

# make

BOOTSTRAPPER'S HANDBOOK

building startups the indie way



Product Hunt's 2x Maker of the Year + founder of Nomad List +

Remote OK + Hoodmaps



Pieter Levels



# Contents

## Foreword

- 0.1. Thank you
- 0.2. Why did I write a book?
- 0.3. But I do want to write a book
- 0.4. The indie way of building a startup
- 0.5. I want to make bootstrapping great (again) #MBGA
- 0.6. Why should you listen to me?
- 0.7. This book is my entire brain dump
- 0.8. This book is continuously updated
- 0.9. App, site, product, startup, business
- 0.10. You'll need persistence, and luck
- 0.11. Practice

## This book

### Idea

- 2.1. Introduction
- 2.2. Solve your own problems
- 2.3. Your problem might just be everyone else's
- 2.4. You are the greatest expert at your own problems
- 2.5. How well do you need to know a problem to solve it?
- 2.6. Be more original, and your ideas will be original
- 2.7. The downside of solving only your own problems
- 2.8. Always start from the problem, not the solution
- 2.9. To get big, you have to start small
- 2.10. Start with a micro niche
- 2.11. From micro niche to multi-niche
- 2.12. From multi-niche to vertical integration
- 2.13. From vertical integration to becoming a platform
- 2.14. You just became big by starting small!
- 2.15. Your idea does not have to be earth-shattering
- 2.16. Create a list of ideas and keep track of them
- 2.17. Should you make ideas alone or in a group?
- 2.18. Don't be afraid to share your ideas

## **2.19. Conclusion**

2.19.1. Resources mentioned

2.19.2. Your homework

## **Build**

### **3.1. Introduction**

### **3.2. Build fast and minimal**

3.2.1. Downsides of fast development

3.2.2. Your enemy is perfection

3.2.3. How viable should a minimum viable product (MVP) be?

### **3.3. Build yourself or outsource**

3.3.1. DIY vs. people who hire others to do it for them

3.3.2. DIY vs. big teams with VC money

3.3.3. My gripe with venture capital in terms of building products

3.3.4. Bootstrapping vs. venture capital

3.3.4.a. The importance of keeping costs down

### **3.4. To build, should you learn to code?**

### **3.5. Tools**

3.5.1. Which tools should you use to build?

3.5.2. My "light" stack

3.5.3. Why are people so obsessed with tools?

3.5.4. How do you evaluate if a new tool is useful for you?

### **3.6. Native vs. web**

3.6.1. Web

3.6.2. Native

3.6.3. User experience and development

3.6.4. Updating native apps vs. web apps

3.6.5. Hybrid web + native apps

3.6.6. What are people using most?

3.6.7. A future where your web app lives inside other native apps

3.6.8. The web and native will merge in the future

3.6.9. Learn both

3.6.10. What are you able to build now?

### **3.7. Building with constraints**

- 3.7.1. No money/investment
- 3.7.2. No office
- 3.7.3. No coding skills
- 3.7.4. No connections

### **3.8. Building a startup without coding**

- 3.8.1. Building a landing page
- 3.8.2. Accepting user data entry
- 3.8.3. Processing & manipulating user data
- 3.8.4. Contacting users
- 3.8.5. Making tasks for contractors
- 3.8.6. Charging users payments
- 3.8.7. Okay, let's try build something

### **3.9. Let's talk APIs**

- 3.9.1. Why are they useful?
- 3.9.2. You can build a business on other people's API's

### **3.10. Conclusion**

- 3.10.1. Resources mentioned
- 3.10.2. Your homework

## **Launch**

### **4.1. Introduction**

### **4.2. What is a launch?**

### **4.3. Why even launch?**

### **4.4. How fast should you launch?**

### **4.5. Preparing your app for a launch**

- 4.5.1. Fix most bugs
- 4.5.2. Add an email box
- 4.5.3. Add push notifications
- 4.5.4. Set up analytics
- 4.5.5. Feedback box

### **4.6. Where to launch?**

- 4.6.1. Typical places to launch a startup

- 4.6.1.a. Launching on Product Hunt
- 4.6.1.b. Launching on Hacker News
- 4.6.1.c. Launching on Reddit
- 4.6.1.d. Launching on Beta List
- 4.6.1.e. There will be hate
- 4.6.2. Things not to do
  - 4.6.2.a. Asking people to share/like/post your product
  - 4.6.2.b. Buying fake upvotes, likes, followers
  - 4.6.2.c. So why doesn't this work
  - 4.6.2.d. Be organic
- 4.6.3. Telling your story
  - 4.6.3.a. Blog
- 4.6.4. Press
  - 4.6.4.a. Why press matters
  - 4.6.4.b. Why press increasingly matters less
  - 4.6.4.c. How to get press
  - 4.6.4.d. Which press outlets
- 4.6.5. Don't stick to one launch, keep launching
  - 4.6.5.a. Make every feature a launch opportunity
  - 4.6.5.b. Side project marketing
- 4.7. How to stay motivated working on one product**
  - 4.7.1. If it doesn't motivate you, sell it or kill it
  - 4.7.2. How many ideas should I work on at a time?
- 4.8. Conclusion**
  - 4.8.1. Resources mentioned
  - 4.8.2. Your homework

## **Grow**

- 5.1. Introduction**
- 5.2. Why is organic growth better?**
  - 5.2.1. Especially...fake users
- 5.3. How to get organic growth**
  - 5.3.1. Get new users

- 5.3.1.a. Keep launching
- 5.3.1.b. Spinning off
- 5.3.1.c. Tell stories to people & press
- 5.3.1.d. Build in public
- 5.3.1.e. Make people share easily
- 5.3.1.f. Human-readable URLs and slugs

## **5.4. Launch an API**

## **5.5. Build with your users**

## **5.6. Measure how you stand up against your competitors**

## **5.7. Conclusion**

- 5.7.1. Resources mentioned
- 5.7.2. Your homework

## **Monetize**

### **6.1. Introduction**

### **6.2. Why is monetization so important?**

### **6.3. Don't be afraid to charge money**

### **6.4. Charge money, get hate**

### **6.5. Build with monetization in mind**

### **6.6. Monetization is validation**

### **6.7. Business models**

- 6.7.1. Limit features to paid users
- 6.7.2. Pay-per-feature
- 6.7.3. Ads
- 6.7.4. Sponsorships
- 6.7.5. Patronage
- 6.7.6. Subscription-based memberships
- 6.7.7. Community model
  - 6.7.7.a. The story of Nomad List's paid membership community
- 6.7.8. Job boards
- 6.7.9. Conditional payments
- 6.7.10. Productizing an agency into a SaaS
- 6.7.11. Learn from your competitors' business models

6.7.12. Keep experimenting with business models

6.7.13. When are you done monetizing?

6.7.13.a. It depends on your objectives

6.7.13.b. How big do you want to be?

