Сейчас вы достаточно сознательны, чтобы самостоятельно направлять обучение в близкое вам русло, и достаточно увлечены, чтобы создавать Фейсбук ночами в общежитии.

«А если я хочу начать *раньше*?»

Средний возраст прихода в фигурное катание будущих олимпийских чемпионов — порядка 4 лет. Старт в юные годы дает вам огромную фору. И это прекрасно. Но важно помнить, что IT и разработка продуктов — это не только скорость набора символов и крутость вашего приложения. Это также учет требований конечных пользователей, решение их проблем. Увы, в погоне за техничностью и модой на инструменты разработчики забывают, зачем они здесь собрались. А вот зачем: решать проблемы людей. Делать их жизнь проще. Позволять каждому тратить больше времени на то, что получается у него лучше, чем у любого другого.

Для эффективного достижения таких целей нужно глубокое понимание предметной области. Для этого, в свою очередь, необходимы зрелость и широта взглядов. Вы будете удивлены, как революционно могут помочь любые знания из совершенно неожиданной сферы. Будьте спортсменами и оттачивайте технику: кто посмеет сказать, что научиться как можно раньше — это плохо? Но не забывайте, как важно сохранить увлеченность работой на всю оставшуюся жизнь, а еще — не упустить за программированием с малых лет тысячи других областей и терабайты знаний об устройстве мира. Ведь именно они с большой вероятностью позволят вам создать новые решения, увидеть скрытое от «программистов-аутистов» и стать лучшим разработчиком, тимлидом, руководителем в Google или создателем стартапа, который изменит мир.

Илон Маск много читал в детстве, путешествовал и, что более важно, жил в среде расового противостояния и сохраненного рабства на землях ЮАР. С раннего возраста он не посвящал свою жизнь изучению программирования, хотя первая его игра увидела свет еще в довузовский период. Нестандартный жизненный путь и широкий взгляд на мир с иного угла дали результат, недостижимый одним лишь корпением над документацией всё свободное время. Ричард Фейнман с самого детства собирал радиоустановки на чердаке. Но в более поздние годы не сидел, запершись в кабинете физики, а осваивал игру на барабанах и ходил с парадами по Бразилии.

Помните, раньше — значит надежнее, но в долгосрочной перспективе не значит результативнее, разумнее, профессиональнее. Да и вообще, мало что значит. И последующие истории тому доказательство.

«Хочу поменять профессию. Что, если начать *позже*?»

Уже окончив вуз по иной специальности, проработав некоторое количество долгих лет вне IT… У вас есть тот самый кругозор, который создает уникальную комбинацию способностей и может привести к революционным решениям. И да, у вас гораздо больше воли, мотивации и целеустремленности — ибо такой выбор серьезнее, он требует кардинальных перемен в жизни — переезда, перестраивания семейных отношений. Да и делается куда более осознанно, нежели выбор школьного абитуриента.

История первая

Первый техлид, с которым мне довелось поработать, казалось, мог всё.

Спортивный и подтянутый, быстрый в каждом движении, Саша был не прочь подзадорить коллег отвлеченным разговором и собрать всех вокруг на чашку несущего передышку чая. Несмотря на разницу в возрасте с нами, новичками и стажерами, он не угнетал авторитетом повидавшего жизнь разработчика, а делал атмосферу легкой, мотивирующей и оперативной. Манера насмешливо

журить за упущения и незнания нисколько не сковывала начинающих интровертов. Самым психологически сложным оставалось обратить на себя внимание: призывно размахивая руками, вынудить его снять неизменно надетые наушники.

Когда приходилось настраивать рабочее окружение и сталкиваться с ошибками, в которых Stack Overflow оказывался бессилен, — достаточно было позвать Сашу. Любая проблема с npm-пакетами, версиями gem-ов, последствиями кривых рук в Linux, настройками серверов решалась за пару минут. Когда нам поступила задача быстро завести партнерский интернет-магазин на PHP, наиболее глубоко знакомым с этим непрофильным для компании языком — разработка у нас велась на Ruby — снова оказался он. Именно Саша занимался ведением основного продукта — высоконагруженного сервиса для известного медиахолдинга. Между делом он грамотно воспитывал стажеров, многие из которых вырастали в успешных разработчиков. В свободное время вел технический блог, находил любопытные решения, пробовал, настраивал новые инструменты. В рабочее — успешно рассчитывал мощности и конфигурации серверов для поддержания устойчивой архитектуры видеоплатформы. Стоит добавить ко всему легкость коммуникаций: редкая общительность для традиционного типажа того времени — замкнутых программистов — шутки над коллегами мимоходом, инициация обеденных групповых вылазок командой на турники с подтягиваниями, отжиманиями

и хардкордной «планкой». Последняя, к слову, была показана им как образец, и никем более на тот момент не осилилась к повторению.

Казалось, вот так должен выглядеть профессионал, наслоивший на хорошо усвоенные академические знания практический опыт нескольких лет работы. Да-да, даже не десяти, пятнадцати или двадцати — ему было всего тридцать с небольшим. Каково же было мое удивление, когда я узнала, что академического IT-шного образования в этой истории нет и вовсе! Более того, нет даже хоть сколько-нибудь смежного технического. А по своему диплому наш лучший техлид:

— Преподаватель. Специалист физической культуры и спорта, — размеренно произнес однажды голос из-под наушников в ответ на поднятую кем-то образовательную тему.

— Ни за что бы не подумала… — все, что растерянно я смогла выдавить из себя тогда.

«Как тебе это удалось?», «Давно ты начал?», «А как ты вообще оказался в IT?» — удивленные вопросы посыпались со всех сторон.

