ПУТЬ ИНЖЕНЕРА-ПРОГРАММИСТА:

развитие навыков для успешной карьеры

Фернандо
Доглио

БОМБОРА
ИЗДАТЕЛЬСТВО
Москва

УДК 004.4
ББК 32.973.26-018.2
Д59

Fernando Doglio

SKILLS OF A SUCCESSFUL SOFTWARE ENGINEER 1ST EDITION

Доглио, Фернандо.

Д59 Путь инженера-программиста : развитие навыков для успешной карьеры / Фернандо Доглио ; [перевод с английского К. А. Вантуха]. — Москва : Эксмо, 2025. — 240 с. — (Manning: профессиональные книги для ИТ-специалистов).

ISBN 978-5-04-201473-4

«Путь инженера-программиста» — это полезное руководство для всех, кто хочет построить успешную карьеру в сфере разработки программного обеспечения. Книга предлагает детальный план действий, начиная с получения базовых знаний и заканчивая достижением уровня профессионала, который способен формировать команды разработчиков и управлять проектами.

Здесь содержатся ключевые аспекты разработки: эффективное написание и оптимизация кода, использование лучших практик программирования, таких как рефакторинг и модульное тестирование, а также освоение навыков для успешного прохождения собеседований и продвижения по карьерной лестнице. Особое внимание уделяется коммуникации и командной работе.

УДК 004.4
ББК 32.973.26-018.2

ISBN 978-5-04-201473-4

Оглавление

Приветствие

Перед вами книга «Путь инженера программиста: развитие навыков для успешной карьеры», которая была написана разработчиком и для разработчиков.

По крайней мере, в одной из своих прошлых жизней, больше 20 лет тому назад, я был им. За эти годы я увидел, как наша отрасль превратилась в могучую силу, приводящую в движение весь мир. Программирование стало настолько необходимой профессией, что мы пока не готовы отказаться от нее. Потребность в программистах остается очень высокой.

И тем не менее по непонятной причине получить должность программиста становится с каждым днем сложнее и сложнее. Профессиональное сообщество активно затрудняет доступ для новичков (что безумно само по себе), а в требованиях компаний-работодателей фигурируют «единороги*» наподобие младших разработчиков с 10-летним стажем — и в силу этого начать карьеру в отрасли становится все сложнее.

Поэтому я решил написать книгу о том, как карьера в разработке программного обеспечения (ПО) выглядит на самом деле, о неписаных стандартах нашей профессии и кодексе поведения, которому вам, как вступающему в это всемирное сообществу новичку, предстоит следовать.

В то же время прошу заметить, что эта книга не научит вас писать код: предполагается, что это вы уже умеете — по крайней мере, на базовом уровне. Некоторые технические аспекты профессии в ней затрагиваются, главным образом в силу того, что они лежат в основе любого проекта создания ПО — а это, в свою очередь, повышает вашу квалификацию как разработчика. В основном же книга посвящена соображениям нетехнического характера, которые вам предстоит учитывать на пути от младшего разработчика до техлида. Я не стану компилировать информацию из интернета, а вместо этого буду пользоваться моим собственным опытом и, надеюсь, здравым суждением.

Опираясь на мои ошибки, мои удачи и ваши мысли по поводу прочитанного, вместе мы создадим ресурс, способный помочь другим в подобном

---

* «Единорогом» принято называть явление или предмет с несовместимыми свойствами. — *Прим. переводчика.*

положении. Поэтому, пожалуйста, заходите на форум liveBook (https://livebook.manning.com/book/skills-of-a-successful-software-engineer/discussion) и оставляйте комментарии. Пытаетесь ли вы получить свою первую должность в отрасли, или у вас за плечами уже есть несколько лет опыта, в любом случае финальная версия «Кодекса разработчика» прольет немного света на тот темный путь, который называется нашей карьерой.

Еще раз спасибо за интерес, проявленный к моей книге. Надеюсь, ее чтение доставит вам хотя бы половину того удовольствия, которое я получил от работы над ней.

Фернандо Доглио

1

Как стать успешным инженером по разработке ПО

В этой главе:

- Как избежать ошибочных представлений относительно начального набора навыков
- Как сосредоточиться на навыках, которые повышают вашу квалификацию разработчика

Со стороны разработка программного обеспечения выглядит весьма привлекательной индустрией: безработица отсутствует, зарплаты приличные, всегда есть куда расти, многие задачи предполагают командировки, имеется возможность работать со своего дивана на стартап в Кремниевой долине. Но раз так, почему же в разработке не трудятся все поголовно?

Правда заключается в том, что хотя отрасль может извне выглядеть интересной, попасть внутрь не так просто.