6.7.13.c. Widen your market

6.7.13.d. Grow the pie

## **6.8. Payment platforms**

6.8.1. Stripe

6.8.2. Braintree

6.8.3. PayPal

6.8.4. Local alternatives

6.8.5. Stripe Atlas

6.8.6. Use a combination of platforms

## **6.9. Use a Typeform to charge fast**

## **6.10. How to deal with refunds?**

## **6.11. How to deal with bookkeeping and tax?**

## **6.12. Conclusion**

6.12.1. Resources mentioned

6.12.2. Your homework

## **Automate**

### **7.1. Introduction**

### **7.2. How is automation relevant?**

### **7.3. What's a robot really?**

### **7.4. Don't automate if it's not worth to automate it**

### **7.5. Where do humans fit in here?**

7.5.1. What if the robots can't fix things themselves. How do the robots ask the humans for help?

### **7.6. So this is like passive income, right? No!**

### **7.7. The "bus test"**

### **7.8. Conclusion**

7.8.1. Resources mentioned

7.8.2. Your homework





**TL;DR**

# Introduction

Because you have little time, here's the mega short TL;DR (too long; didn't read) summary of this book.

-  **Idea**

Get an **idea** from problems in your own life. If you don't have problems that are original enough, become a more original person. Don't build products that are solutions in search of a problem.

-  **Build**

**Build** your idea with the tools you already know. Don't spend a year learning some language you'll never use. Don't outsource building to other people, that's a competitive disadvantage. Build only the core functionality. The rest comes later.

-  **Launch**

**Launch** early and multiple times. Launch to famous startups websites (like Product Hunt, Hacker News, The Next Web), mainstream websites (like Reddit) and mainstream press (like Forbes). But also remember to find where your specific audience hangs out on the internet and launch there. Launch in a friendly way, that means "here's something I made that might be useful for you", instead of acting like you're some big giant new startup coming to change the world.

-  **Grow**

**Grow** organically. A great product that people really need which is better than the rest will pull people in. You don't need ads for that. Don't hire

people if there's no revenue yet. Don't hire many people if there's revenue either. Stay lean and fast. Do things yourself.

-  **Monetize**

**Monetize** by asking users for money. Don't sell their data. Don't put ads everywhere. Don't dilute your product. Be honest that you need money to build the product they love and they'll be fine paying for it.

-  **Automate**

**Automate** by writing programs that do stuff that you do repeatedly. Only automate if it's worth the time saved. For stuff that's too hard to automate or not worth it, hire contractors. Let them work as autonomously as possible. Where possible let robots manage them (for example by giving them alerts when things happen in your product).

-  **Ethics**

Not a chapter but important: **be ethical**, and don't cut corners on ethics. You'll be rewarded by not doing dodgy stuff like spamming, manipulating your users into doing stuff, growth hacking your search rankings or faking your social media, or abusing your power to compete unfairly if you're successful. If you make a good product, you don't need any of this. If you make mistakes, own up to them and say sorry. Be nice as a person and especially as a company. Karma always pays back in the end. Just being ethical and nice is a competitive advantage these days because most companies (and people) are not!

-  **Homework**

**Homework:** Each chapter ends with homework exercises that you can do. Instead of just reading, I'd like you to use this book as a handbook while actually building and shipping a product. It doesn't matter if it fails. But

you need to do something instead of just read! This is not startup porn!  
This is startup life.



# Foreword

Even with all these things stacked against writing a book.

I want to write a book, if only because I see so much bullshit going around in the world of startups and tech. The media is presenting startups in the wrong way. People think they need to build billion dollar companies. They need to fly to San Francisco and build a "network" and get \$10 million dollar investment from old rich guys. They need to hire 10x power developers and work them for 100-hour work weeks while feeding them pizza and soda. It will be great, they said.

But it won't. It'll probably suck. And you probably won't get rich. Because the odds of a venture capital (VC) funded startup are *by definition* stacked against you. Only 10% or less exit and that doesn't even tell you if the founders make good money. There's giant company exits where the founders barely made money. That's why I'm writing this. To show you, you might be able to do it differently.

## **The indie way of building a startup**

What's the alternative? How about this: do things yourself and build a nice side project, that then can maybe turn into a bigger project, that then maybe becomes a company that makes you enough money to quit your day job and stop working for the man. Enough money to build up good savings, that if you invest well, will give you a nice early retirement. Your own company that can give you a little more freedom in your daily life, so you can spend it with friends, your family, your pets or just doing the things you love. Which in the best case is actually building an app, project, startup or company you love to work at/on/for!

**I want to make bootstrapping great (again)**

**#MBGA**

This way of building a company is called "bootstrapped". Which means you're self-funded. You use the resources you have to get started. The odds of building a successful bootstrapped business are way higher than building a venture funded billion dollar company. Because the goal of a bootstrapped business is much more reachable. You don't need to do a billion dollars in revenue. You're already there if you can pay yourself enough to live from. Any money that comes extra is even better! You'll probably have less stress, be happier and be therefore a better friend, lover, partner or parent. Just....relax.

The coolest thing about bootstrapping it is that it doesn't exclude "going big" later. Venture capital investors LOVE to invest in companies that already have proven revenue. And that's literally what a bootstrapped business is. You'll be miles further than the person next to you pitching with just a PowerPoint deck. When you go for millions of dollars of VC investment on day one, it means you do exclude building a healthy simple business. Your company is now strapped to a rocket and you need to go big or explode. That's why I think bootstrapping is the better way to build a business now.