Те самые смежные софт-скиллс из другой профессии сыграли здесь свою весомую роль. Более осознанный выбор и интерес к области, увлеченность и пробы собственных проектов в нерабочее время позволили ему за несколько лет догнать ведущих специалистов. А сохранившееся отношение к IT как к зарожденному изначально хобби помогало лучше многих поддерживать

актуальный уровень знаний. Успешно выступать с полезными докладами на IT-встречах и бегать на лыжах, побеждать в волейбольных турнирах и быть в отличной физической и профессиональной форме.

История вторая

В компании остро стоял вопрос найма новых кадров.

Сложность поиска верстальщиков и JS-специалистов заключалась в отсутствии в явном виде таких специализаций. Именно в тот период мне как руководителю фронтенд-разработки довелось пребывать в непростой нанимающей роли. Приходилось или брать студентов со скамьи и обучать с нуля со всеми вытекающими, или фильтровать огромный поток уже успевших поработать в смежных сферах с надеждой найти нужный уровень и типаж. Большинство IT-выпускников, не блиставших алгоритмическими способностями, шли в студии сайтов или подобные некрупные фирмы, стремясь получить работу по специальности. И зачастую предпочитали верстку как «облегченную» версию профессиональной деятельности. Лучшие же кадры выбирали чистое и «профильное» программирование — уходили в бэкенд. В итоге квалификация клиентских разработчиков даже спустя год-другой работы оставляла желать лучшего. Сеньоров и умудренных на тот момент не искали. Перепробовав различные техники отбора, мы в итоге пришли к популярной в известных мировых фирмах методике, где кандидату даже чисто на верстку помимо прочих вопро-

сов предлагалось решить и запрограммировать некую алгоритмическую задачу. Да-да, были такие времена, когда верстальщик и JS-программист были отдельными сущностями.

Шли собеседования, ротировались кандидаты, разочарование росло. Никто не мог справиться с задачей на приемлемом уровне. Причем затруднения вызывало логическое мышление: способность рассуждать, строить жизнеспособную модель решения. Кандидаты, требовавшие неприличные для себя зарплаты, специалисты по искомым фреймворкам, — все терялись и оказывались неспособны отойти от стандартных приемов. Они старались применить «прецедентный подход» и, не находя готовых решений из своей практики, что зачастую оказывалось синонимом «не вспоминая нагугленного ранее», — становились бессильны. Единственным справившимся со всеми заданиями кандидатом оказался разработчик, переехавший из другого города. Немногословный, сдержанный молодой человек, чаще с сосредоточенным и немного напряженным, будто что-то обдумывающим, выражением. Последние года два он работал в веб-студии, а до этого честно строил карьеру агента по недвижимости — иными словами, являлся профессиональным дипломированным риелтором.

На данный момент, спустя несколько лет, он продолжает работать на всех наших основных проектах, участвует в создании Smart TV-приложений, видеоплеера и иных высоконагруженных сервисов.

История третья

В наших кругах телевизоры — не самая модная отрасль, потому, когда успешно поборовшим задание на разработку приложения Smart TV стал пусть и непрофильный кандидат, я была безгранично довольна. Те, кто застал времена зарождения части «smart» в составе TV в не такие уж далекие 2010-е, возможно, помнят настоящий коэффициент IQ тех девайсов. Под капотом «умные» приложения содержали в себе обычный веб-браузер — казалось бы, что может быть проще. Но нет: видимо, опасаясь восстания машин, производители того времени прилагали значительные усилия по созданию всевозможных препятствий к торжеству ТВ-интеллекта. Документации и гайды по разработке тщательно запрятывались от незнакомцев, индусская поддержка брала недельные паузы на медитации, а сборка приложения и его тестирование на реальном телевизоре представляли собой путь из 9 кругов ада. Сюрпризы оказывались в самых неожиданных местах. Фокусы не ставились на элементы, стандартный HTML-ный select отказывался назначать выбранный элемент через JS, свойства CSS выделывали трюки уровня старых IE, а встроенные медиаплееры с их непредсказуемыми обработками сиков и сдвигов live-трансляций позволяли разработчикам медленно, но неотвратимо постигать дзен. Особенно отличался Philips, не желавший перезапрашивать и отрисовывать даже несколько небольших тумбнейлов на странице за небольшой промежуток времени и благо-

получно «вешавшийся» на этом так, что не срабатывала даже перезагрузка.

SDK, ОС платформ и их версии, системы сборки и тестирования на ТВ постоянно менялись. В каждом аппруве в стор оказывались какие-нибудь новые удивительные требования. Появлялись то лишние десятки полей, то дополнительные чек-листы, которые надо было заполнить. А еще требовалось подготовить 2 презентации в PowerPoint с картинками каждого экрана, таблицами элементов на нем, их подробным описанием на уровне «что должна делать каждая кнопка и каждый компонент приложения». Прошедшее все муки разработки — порядка нескольких месяцев! — приложение попадало в руки прилежных корейских или индийских тестировщиков. Спустя пару недель оно неминуемо возвращалось с замечанием вроде «ваша текущая выбранная кнопка недостаточно заметно обозначает фокус на себе с расстояния нескольких метров». После чего под крупные слезы дизайнеров рамка становилась еще жирнее и краснее. Продукт заново отправлялся на очередной цикл тестирования. Кстати, хочется снять кепку перед специалистами тестирования Индии и Кореи — как они проверяли функционал приложения на русском языке во времена отсутствия Google Translate по фото — для меня до сих пор остается загадкой.