Я буквально знал, что хочу быть разработчиком, еще до того, как у меня появился первый компьютер. Свой выбор я сделал исходя из той стоимости, которую генерировали компьютеры во время моего детства. Но когда мне пришло время начать карьеру и окунуться в реальность, войти в профессию было не просто трудно: профессиональная среда была пугающей и негостеприимной.

У меня не было проводника, способного указать путь в лабиринте собеседований или хотя бы объявлений о приеме на работу. Каждые выходные я проводил по нескольку часов за чтением раздела объявлений местной газеты в поисках текстов о найме младших разработчиков без опыта.

Найти первую работу в качестве инженера-разработчика ПО может оказаться, мягко говоря, непростым делом. Большинство компаний, набирающих инженеров начального уровня, ожидают от них навыков работы

с какими-нибудь из самых свежих фреймворков и технологий или понимания достаточно сложных концепций, таких как шаблоны проектирования, наилучшие практики в области разработки и управление версиями. Покончив с этим списком, они переходят к расплывчатым требованиям типа хороших «межличностных навыков» или знания смежных областей ИТ.

Приведенный ниже список требований к кандидату составлен из множества объявлений в Интернете, где приглашаются на работу младшие разработчики:

- Диплом бакалавра в профильной области и минимум год опыта на аналогичной позиции.
- Знание методики безопасной разработки программного обеспечения.
- Знания среднего уровня в области проектирования, разработки, модификации и внедрения ПО, включая объектно-ориентированное программирования.
- Знание других областей ИТ.
- Демонстрация навыков работы с репозиториями ПО.
- Демонстрация навыков эффективной коммуникации и межличностного общения.
- Способность к автономной мотивации и эффективной самостоятельной работе.
- Демонстрация навыков решения нестандартных задач.
- Владение на среднем уровне C#, ASP.NET MVC, SQL Server, TypeScript и React.js.
- Опыт использования Git и GitHub.

Глядя на этот список, как может любой человек, желающий войти в отрасль, не почувствовать себя растерянным и не испугаться предложить свою кандидатуру? Всякий, читающий этот перечень, решит, что ему нужно минимум еще два года опыта, прежде чем его начнут воспринимать всерьез.

Я сам проходил через этот опыт 18 лет назад, и все еще помню, какие вопросы задавал себе тогда:

- Стоит ли мне вообще тратить силы на подачу резюме? Из 10 запрашиваемых навыков у меня в наличии всего 3.
- Стоит ли мне перестать изучать X и переключиться на Y, потому что на этой неделе всем потребовались разработчики на Y?
- Где взять опыт, если я ищу свою первую работу?

По-видимому, каждый разработчик в начале карьеры задает себе те же самые вопросы. Но принципиальный момент заключается вот в чем: они нормальные. Вы не приходите к неизбежному выводу, что не рождены быть программистом, а просто проходите через опыт младшего разработчика.

Именно поэтому я и пишу книгу: чтобы помочь вам построить путь к успешной карьере. Я проходил через все круги ада, выпадающие на долю

любого начинающего разработчика; я получил свою первую позицию, где моя зарплата была мизерной из-за отсутствия опыта. Мне случилось встретить хороших людей, научивших меня многому, касающемуся командной работы, и плохих, показавших мне на своем примере немало вещей, повторять которые не следует.

На протяжении книги я буду рассказывать о том или ином опыте, приобретенном в дороге. Я расскажу о наборе наилучших практик, которые вам предстоит применить к собственному коду. Я покажу вам некоторые основополагающие технические понятия, например, модульное тестирование и рефакторинг, чтобы вы понимали значение этих терминов и важность обозначаемых ими процессов в программном проекте. Расскажу, как находить баланс частной и профессиональной жизни так, чтобы не выгореть в процессе поиска, и дам несколько намеков, полезных на первом техническом интервью. В завершение поговорю о том, что значит быть разработчиком в большом коллективе и даже о том, что значит руководить коллективом, — чтобы вы лучше понимали, чего добивается от вас менеджер. Я надеюсь показать, что вы просто находитесь в начале пути, и что ваши сложности и сомнения совершенно нормальны и представляют собой часть опыта разработчика, — а вы согласились на его приобретение во всей полноте в тот момент, когда решили зарабатывать на жизнь умением писать код.

А что касается этой главы, то в ней я хотел поговорить об основополагающей вещи: что именно нужно, чтобы работать в качестве программиста? Дело в том, что в Интернете много информационного шума и, задав этот вопрос Google, можно получить много непохожих друг на друга статей. У каждого свое мнение, и многие имеют тенденцию концентрироваться на функциональных навыках, то есть на вещах, которым необходимо научиться, прежде чем вообще начинать задумываться о рассылке резюме.

Мой опыт говорит, что эти навыки не самая важная вещь — если понадобится, приобретете их в процессе работы. Самые важные качества разработчика (даже если он ни одного дня не работал по специальности) не относятся к разряду технических.