I really, truly, honestly want to see the mainstream startup narrative change into one where bootstrapping, revenue and actual profit is "cool". Writing a book on it with a proven framework people can apply, may help accelerate this change.

There's a personal legacy aspect here: if I can have a small influence in changing this, it feels good as a person. It's nice to change things for the better. And if it doesn't, well, thousands of people paid me money for this book, so it's a nice backup for me in case I go bankrupt.

## **Why should you listen to me?**

Because I went from really scrappy side project to profitable company with users a few times now. Most times it failed miserably, but a few times it worked out for me.

At time of writing, my website **Remote OK** just became the most visited remote jobs board in the world with 1 million monthly visits. **Nomad List** is near that amount too, and ushered in a new era of digital nomads and remote work from **2014 onward**. They're both manually built by me, profitable with high margins (up to 90%), and highly automated. I was **Product Hunt's Maker of the Year twice**. I've launched my startups to Reddit's **frontpage twice**. I grew my projects together into **\$50,000 monthly revenue** while **blogging** and **tweeting** about all the personal ups and (lots of) downs for the past years. Most of my other projects failed (some miserably), but I was able to get an idea to success a few times.

There is a good chance that there is strong **survivorship bias** at work here though. Remember that.

## **This book is my entire brain dump**

I've been getting thousands of questions the last few years. I think if I started answering them I'd simply not get to working on my own projects anymore. That's why this book is the easiest knowledge transfer from me to you.

Literally every single thing I learned in the last few years building bootstrapped startups is in this book. It's my entire brain dumped on paper. It can be messy but it's everything I know. I hope it'll be something like giving back to the community and people will use it as guide in becoming indie makers and ship products. I've seen the drafts of this book already applied in hundreds of launched startups (because people will usually send me a message), which is super awesome. I'd love to see more. Having some positive influence on people's lives is a lot more interesting to me than more revenue, at this point.

## **This book is continuously updated**

I'll be working on this book just like any of my startups. It's a continuous project. I'll keep updating when I learn new things.



# App, site, product, startup, business

You'll see these terms in the book used somewhat like synonyms. Because most of the theory in this book applies to all of these. Sites these days are like web apps, and apps are more like sites, together they are products and a few of these products make up a startup which in turn is a business. Generally, they're all the same thing.

## You'll need persistence, and luck

You may need to try shipping 10 to 30 products for 1 to 3 years before you have anything that works. That's how this approach works. You build stuff and see what sticks. I don't know anybody who shipped one product and instantly became successful. It takes a long time to "get" it and even then it's a lot of luck and timing. If something doesn't seem to take off early on, it probably won't take off later, so make something new and try again.

As I, and this book practice radical honesty, there's a chance nothing you make will be successful. But by doing you'll have figured something new out, that might lead you to somewhere else, that will make you successful. Startups, and life, are about constantly pivoting when things don't work out. If you don't take action though, you can be sure nothing will ever happen. Stagnancy kills. So ship.



Always keep shipping.

## Practice

I want you to learn from actually shipping a product. This book is just ideas that might be wrong or right, and biased, but your own personal practical experience will be the thing (if anything) making you successful. Not this book! This book is just me pushing you to go sit on the bicycle. Now learn to ride it yourself. Practice is everything. Get your own style. And most importantly, ship.



You shouldn't be scared of sharing your idea because execution gives the idea its details and specifics. That means 10 people with the same idea will execute it in 10 completely different ways.

The benefit of being able to share your ideas is that you'll be discussing them with everyone. Potential customers, vendors, suppliers, whoever. Everyone will have some input on it which you may or may not use as feedback. Again, the market remains the most important feedback though.

Not sharing your idea is stupid because it'll stay only in your head. You for sure won't be objective at judging it since you have something called "optimism bias" which is "the tendency of individuals to underestimate the likelihood they will experience adverse events", e.g. you think it'll definitely be very successful.

It doesn't matter if people say "that idea will never work" because they're not the validation. The user paying/using it is the validation, not other people judging your idea! The point of sharing your idea is thus not to get people loving it or hating it. The point is that you get your brain working outside of its comfort zone (of talking to itself) and you'll evolve your idea. You'll come up with adaptations of your idea, or entirely new ideas by talking about it.

## **Conclusion**

To get ideas, try to find problems in your daily life. You're the foremost expert at problems you have, more than anyone else who doesn't have them. If you keep coming up with the same ideas as everyone else: try to make yourself a more original person by actively experiencing different things. Don't shy away from taboos and fringe ideas, that just mean you're ahead of the curve, they might become the next big thing. Don't think big, start thinking small first, then take it one step at a time, you'll become big by starting small. To avoid group-think and drama: work alone, especially early on. Share your ideas freely to get other people's input on them. Log every idea you have, filter them, and see which ones you can execute upon.

## **Resources mentioned**

- **Workflowy**
- **Trello**
- **Stripe**
- **Zapier**
- **Intercom**
- **Olark**
- **Typeform**

## **Your homework**

- Spend the next 7 days making a list of problems you have in your daily life, they can be small or big, try to find 3 ideas per day, so that you have at least 21 at the end of this week.



**Build**

# Introduction

**You made it to the second chapter. We'll now discuss how to build your idea.**

In the previous chapter we looked at how to find ideas for your product. To avoid getting stuck in an infinite loop of brainstorm and bullshit, you want to start building as soon as possible. The faster you get it out, the faster you'll see if people want to use it, how they use it and what other features they want to see you build. Without shipping, it's difficult to get any idea of what they want. There are obvious exceptions here: Apple hardly uses any focus groups or user testing, they choose to ship the very best (according to them). That's fine, but you're not Apple (yet).

## Build fast and minimal

While a decade ago development time was long and took lots of planning to get right, today we can go from idea to basic product in a matter of days.