Чтобы сражаться и побеждать, в этой сфере нужен был человек, разбирающийся не только в наборе качественного кода в редакторе, но и в сетевой и аппарат-

ной частях устройств, необходимых для подключения/настройки ТВ под отладку. И он нашелся…

…среди выпускников кафедры машиностроения. До прихода к нам Игорь уже поработал несколько лет во фронтенде. Живость ума и самообучение позволили освоить верстку и JavaScript на профессиональном уровне, сохранив при этом совершенно не похожую на разработчика индивидуальность. Уникальный юмор, порой излишняя прямота, молодость и подчас пугающий менеджеров оптимизм выделяли его среди «типичных программистов». Широта принятия и познания в других слоях жизни отлично вписали его в коллектив, очертив ореолом легкости и позитивного мышления. А еще — нередко несли нам спасительную перезарядку.

Именно Игорь справился с гибридной, аппаратно-фронтендерской, спецификой Smart TV успешнее многих. Сначала выполнил тестовое задание по написанию несложного ТВ-приложения, с нуля разобравшись со всей платформенной инфраструктурой. Затем практически единолично разработал для нашего крупного заказчика проект с не самым тривиальным интерфейсом, платежными сервисами, VoD и live-видео.

И хотя позднее, перейдя фронтенд-разработчиком в одну из крупнейших российских IT-компаний — «Рамблер», — он зарекся никогда не иметь дело с «теликами», заслуженного места в историях успехов непрофильных кандидатов ему уже не изменить.

История четвертая

Смена работы выглядит волнительной вдвойне, если сопряжена с переездом в другой город и выходом в одну из самых желанных компаний нашего IT-рынка. На деле же все гораздо комфортнее. Крупные корпорации несут в себе отлаженный процесс. Меня ждали теплая встреча, экскурсия по атмосферному, живому и бесконечно уютному офису. Место с видом, заставляющим хотеть покорить мир: на Неву, Литейный, полный историй и духа побед-центр. И уже подготовленный с учетом пожеланий персональный ноутбук. А главное — уникальные люди, такие, каким оказался Рома — мой будущий напарник.

Молодой, со стильной прической и холеным внешним видом — немецкие корни удивительным образом переплелись с сибирским происхождением. Его выделяла открытость, а желание поделиться знаниями полезных утилит и важными тонкостями проекта захватило меня с головой на весь оставшийся день.

Здесь, в Яндексе, больше всего меня поразило умение разработчиков молниеносно ориентироваться в любом инструментарии. Быстро пробовать, настраивать — не важно, насколько полна документация. И это-то при всеобщей редкости подобных навыков для типичного фронтендера. Казалось бы, нужно сочетать в себе познания DevOps, где-то становиться немного админом и бэкенд-специалистом, чтобы умело манипулировать системами выкатки и развертыванием на серверах.

Среднестатистический, порой даже опытный, мидл-фронтенд-разработчик зачастую стремится избегать подобной деятельности, не чувствуя себя в ней как рыба в воде и, пожалуй, слегка побаиваясь.

Мой напарник писал скрипты склеивания файлов и настройки конфигов будто бы между делом. Всё это получалось так же легко, как центрирование какой-нибудь банальной фронтендерской кнопки, выполняемое не первый год. Самые современные полезные инструменты были подключены к проекту по инициативе одного только Ромы. Не успевали плавно распространиться по проекту одни из них, как в свободное время, пробуя, изучая, осознавая преимущества, он выбирал новый оптимальный для решения некой задачи вариант и встраивал его в код.

Лучше всех знакомый с проектом, Рома был назначен мне ментором. Как руководитель в прошлом, я сразу оценила верный педагогический подход: преподнесение информации, немаловажный начальный ввод в курс дела, пояснения по каждой из первых задач, экскурсии по кодовой базе и полезным внутренним инструментам. Присутствовали и необходимые рассказы на вечную тему: «Почему именно в этом месте “так исторически сложилось”». Хотя на тот момент я работала фронтенд-разработчиком дольше него и считала себя опытнее в плане обучения сотрудников — с легкой долей зависти про себя пришлось отметить профессионализм его преподавательских действий.

Поначалу мне виделось что-то непривычное в суждениях Ромы для порой типовых фронтендерских задач. Однако поток новой информации, обрушившийся на меня в тогда еще чуждой компании, и лояльность к окружающим не оставляли времени и весомых оснований для развития своих мыслей на этот счет.

Пока разработчики других отделов десятками решали, какие средства для тестирования выбрать, с чего начать, как вообще подойти к написанию кейсов, мой напарник успевал не только определиться, но и настроить, написать и вдобавок по собственной инициативе — интегрировать в процесс Continuous Integration. При следующей встрече не полениться набросать презентацию, демонстрирующую ключевые моменты его решения коллегам. Стоит ли упоминать, что у человека, который так горел глазами на работе и успевал так много, были и внеурочные проекты, где он не пытался зарабатывать старыми знаниями, а пробовал новинки IT-шной сферы, держал нос по ветру и стремился улучшить свои программистские подходы.

Когда в одной из задач потребовалось динамически менять заголовки страниц, он подсказал мне релевантную React-библиотеку.

— Посмотри Helmet, — нетрадиционно для привыкших к чатам разработчиков вслух произнес напарник. Максимальное время живого общения голосом было любопытной деталью нашего взаимодействия. Десятки фич и сотни строк рефакторинга доехали до пользовате-

лей в разы быстрее благодаря такому незамысловатому подходу.

— Как-как? — переспросила я, не зная подобного слова и, как следствие, не найдя аналогий с решаемой задачей.

— Как шлем по-английски, — доходчиво пояснил мой ментор.

Английским я занималась то параллельно, то последовательно всю сознательную взрослую жизнь: вместо, пыталась читать оригиналы, смотреть фильмы, использовать для общения в поездках за границей. Да, асом не была, но всё же. Та ситуация со шлемом меня задела. Неужели всё зря, и для всех, но только не меня, очевидно, как будет шлем по-английски?