Давайте пройдемся по обоим спискам: общепринятых ошибочных представлений о том, что необходимо для работы программистом, и действительно полезных и нужных навыков.

1.1. Ненужное

Если вы пытаетесь выяснить в Интернете, как строить собственное обучение, чтобы предложить кандидатуру на первую свою должность в отрасли, или просто размышляете над планированием 5 лет после начала работы, то

позвольте мне поговорить о распространенных ошибочных представлениях, каковы требования для входа в профессию разработки ПО.

Заметьте, что наличие любого перечисленных ниже навыков не снизит ваши шансы, — но они не относятся к числу вещей абсолютно необходимых, чтобы начать работать. Воспринимайте навыки из перечня не как обязательные к исполнению задачи, а как навыки, которые со временем невредно было бы приобрести.

1.1.1. Диплом бакалавра по информатике или в аналогичной области

Или, по существу, от четырех лет формального образования. Я пытался его получить, приложил все усилия к тому, чтобы его закончить, но моя первая попытка закончилась неудачей. Я хотел быть инженером-разработчиком ПО, что в моей стране требует 5 лет карьеры, но на втором курсе у меня уже была первая работа, и формальное образование на том закончилось.

Сожалею ли я? Раньше сожалел, и именно поэтому впоследствии получил диплом в технической области (2 года учебы + дипломный проект), чтобы формализовать таким образом полученный к тому времени значительный опыт.

Нужны ли мне были первые 2 года университета? Да, но только потому, что они формализовали многие программные концепции, до которых я так или иначе дошел своим умом, обучаясь программированию для собственного удовольствия. Но в те времена не было экспресс-курсов, доступ в интернет ограничивался модемом со скоростью 2400 бод, а учебные ресурсы сводились к электронным журналам для хакеров (представлявших собой текстовые файлы, разбавленные некоторым количеством псевдографики).

Сейчас ситуация совсем иная, и к услугам всякого желающего научиться кодировать — почти неограниченное количество ресурсов. Это не преувеличение; существуют бесплатные онлайн-источники, например, YouTube или freeCodeCamp.org, где есть все, чтобы пройти путь с нуля ко вполне пригодному для реальной работы состоянию. Разумеется, справедлива истина, что не всякий может научиться, просто впитывая информацию, но существуют и другие варианты, платные курсы в местах типа Udemy или Skillshare, и там, если вам по силам вложить сумму, меньшую, чем плата за университетский диплом, вы получите доступ к онлайн-аудиториям, сессиям вопросов и ответов и возможность контактировать с другими студентами, которые решают те же задачи, что и вы.

Там можно приобрести практические знания, необходимые для того, чтобы приступить к написанию кода, и хотя для многих имеет ценность

упоминание диплома колледжа в резюме, но многие обращают все меньше внимания на этот пункт. Само собой, что это утверждение более верно для одних стран, чем для других, но так же верно, что наша профессия в высшей степени интернациональна: мы можем работать на компанию, находящуюся в любой точке глобуса, и как только мы начинаем это делать, университетский диплом начинает значить все меньше и меньше с течением времени.

1.1.2. Знание жизненного цикла разработки программного обеспечения

Типичный проект разработки ПО должен пройти через следующие стадии:

1. Сначала нужно провести анализ требований, чтобы в точности понимать, что нужно получить на выходе.
2. Затем перейти к планированию проекта, чтобы понимать, когда делать все эти вещи, и сколько времени они потребуют.
3. Третьим шагом будет разработка архитектуры. Как только известны ответы на вопросы «что» и «когда», пора начинать думать об ответе на «как», и в основе своей он будет содержаться в архитектуре.
4. И вот только тогда можно приступать к написанию кода. Большинство разработчиков концентрируется на этой стадии, но, как видите, работа начинается не с нее.
5. Вслед за этим происходит тестирование кода и продукта целиком. Убедиться, что на предыдущей стадии мы получили правильные результаты, абсолютно необходимо перед переходом на следующий шаг.
6. Внедрение продукта в эксплуатацию. Иными словами, передача его пользователям, после чего они уже могут тестировать продукт сами и сообщать вам результаты.

Поскольку процесс цикличен, обычно мы, получив результаты тестирования продукта пользователями, возвращаемся к началу, но я думаю, что идею вы поняли.

Само собой, этот цикл будет отражаться в ваших рабочих задачах, и применять знание его вам придется ежедневно. В то же время выставлять его знание в качестве требования к младшему разработчику — это все равно что требовать от молодого хирурга в его первый рабочий день быть главным на операции, минуя стадию ассистента.

Со временем он будет способен это делать, он, вероятно, читал об этом, — но вряд ли вы захотите поручить ему оперировать самостоятельно в первый же день.

У разработчика все то же самое: не следует рассчитывать, что вы с первого дня будете понимать содержание всех стадий цикла разработки: все равно вы