We have cheap and easy to set up servers (virtual private servers or VPS, like **Linode** and **Digital Ocean**), and platforms as a service (PaaS, like **Heroku**) that take out all the difficulty of setting up your own server. Server operating systems like **Ubuntu** have become relatively simple to set up with tutorials readily being available online. The web server software itself has become fast and simple too: **NGINX** allows us to set up a basic default server that doesn't need much further configuration. Languages like Ruby, JavaScript, PHP and their frameworks like Ruby on Rails, Meteor, Laravel make it easy to build products by skipping the ground work.

What I mean sounds crazy but it's where I see this going. People will use your web app INSIDE these apps. What happens if you send a link to somebody on WhatsApp? They see a URL, they tap it, it opens INSIDE WhatsApp's web browser. It's a fully functional browser and your web app can run inside it. A web container inside some big company's native app is the real future platform for your web apps I think. Design for that use case. It's intrinsically viral as people will send your website's URL around and open it from there. They won't have to install anything.

## **The web and native will merge in the future**

I believe that web apps and mobile apps are converging. You see web apps get more and more powerful and getting access to more functionality of the device that was usually limited to mobile apps (for example the iPhone's gyroscope is now accessible through web). Other examples are how recently (2016), iPhone's browser doesn't show entire URLs anymore, like "http://nomadlist.com/amsterdam-netherlands", but instead shows the name of the site in the URL bar "nomadlist.com". That's obviously pointing towards seeing the web more as apps. In the long term, there are a few obstacles to overcome for the two to converge. Honestly though, it's impossible to predict. Five years ago, I would have thought by now everything would be, but we still have users both on the web and inside native apps.

## **Learn both**

If you have spare time, learn to do both web and native app development. It's a remarkable (and sought after) skill to be able to build both web apps and native apps. And of course, hybrid ones!

## **What are you able to build now?**

This book is about getting something out as soon as possible. Therefore, you should choose the platform where you already have some skills. This way you can get something out as fast as possible. For most people, and definitely for me, that'll always be web. Programming (and shipping) native apps to mobile



devices and getting them into app stores is more work than just building a website, uploading it to a server and hooking it up to a domain name. Do what's fastest for you.

Other people will say I'm wrong there, and you should base it on how users will use your app. That's a good point. But I don't want to see you get stuck learning Objective-C and Swift to get an iPhone app out, or paying a mobile developer \$25,000 to build it for you.

**Remember: you can ALWAYS go native later. If people use your site already, you can bug them to install your native app later. This is annoying but it's possible. No platform choice has to be permanent.**

## **Building with constraints**

I'll now discuss how to build stuff if you're constrained. Having constraints seem like a disadvantage but you can turn them in an advantage. So many people look at the negatives in their current position, but in fact, most of these can be considered a positive advantage.

### **No money/investment**

As mentioned previously, bootstrapping has become an advantage. You keep your costs low naturally. You get 0 of the bullshit attached with VC money. You maintain full ownership over your product and its roadmap (where you want to go with it). You maintain full ownership over the equity so that when you sell it later, you'll get lots of money (instead of just a diluted 5% because VCs took the rest).

And in our times you don't really need a lot of money to build something yourself. A simple web server is \$5/m. A code editor like Sublime Text is \$60. To make iOS apps, XCode is free. A developer license with Apple is \$100/year. It's not a whole lot compared to the money you can make back from it if you get people to use your app and pay for it.

Bootstrapping has become a very viable option for most software-based start-up ideas.

## **No office**

I obviously had to put this in here as I'm a big supporter of remote work. Not having to spend money on office rent is now an advantage. If you (and the people you work with) can work from anywhere, it also means you now have access to a worldwide pool of talent. Starting fully remote is much easier than switching to remote after you already have an entire set of office buildings and workers.

## **No coding skills**

Not being able to code means you can use off-the-shelf tools to quickly prototype stuff without losing yourself in giant codebases. As you'll see later in this chapter, you can build an entire landing page, get data from users, process that data, charge money without writing code. This means that you can spin up lots of different MVPs and see which one sticks, without much effort. Although you're more limited in *what* you can make, you'll be shipping faster than people coding stuff for months.

Once one takes off, you can always learn to code on-the-fly, or get someone to help you build stuff out.

## **No connections**

You're not a famous startup person with lots of connections in "the scene". Well, guess what. Those connections are mostly bullshit any way. And if you're "outside" the scene and not famous, it means you can act like an underdog. You'll be more indie than most and, guess what, people LOVE to support independent underdogs. You're fighting against big companies and people want to see you win, that is, until you become a big company yourself (the cycle of life). The "connections" people have after they get successful also put them in

a monoculture bubble, which you're not in (yet). You think more freely and that's better for creativity.