Впрочем, по этой части чуть позже я позволила себе успокоиться — когда однажды за обедом я узнала от своего наставника, что он дипломированный преподаватель английского языка и даже проработал несколько лет в школах.

Его история еще одно яркое доказательство: университетская профессия и время старта не так важны, как желание, природная гибкость ума и огонь в глазах.

Не все истории смены своей текущей специализации на работу в IT успешны. Не все долговечны.

На моей памяти было резюме кандидата, где строки о работе в должности фронтенд-разработчика в равных пропорциях чередовались со строками вида «повар-тех-

нолог» и «главный суши-мастер». Подтверждая примеры выше, с тестовым заданием он справился вполне прилично — хотя и для уровня джуниора, а не искомого мидла. Мы решили не нанимать. Всё же в здании офиса недавно нашим же гендиректором был открыт стильный ресторан, куда мы все ежедневно ходили обедать. И очень уж не хотелось быстро потерять ценный найденный IT-шный кадр.

Минуточку, скажете вы, там, парой страниц выше, было нечто такое, что зацепило меня, резко выбилось из общего вполне себе политкорректного контекста. Что-то такое, не очень приятное на глаз и слух. Ах да, «программисты-аутисты»... Позвольте, звучит как оскорбление! К чему же это?

Компания, в которой я провела самые яркие карьерные годы, была выстроена не то чтобы с нуля, а скорее даже с минуса. Малоприбыльный регион — Орловская область, недостаток высокопрофессиональных кадров. Первый офис в подвале, где ведрами приходилось вычерпывать воду. А сейчас «Инвентос» — известные каждому клиенты, серьезные решения в своей нише, команда глубоких и мыслящих профессионалов и собственное здание компании.

Большинство выпускников-программистов начала 2000-х с радостью погружались в дебри отрешенного от мира кодирования только-только набирающей обороты отрасли. Но, к счастью, случались редкие и поворотные

исключения. Наш генеральный директор и основатель компании обладал решающим для IT тех лет талантом — умел объединять людей. Профильное образование, нежелание «кодить» и поразительное чутье в вопросах того, за что стоит браться и что выстрелит в перспективе, привели к созданию многих прорывных продуктов. Одним из них стал будущий Rutube, проданный впоследствии холдингу «Газпром-медиа».

Прямота, требовательность к сотрудникам и легкость прощания с теми, кто не болел душой «за дело», позволили построить не только успешный IT-бизнес, но и собственный ресторан, и даже радиостанцию.

Когда я впервые услышала то сравнение — «программисты-аутисты» — из уст «отца», моему возмущению не было предела. Разве можно относиться так к своим сотрудникам, тем, кто отдает большую часть своей ежедневной жизни на благо бизнеса? Масштаб эмоций усилился атмосферой: знаменитая фраза прозвучала на ежегодном выступлении в канун новогодних каникул перед всеми трудившимися в компании разработчиками. Только позже я смогла понять, насколько жизненно и справедливо то сравнение и насколько дальновидным было его озвучить.

Признание проблемы — половина решения. Услышав подобное, мне захотелось не быть тем ограниченным человеком, не умеющим общаться с внешним миром. Да, задачи требуют концентрации, но для великих продуктов всегда приходится выходить из зоны комфорта.

Выбираться за рамки, чтобы было куда расти. Достроить карту за дверцей дома и не упасть в пропасть. Важно мыслить нестандартно, комбинируя знания и опыт. А к человеку, открывшему после запуска IT-компании не менее успешный ресторан, свою радиостанцию, отстроившему современный офис и спроектировавшему научный городок, — стоит как минимум прислушаться.

«Так когда начать?» — спрашиваете вы. Будучи родителем и беспокоясь за будущее своих детей, думая, что сделанный ими или настоятельно рекомендованный вами выбор в пользу IT вечен, как набитая подростком татуировка, — вздохните и расслабьтесь. Надеюсь, приведенных примеров силы последующей и осознанной смены специальности достаточно. Если же вы школьник или представитель иной профессии, сомневающийся в сроках начала, то у меня для вас, как читателей этой книги, хорошие новости. Вы уже начали.

Советы в конце подраздела

- Определитесь как можно раньше.
- Срок вторичен, сбалансированное развитие первично. Ставьте на первое место кругозор, не жертвуйте всем ради одного.
- Не беспокойтесь о четкой точке старта. Постепенно окружайте себя связанными источниками. Уже сейчас.

С ЧЕГО НАЧАТЬ

Создать вокруг себя правильную атмосферу так важно и так непросто поначалу. Попасть в хорошие руки. Сформировать победное чутье.

Переделать — почти всегда значит устранить последствия, вернуться в начальную точку и сделать заново. Чем ближе к началу было совершено неверное ветвление, тем больше ресурсов потребует нормализация ситуации до желаемой. Только ли вы задумываетесь о поступлении на IT-специальность, ищите ли первый источник материалов о программировании, или же выбираете стек посовременнее для нового крупного проекта — отнеситесь к первым шагам втройне внимательнее. Но и не тяните. Если время идет, а видение не приходит — возможно, у вас просто недостаточно операционных данных. Сделайте любой интуитивно перспективный первый шаг и со временем дорога возникнет.

Вспоминая себя в школьные годы, могу сказать, что информатика и математика мне нравились всег-

да. Я выделяла их в числе любимчиков, однако никаких особых планов вокруг не строила. Школа значилась как лицейская, с веяниями современных подходов, и по ряду предметов полагались факультативы: некие кружки с углубленным изучением отдельных аспектов. Так, среди прочих дополнительных занятий по математике и английскому, я записалась на курсы программирования. То событие и послужило мощным толчком моего дальнейшего пути. Да, мы, программисты, любим быть впереди. А любые дополнительные курсы как нельзя лучше этому осознанию способствуют.