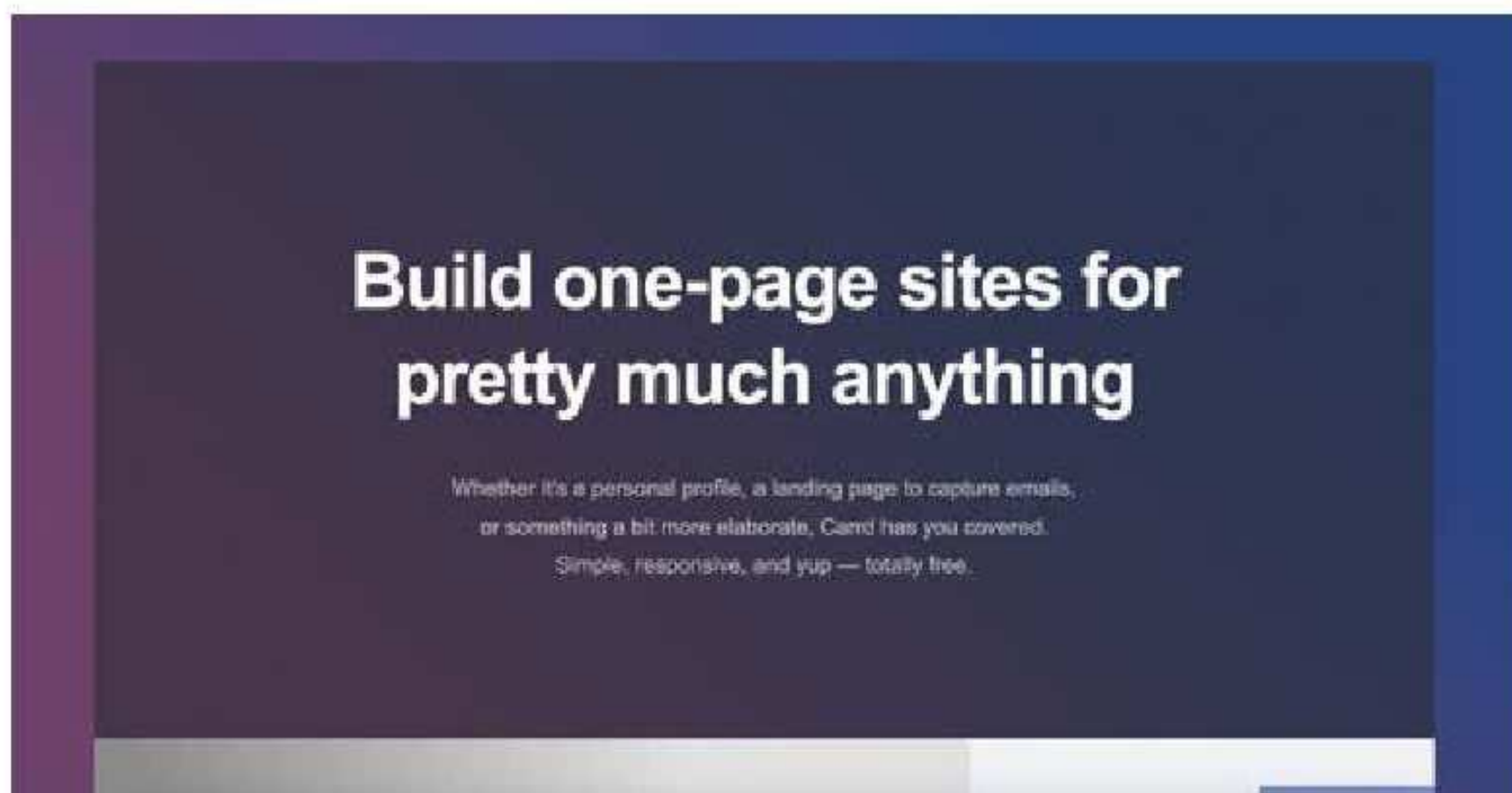
## Building a startup without coding

What if you really don't want to learn to code? That's what this part is about. You can still do it. I'll show how you can build a basic prototype with off-the-shelf tools. You'll be able to make a landing page, let users enter data, manipulate and process it, charge them money, message them and add a task for your contractor (or you) to execute, and without writing a single line of code.

I'll discuss tools to use for each section and give some examples. These tools are obviously subject to change and might be out-of-date. If they are, the general concept remains. I'll give you some guidance. It's up to you to connect everything and execute. Be creative!

### Building a landing page

To get your users in you need a landing page. Luckily these days they're easy to build with existing website builders that give you templates to customize.

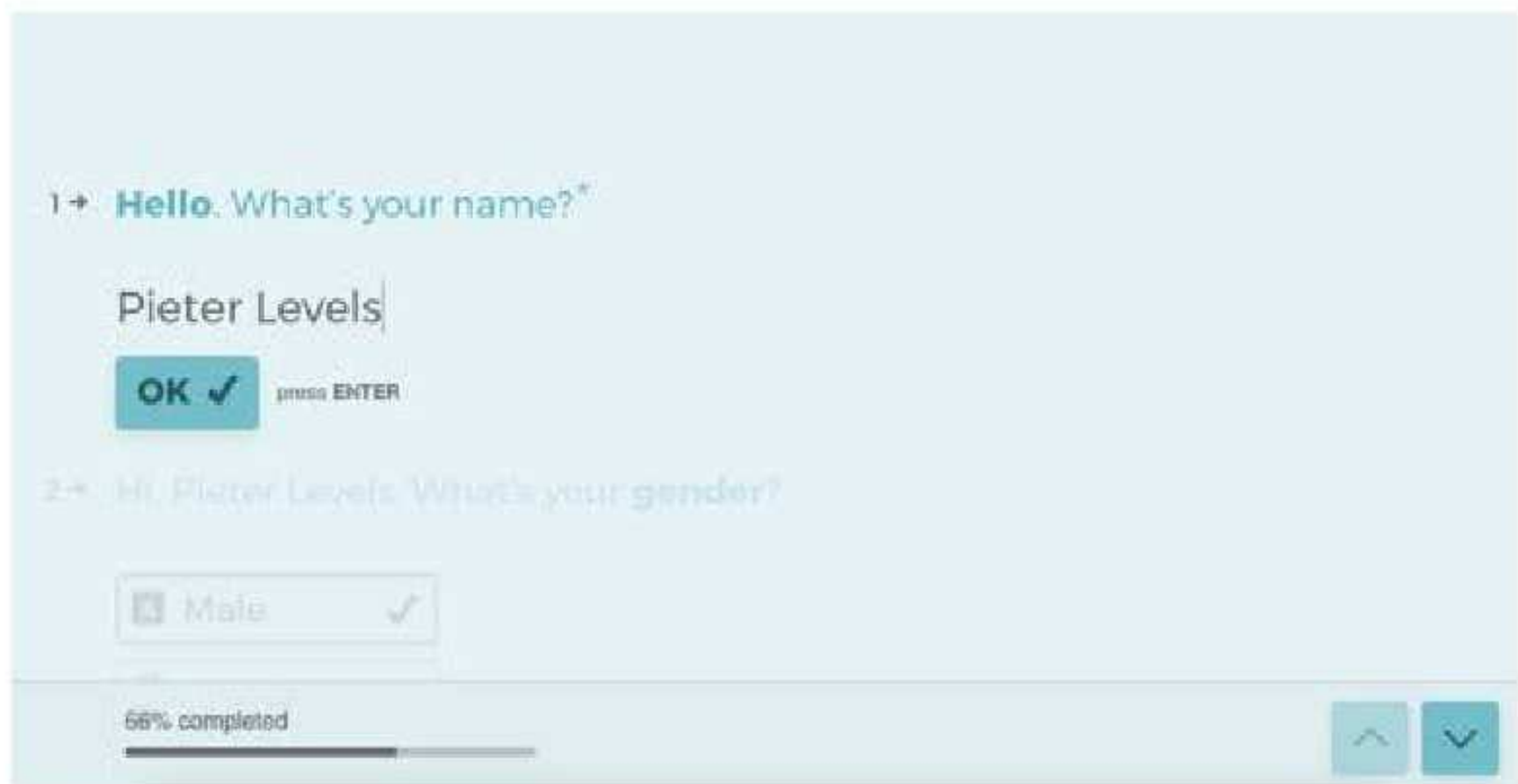


One of the most famous is **Squarespace**. A more recent indie website maker is **Carrd**. Others are **Tilda** and **Wix**. If you need a bit more freedom and the ability to add custom code later, try **WordPress**, it allows you to write PHP or JS to customize your website and add features later on easily.

You'll want to use your landing page to explain your product or service. And from there lead them towards a so-called call to action (or CTA). What do you want from your users? Do you want to save their name and email? Do you want them to pay you money? Adding a big colored button in the center top of the page as a call-to-action will lead them click there. When they click, link them to the next part (which in most cases means, collecting data from your user).

## Accepting user data entry

To get people to enter data, and then save it, you can use **Typeform** or **Google Forms**. Typeform has better forms, but Google Forms lets you directly put the data in a Google spreadsheet, which is great.



1 → Hello. What's your name?\*

Pieter Levels

OK ✓ press ENTER

2 → Hi, Pieter Levels! What's your gender?

Male ✓

66% completed

Google Forms is a bit less intuitive and more formal:

1 → **What's your address? \***

Apt 2F, 1264 Broadway, New York City, NY, United States

OK ✓ press ENTER

2 → **What day should we pick up your luggage? \***

(YYYY-MM-DD format)

2019-06-21

3 of 7 answered Create your own Typeforms... ↑ ↓

The good thing about Typeform is it also supports Stripe. So we can also charge the user's credit card from within (!) the form:

3 → **Pay \$129.99 \***

Your credit card will be charged: \$129.99  
We never store your Credit Card number or CVC number!

Secured by **stripe**

4. **Please enter your Credit or Debit Card number: \***

5. **The CVC number: \***

2 of 7 answered Create your own Typeforms... ↑ ↓

Now let's go into Zapier and make sure that after payment happens we do a few things. This Zapier zap only runs if a new paid entry hits the Typeform.

We automatically send an email to the customer using Gmail with information about the pickup:

✓ Process new pickups from Typeform

Add a note



Subject (required)

Your pickup is scheduled!

Body Type (optional)

plain

Body (required)

Hi

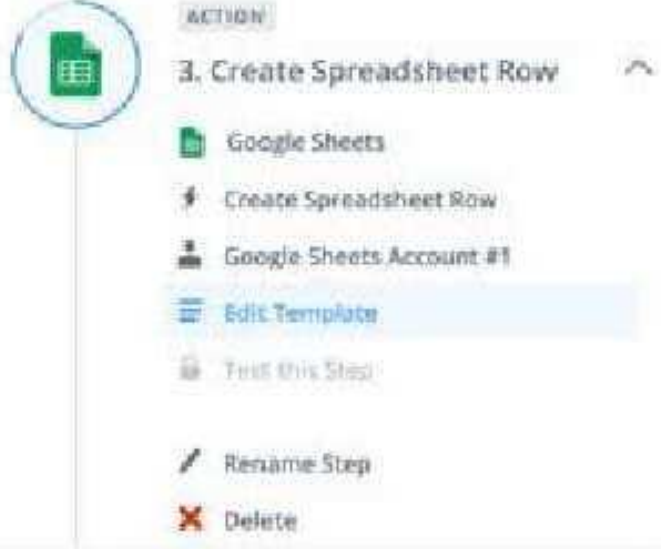
Your pickup is scheduled!

Label/Mailbox (optional)

We automatically add the pickup order to our Google Sheet with active pickups:

✓ Process new pickups from Typeform

Add a note



Set up Google Sheets Spreadsheet Row

Spreadsheet (required)