В самой программе тех школьных факультативов не содержалось ничего примечательного. По правде сказать, я не могу вспомнить никаких занятий, кроме двух: на одном мы программировали знаменитую черепашку ЛОГО, на других — осваивали Всемирную сеть и возможности коммуникаций в WWW. Но именно подобное чувство причастности к неизвестному многим, коллекционирование десятков мелких кусочков знаний подогревали интерес, заставляли больше смотреть, читать, впитывать и вне часов курсов. Каждый нюанс, каждый новый навык дает незаменимую фору, и кто знает, как она пригодится вам спустя многие годы. Пусть даже и в ничтожных бытовых мелочах. Почтовый сервис Mail.ru предлагает сейчас скидки от партнеров, процент по которым вычисляется по формуле, учитывающей стаж регистрации в почте. Максимальной скидкой ныне я обладаю исключительно благодаря тому, что на одном из тех школьных факуль-

тативов мы регистрировали почтовый ящик именно на этом самом, тогда относительно недавно появившемся сервисе.

Для всех, кто только мечтает кодить или уже работает и ищет точки роста, можно выделить основные формы деятельности и рекомендации, способствующие погружению в правильную и сильную атмосферу. Они могут сформировать надежную основу, если вы новичок, а если заказчики и релизы уже успели помотать ваши нервы — посмотрите на него как на чеклист профессионализма и дальнейшего развития. Виды занятий и принципы поведения во многом схожи на разных этапах пути, их же наполнение выбирайте исходя из своего текущего уровня.

Посещайте факультативы. Записывайтесь в онлайн-школы. Ходите на мастер-классы. Смотрите видеоуроки — самое популярное и легкодоступное образование для самых первых шагов. Действенная, с быстрой обратной связью методика наполнения себя новыми знаниями. Сами разработчики крупных фирм, опытные профессионалы ищут своим детям курсы, отдают их на занятия в IT-школы. Почему, если знают больше многих преподавателей? Один из моих коллег-тимлидов, водивший сына на занятия по Arduino и робототехнике, объяснял это так: «Нет времени придумывать, как увлечь». В школах же — готовые проработанные программы, отточенные на сотнях учеников. Специальные материалы,

красочная атрибутика, наборы тех же конструкторов и наглядные схемы, плюс дух соперничества с равными, который не воспроизвести, — ребенком ты подсознательно понимаешь: твой проигрыш отцу-профессионалу простителен. Да и преподавательский талант есть не у каждого разработчика.

С другой стороны, на волне популярности появляется масса некачественного контента, видеоуроков, онлайн-школ, находятся нечестные и непрофессиональные организаторы курсов офлайн. В Яндексе есть отличная возможность узнать у огромного числа коллег совет по волнующей тебя проблеме — не только рабочей. И одним из поднимаемых жизненных вопросов, которые мне доводилось застать, был поиск достойных подростковых курсов по программированию. Нет ничего удивительного в том, что и умудренные разработчики задаются подобными вопросами: важно развить и подогреть интерес школьника, без сухой теории и с грамотными лекторами. Даже если подросток уже сам пытается изучать и пробовать языки.

Читайте новости. Начните с лежащих на виду популярных источников, используемых ежедневно профессионалами с любым стажем. Пусть даже вы пока не можете понять почти ни слова. Зачем? Чтобы держать руку на пульсе, понимать, в какие области стоит углубиться именно вам. Вдохновляться разработками других, препарировать работу «черных ящиков»: отдельных методов, браузеров, баз данных и фреймворков. Читайте

иностранные, англоязычные сайты. Блоги известных разработчиков, авторов фреймворков, например, Дена Абрамова; авторов топовых IT-шных книг, евангелистов профильного мира и энтузиастов: Лию Веру, Сару Суадан и др., — причастных к разработке браузеров, таких как Вадим Макеев; создателей языков: Дугласа Крокфорда из мира JS, Страуструпа из века C++, легендарного Сэнди Метца и др.

Изучайте чужой код. GitHub, Bitbucket, примеры в статьях влиятельных авторов, выдержки из документаций. Как маленькие дети слагают свою речь, еще не понимая всей глубины отдельного слова и составляющих его букв, так и вам стоит принимать авторитетные фрагменты кода как слова, из которых сформируется вся программа. Пусть понимание отдельных нюансов придет чуть позже — окружайте себя готовыми классическими решениями и используйте их в своих работах. Старайтесь не просто копировать и вставлять, а самостоятельно набирать фрагмент — так стандартные решения быстрее отложатся в голове, подкрепившись моторной памятью. И в следующий раз, при написании своего кода с чистого листа, рука сама потянется добавить «content:»»» для псевдоэлемента или проверить на undefined переменную перед обращением к свойству.

Учите основы. Да, пусть это скучно, но необходимо. Без них вы никогда не станете уважаемым профессионалом. Представьте, к вам подходит подопечный джуниор-разработчик с вопросом: почему “2” + 2 = 22? А вы не

можете ответить, потому что не знаете про механизм приведения типов и наличие toString () у прототипа объекта Number. Сложно представить? Правильно, потому что у таких программистов подопечных не бывает. Более того, сейчас всё стремительнее сменяются фреймворки и дополняются возможности языка. Без понимания, что там «под капотом», практически невозможно быстро переключаться над высокоуровневыми реализациями. Угнаться за темпом IT не получится, запоминая только методы популярного на текущий момент фреймворка: завтра он сменится другим, ваши познания превратятся в тыкву, а цена на рынке как специалиста уйдет в –1. Все крупнейшие IT-компании уже давно просят кандидата на собеседовании написать алгоритмический код, зачастую без привязки к языку, а если и спрашивают конкретику, то исключительно про подводные камни, языковые тонкости и их комбинации, требующие глубокого понимания того, «как работает мотор».