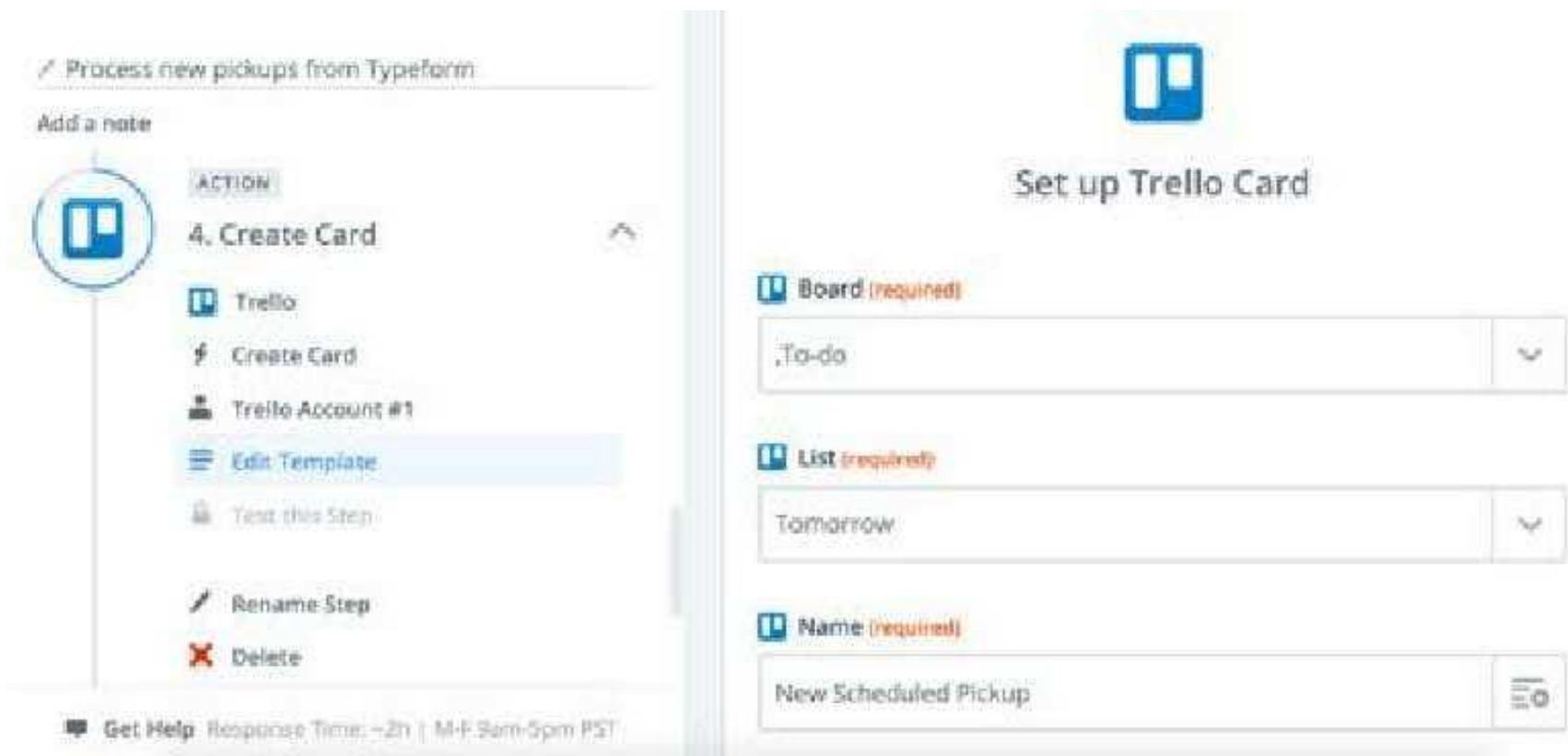
Untitled spreadsheet

Worksheet (required)

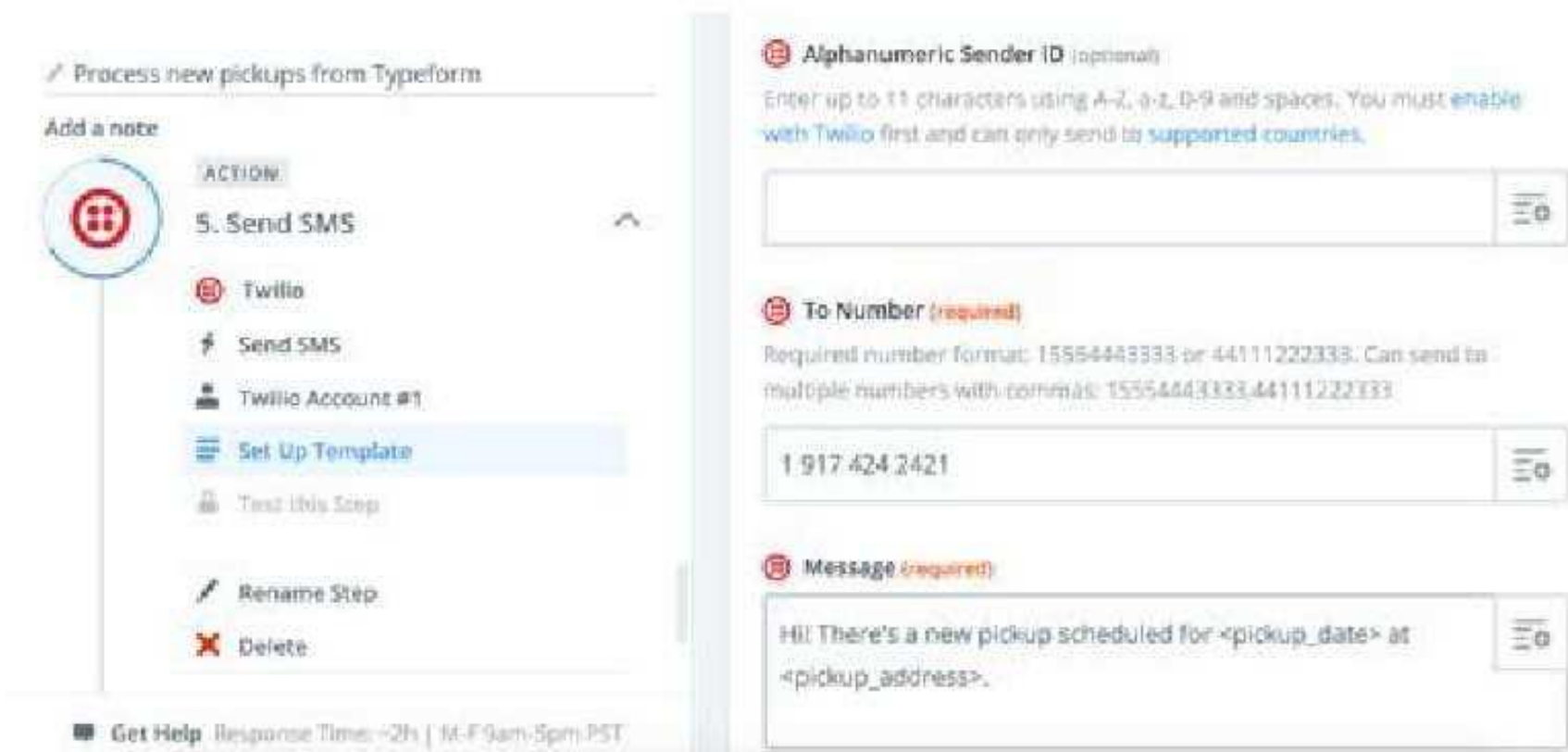
Sheet 1

City (optional)

We automatically add a task into Trello for our pick up contractor to get the luggage:



We also automatically SMS our contractor an alert with Twilio that there's a new pick up ready to be fulfilled, with the date and address:



See how easy that was? We now have a fully functioning minimum viable product (and actually a basic startup) built. And it took about 30 minutes! You can go a lot more complicated from here. The possibilities are endless. And when you have validated your non-code MVP, you can start adding your own coded parts too to increase the complexity of your product. You can replace the "off-the-shelf" web apps with your own coded scripts.