Старайтесь понять. Думаю, предыдущего упоминания процессов отбора на работу IT-корпорациями достаточно, чтобы отказаться от сладостной мысли выучить/запомнить и попытать свое счастье в отрасли. Часы университетских лабораторных или стоящих курсов призваны заставить пробовать написать самому, наткнуться на проблемы, задуматься, почему так, понять и исправить. При необходимости — повторить. Но, увы, никак не «скопипастить» готовый код, как принято наи-

вно полагать у многих не слишком прилежных студентов. Любой успешный разработчик проводил порой десятки минут, раздумывая над логикой создателей языка, причинными связями, устройством браузера, взаимозависимостями нового материала и уже известных ему фрагментов знаний. Только понимание, как происходит рендеринг страницы, как влияют на него заданные вами CSS-свойства, помогут исправить «тормозящий» интерфейс. Без глубокого осознания нет качества, нет надежности, а часто и нет самого результата — столкнувшийся с неизвестным разработчик не в силах предпринять что-либо, ведь малейший шаг по диагонали делает его совершенно беспомощным в мире, законы которого он так и не осознал.

Впитывайте best practices. Как известно — devil in details. То, как написан код, насколько он читаем другими разработчиками и расширяем по мере жизни и развития продукта, — главные отличительные черты новичка от профессионала. Чтобы преодолеть этот путь быстрее, лучше взять велосипед, собранный по проверенным чертежам и отшлифованный до мельчайших деталей, нежели начать выяснять, овальное или круглое колесо лучше зарекомендует себя для комфортных путешествий. Знакомство с Бандой Четырех, принципами DRY, SOLID, классикой вроде «Рефакторинга» Мартина Фаулера, Стивом Макконнеллом с его «Совершенным кодом» и «Чистым кодом» Роберта Мартина — ваш необходимый чеклист по данному пункту.

Не замалчивайте возникшие вопросы. Непонятно — гуглите, ищите в Яндексе, добирайтесь до сути. Такие знания — базовый каркас, ваш алфавит, не освоите — никогда не сможете заговорить.

Смотрите конференции. Сейчас их число неизменно растет, а уровень спикеров из-за конкуренции в сфере проведения подобных мероприятий повышается. Highload, HolyJS, Dev Conf — нет смысла перечислять конкретные названия, они изменяются и дополняются каждый год. Да, большинство стоят немалых средств, но видео спустя несколько месяцев, как правило, выкладываются в открытый доступ. Для вас как для абсолютного новичка подобная задержка не играет ключевой роли, поскольку объем другого, еще неизвестного материала остается достаточно велик.

Интересуйтесь научно-популярными источниками, *изучайте прикладные отрасли*. Любые книги по созданию лекарств или поиску вирусов. Сфера автоматизации рутины уже пройдена, на первый план выходят биоинформатика, исследования на стыке IT, химии и физики — все, где фундаментальные науки и высокоинтеллектуальные сферы могут вобрать в себя информационный прогресс. Очень мало математиков, биологов и медиков, способных двигать развитие робототехники и исследований ИИ, мало и тех, кто обладает научными знаниями и кому по силам скрестить их с миром программирования. Найдите смежную область, станьте экспертом — еще есть шансы создания революционных

разработок, идей и алгоритмов, способных перевернуть мир.

Изучайте биографии, автобиографии и труды авторства великих людей как из мира IT, так и из иных знаковых исторических и научных сфер. Стив Джобс, Марк Цукерберг, Павел Дуров, Илон Маск, Айзек Азимов, Никола Тесла, Луи Пастер, Крис Хэдфилд, Ричард Фейнман, Стивен Хокинг. Ход мыслей, ситуационные решения, масштаб целей, организация их достижения — всё полезное на примере их жизней, что готовы разглядеть именно вы. Только не обольщайтесь — еще ни один, прочитавший биографию создателя Apple, не стал вторым. Таких, как Джобс, — единица. Но десятки и тысячи крутых профессионалов в сотнях значимых для каждого человека компаний ежедневно вносят решающие вклады в то, каким вы встретите новое утро. Глупо рассчитывать повторить личность. Еще одним Биллом Гейтсом не стать, но можно стать вторым и третьим крутым разработчиком в вашей области. И я очень надеюсь, что собранные здесь советы, подчас пусть и кажущиеся явными, на пути не единичного, но предстоящего большинству, — помогут вам куда более гарантированно сделать мир лучше.

Учитесь общаться. IT — коллективная работа. Крупнейшие компании индустрии при оценке своих сотрудников учитывают отзывы коллег. Яндекс, Google, Avito проводят регулярные ревью уровня разработчиков. Один из первых этапов процесса и конечная составляющая успеха — фидбеки от менеджера или product owner'а,

тестировщиков и коллег, с которыми вам хоть сколько-нибудь приходилось взаимодействовать. К сожалению или к счастью, психология межличностных отношений такова, что ваша уникальность и высокий профессионализм вряд ли помогут услышать восторженные отзывы коллег, если ежедневно вы, не проронив приветствия, гордо размещаетесь в своем рабочем углу, последовательно воздерживаясь от участия в общих дискуссиях, ходите обособленно обедать в другое кафе и не утруждаетесь поздравлением коллег со знаменательными событиями.

Будьте звездой и учитесь приглушать свет. Я видела много талантливых программистов. Победители олимпиад, создатели стартапов, покупаемых за миллионы долларов ключевыми игроками индустрии, технические лидеры огромных корпораций. В крупных компаниях с сотнями и тысячами сотрудников, не знакомых друг с другом, вы можете пообщаться с любым случайным человеком в компании. Подписываясь под этой активностью, раз в некий промежуток времени вы отправляетесь с коллегой из другой части компании выпить чашечку чая, стакан кофе или создать себе иную вольно-рабочую атмосферу, чтобы поговорить. Вашим оппонентом может оказаться другой разработчик, либо же сотрудник абсолютно отличной сферы, смотрящий на компанию с другого ракурса. Обмен опытом, расширение понимания, что происходит вокруг, да и, в конце концов, завязывание дружеских и не только отношений между сотрудниками — цели ясны.

На очередной из таких встреч мне довелось пообщаться с разработчиком одной из ключевых частей продукта — поисковой выдачи. Приятный внешне парень с располагающей улыбкой и добрым приветливым взглядом. Краткость и содержательность отличали его речь от многих других моих оппонентов, ничуть не вредя при этом теплоте и глубине беседы — с ним хотелось говорить. Он выглядел довольно скромным молодым сотрудником, недавно пришедшим в компанию. Легкое недоумение, «почему именно он» составляет ядро сложного продукта, быстро сменилось угрызениями совести за свою предвзятость. Так, в ходе разговора речь зашла о соревнованиях по программированию, в том числе чемпионатах мира. Случайно всплыло имя общего знакомого, и выяснилось, что мой собеседник — также участник и финалист международной олимпиады ICPC — читай: программист с логическими способностями и алгоритмическими знаниями выше среднего. И вне стен текущего места был частью создания сложнейших систем, обеспечивающих безопасность в нашей повседневной жизни. При этом располагающий и простой для коммуникаций человек. Именно такой — подкованный и сливающийся с командой — ценен каждой престижной IT-компании, стремящейся изменить мир и перестроить процессы будущего. В таких корпорациях зачастую мечтают работать профильные студенты и начинающие свой путь в отрасли. Позволить своим мечтам сбыться вы сможете, научившись быть органичной частью целого. Подчас

вам придется менять линию поведения на приносящую продукту максимальный успех в счет собственного квалифицированного эго. Знайте многое, показывайте по необходимости.

Продолжайте непрерывно. Выбрав IT, нужно отдавать себе отчет в стремительности изменений данной сферы. Даже простое поддержание уровня своего профессионализма потребует ежедневных затрат. Выходят новые версии языков и фреймворков, появляются более эффективные инструменты, библиотеки, редакторы кода, формируются полезные практики, проходят конференции, крупные компании делятся своими решениями. Все знакомые мне успешные разработчики регулярно тратят время на приведение себя в IT-форму. Полчаса на обеде, час вечером, полдня на выходных, видео между делом, обсуждение нового стандарта с друзьями-программистами, чтение списка новых фич фреймворка вечером — ваша будущая регулярная плата за членство в увлекательном мире IT.

Воспитывайте самоорганизованность. Вы должны уметь встроить в свой ежедневный рацион час чтения новостей, изучения новой технологии, выбирать верные объекты освоения вне основной работы. На рабочем месте, в свою очередь, уметь спланировать свой день и неделю так, чтобы все требуемые фичи были реализованы в срок. Например, я начинаю свой рабочий день с анализа панели задач таск-трекера. Затем просматриваю почту и мессенджеры. Это позволяет не пропустить

срочные вопросы, пришедшие поздно вечером, ночью или рано утром, и скорректировать план на день. Быстрое и важное сначала, крупное и плановое — «в рабочем порядке». Каждый срочный запрос предварительно переосмысливается и фильтруется — действительно ли здесь «горит» для пользователей и продукта, не имеет ли место наносной человеческий фактор? В точках взаимодействия с коллегами придерживаюсь принципа максимально оперативных «спрашивать и отвечать». Если вы точно знаете, что под конец разработки фичи вам понадобится дополнительное железо или дизайн новой иконки — запросите сразу. Распараллельтесь, узнайте о рисках раньше, чем возникнет неожиданная задержка от смены процесса выдачи квот или внезапная болезнь дизайнера. Со своей стороны, быстро выданная вами информация помогает не тормозить работу других команд и не навлечь справедливый негатив их ненароком сорванными сроками.

Современные методологии разработки, используемые в серьезных IT-компаниях, такие как Agile или Scrum, требуют адекватной оценки, планирования и обеспечения выполнения в оговоренный срок самим разработчиком. И если раньше менеджеры несли вину за срывы сроков, сейчас всё больше культура компаний ведет к тому, что ответственность за оценку времени и их соблюдение перекладывается на программиста. Если же вы, выбирая IT, грезите фрилансом и удаленной работой — самоорганизация понадобится вам вдвойне. Например, во

фрилансе вы сами себе и менеджер, и руководство — необходимо верно рассчитать свою нагрузку и сдавать вовремя без лишних мотиваторов. А трудясь удаленно, без рабочей атмосферы офиса и темпа, задаваемого вашими коллегами бок о бок, приходится противостоять соблазнам и раздражителям, переключающим ваше внимание.

Начните с малого — плана на день. Научитесь полностью выполнять его. Честно отвечайте себе, почему не смогли. Были недостаточно быстры? Оттачивайте мастерство, используйте вспомогательные инструменты. Запланировали слишком много? Отделите желаемое от действительного, учтите предыдущий опыт выполнения подобных дел для понимания требуемого времени. Нашлись более срочные дела? Оцените важность всех задач и выберите меньше, но из самых нужных.

Привыкайте *брать ответственность* и *проявлять инициативу*. Тренируйтесь на других сферах — подумайте, как оптимальнее пройти от дома до работы или университета, как передернуть привычный шаблон поведения в стандартной ситуации, как сократить время сбора утром. Попробуйте мыслить нестандартно, организовать расстановку книг в домашней библиотеке или провести небольшой митап в известной вам области. Зачем? Затем, что это именно те важные софт-скилс, которые все чаще желает видеть в своих сотрудниках индустрия ПО. Методологии построения рабочего процесса, равно как и критерии оценки сотрудников, — быть может, сами того не осознавая, — исходят из наличия у топовых со-

трудников способности предлагать улучшения, внедрять их в команде и… отвечать головой, если твой «улучшающий код» внезапно отправил сервера прилечь на пару часов на незаслуженный отдых. Подход управления разработкой Scrum поощряет участников команды, склонных брать на себя дополнительные роли, например, роль scrum-мастера. Подход с наличием начальника-надзирателя, увы, устарел. Сейчас до разработчика спускается задача, детали реализации которой, ожидается, предложит он сам. Более того, всё больше распространяется презентация результата заказчику напрямую самим разработчиком. А это значит, что вы должны уметь собрать информацию о проделанной работе со всей команды и красиво ее подать, общаться и выступать перед людьми, держаться уверенно, ориентироваться в моменте и немного быть политиком, отвечая на неудобные вопросы заказчика. Выступления на конференциях, написание статей, предложения по организации процессов учитываются наравне с ответственным и качественным программированием нужной фичи — прямой обязанностью разработчика. Проникнитесь, насколько нужные здесь качества далеки от стандартных базовых черт хорошего разработчика. По моему опыту, компании склонны нанимать «человека, умеющего говорить», чем подчас более глубокомысленного и вдумчивого специалиста с яркими чертами интроверта. Как по мне — уход не на ту тропинку с главной дороги, но реалии таковы, и вы должны быть к этому готовы.

Обозначьте *верные приоритеты*. Сейчас жизнь IT бурлит и кипит, отнимая у наиболее увлеченных кадров всё свободное время, занимая их молодые умы. Я видела многих, кто сгорал на работе. IT-успех — удел молодых, амбициозных и азартных. Регулярно сидеть, день и ночь дописывая захватывающую фичу, доделывая нестандартный проект, — сродни зависимости. Образ небритого и немытого, физически мало развитого молодого человека органично примеряется и на паренька-геймера, и на талантливого программиста, потому что грань отчасти стирается. IT может поглотить, увлечь и завладеть вашим разумом гораздо больше, чем на рабочие 8 часов в сутки.

Если в вашей системе ценностей важнее семья и дети, если вы хотите жарить котлеты, гармонично сочетать в своей жизни другие сферы и любимые хобби — пожалуйста, смиритесь, что вероятность сделать что-то революционное в сфере IT низка, ибо всегда найдется более поглощенный и азартный, на грани нормы и безумия. Пока вы будете сплавляться с друзьями на байдарке, он придумает, как оптимизировать архитектуру подключения серверов или как изменить алгоритм, чтобы качественнее обучить нейросеть. Посему сразу установите себе верхний «keep balance» — соотношение времени и важности между IT и семьей, спортом, увлечениями. Иначе, поставив всё на красное, вы рискуете остаться у разбитого корыта, если, что не редкость, наступит разочарование в работе программистом или ваша карьера не сложится профессионально.

Советы в конце подраздела

- Поймите, интересна ли вам эта сфера. Да. Раз вы читаете эту книгу.
- Убедитесь, что можете. Трезво оцените силы и способности.
- Подумайте, обладаете ли нужными софт-скилс и готовы ли развивать их в себе.
- Вникайте, переосмысливайте стандартные школьные/университетские материалы.
- Ходите на курсы. Пробуйте. Давайте второй шанс. Бросайте, если нет искры.

КАК ВЫБИРАТЬ ИНФОРМАЦИЮ И ОТСЕИВАТЬ НЕНУЖНОЕ

Меня всегда раздражали онлайн-уроки, выложенные неизвестно кем на YouTube, книги вида «Осваиваем PHP за неделю» и прочие сочные источники, своими заголовками больше напоминающие новостные статьи желтой прессы. Всегда возникал вопрос: зачем вы тратите время на многочасовые просмотры сомнительных видео и чтение подобных книг? Просто пойдите и сделайте. Смотреть 60 минут урок, где пишут 1 строчку кода — что может быть менее эффективным! Соберитесь, почитайте документацию, напрягите мозг. Вы же не развлекаться сюда пришли и смотреть видео с котиками. Займитесь делом. И если после этого заявления вы всё еще здесь — отлично. Сойдемся на том, что кому-то такой подход ближе и, смотря затянутые часовые ролики про вывод «Hello, world!» от эникейщика, вы вдохновляетесь и верите, что способны перерасти в высококлассного специалиста, — что ж,

«на войне все средства...». Но для серьезного подхода нужны качественные, профессиональные и глубокие источники.

Закладываем фундамент

Скучная теория? Увы, она важна. Форму над содержанием можно несколько скрасить, но суть останется неизменной — вы должны впитать, осознать, выстроить в своем сознании базис.

Если вы начинаете с нуля и самостоятельно, найдите фундаментальные источники:

- программы мировых университетов. Массачусетский технологический институт, Калифорнийский университет, Стенфордский, Кембриджский и Оксфордский, университеты, Национальный университет Сингапура, Австралийский национальный университет, Мельбурнский университет и другие признанные мировым сообществом учебные заведения. Посмотрите актуальный рейтинг IT-вузов и интересующих вас специальностей, зайдите на их сайты и сообщества, найдите план обучения, списки источников и рекомендуемые материалы;
- списки книг по базовым дисциплинам, признанных лучшими интернет-сообществом. По теории языков программирования, в т. ч. актуальные учебники для вузов; по алгоритмам — хотя бы знаменитый Кнут и его серия «Искусство программирования»; по ма-