# Let's talk APIs

The entire previous section where we connected all these web apps without code is only possible because of the existence of so-called APIs. They're Application Program Interface. Which simply means apps that can communicate to each other in slightly human-readable data. Zapier's entire app is build around connecting different APIs together.

You don't need to use Zapier to use APIs though. If you can code, you can query APIs too and save their data, and in turn send it to other APIs.

One of the easiest ways to quickly build a prototype (if you can code) is to use thse third-party APIs. They can provide you with data (like cost of living or weather data), platforms to build on (like Facebook and Twitter) and services (like sending emails easily).

## Why are they useful?

APIs are ways that computers and servers can share data in a computer readable format. It means that when I open <https://nomadlist.com/amsterdam-netherlands> a computer will literally read this:



```
<!doctype html><html class="nomadlist no-js pageType-cities user
units-metric "><head><!--<script type="text/javascript"
src="https://www.google.com/jsapi"></script>--><meta
charset="UTF-8" /><title>Nomad List - The Best Cities to Live
and Work Remotely</title><script
src="//use.typekit.net/zfz4zmf.js"></script>
<script>try{Typekit.load();}catch(e){}</script><meta http-
equiv="Content-Language" content="en-us" /><meta name="google"
value="notranslate" /><meta name="viewport"
content="width=device-width, initial-scale=1.0, maximum-
scale=1.0, user-scalable=no"><meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1" /><meta name="apple-mobile-web-app-
capable" content="yes" /><meta name="apple-mobile-web-app-
status-bar-style" content="black" /><meta http-equiv="cleartype"
content="on" /><meta name="HandheldFriendly" content="true" />
<meta name="description" content="The best cities to live and
work remotely for digital nomads, based on cost of living,
internet speed, weather and other metrics. For startups that
work remotely and digital nomads." />
```

...etcetera (it goes on for pages). This stuff is nice to look at when the computer visualizes it and interprets the HTML code. But it's hard for the computer to get specific data from it. Because the layout is kinda made for humans.

But now, if I open the same page through an API at <https://nomadlist.com/api/v2/amsterdam-netherlands> (this URL will probably not work anymore when you read this book but you get the point), you and I, and even a computer, will be able to read this:

```
{
  "name": "Amsterdam",
  "country": "The Netherlands",
  "cost_of_living": "$2,500/m",
  "safety_score": 4.3891,
  "nomad_score": 4.2183,
  "fun_score": 3.394,
  "cost_score": 2.238
}
```

Now you can get the city name Amsterdam, the country The Netherlands and how fun it is there 3.394 (out of 5 stars) and display it on your own site. That's useful. But even more useful is if you combine multiple APIs together.

What if I want to find the most fun places from Nomad List, and get the live weather data from somewhere else? To see where it's fun and not raining?

Well, just a quick Google for "weather api" reveals there's a free weather API called [OpenWeatherMap.org](https://openweathermap.org/) which can give you the current weather and forecast in a computer readable format, just like above. Let's try it:

<http://api.openweathermap.org/data/2.5/weather?q=Amsterdam>

```
{  
  "name": "Amsterdam",  
  "coord":  
    {  
      "lon": 53,  
      "lat": 4  
    },  
  "weather":  
    [  
      {  
        "main": "sunny",  
        "description": "clear sky and sunny"  
      }  
    ],  
  "main":  
    {  
      "temp_celsius": 25.5,  
      "humidity": 89,  
      "pressure": 1013,  
      "temp_min": 20.04,  
      "temp_max": 26.04  
    },  
  "rain":  
    {  
      "3h": 0  
    },  
  "clouds":  
    {  
      "all": 5  
    },  
}
```

Yay, it's not raining in Amsterdam! And it's actually 25 degrees celsius! That means we can go outside and have some drinks :)

Now we can combine that weather API data, with the data from the **Nomad List** API and show it on our site. But we can go much much further.

With API services like **Twilio.com**, we can make an entire telephone voice and SMS service by just making calls to their API. I'll spare you the technical details as it's quite some work. But it's not difficult. It's simple and anyone can set it up.

Now start thinking of ideas. You can have a phone number you call that'll ask you some questions over voice, and even have the person answer them (yes Twilio has Speech-To-Text and Text-To-Speech). You can call that number to ask what's the city you should go now with the best weather and most fun.

The more APIs you start combining, the more fun it becomes. And because the heavy load is lifted by the APIs already, you are pretty much just connecting them together into new functionality. Which can be your product!

## **You can build a business on other people's API's**

You can build entire businesses based on other company's API's. Many have done so before you. There's one big thing to remember though: if you become solely dependent on one company's API, you're in a bad place. Without even telling or asking you, they can shut down their API at any time. And that will immediately destroy your business.

More commonly, companies change how their APIs function whenever they want. That means you have to constantly monitor their API and see if it functions correctly. Whenever they change something (even something as small as changing the key of "nomad\_cost" to "nomadcost"), it will break your app and you have to change your code.

## **Conclusion**

When building, try to build fast and minimal. Instead of learning new stuff, use the tools you already know to build your idea. Make sure your MVP actually works and is not just a landing page that doesn't do anything. Lose your perfectionism, it'll never be perfect any way. Don't outsource, build it yourself. If you can't code, use off-the-shelf tools and connect APIs together to build it. Appreciate your constraints and limitations, they can be a giant advantage vs. people with lots of resources. Keep your costs low while building and later on. Build for the web first, you can go native mobile later. Don't build on an MVP too long, a good rule of thumb is to spend max. one month on it and launch.

## **Resources mentioned**

- **Linode**
- **Digital Ocean**
- **Heroku**
- **Ubuntu**
- **NGINX**
- **Olark**
- **Twilio**
- **Zapier**
- **WordPress.com**

## **Your homework**

- Rank the list of ideas you made previously by which you think are best.
- Now see which of those best ideas you can execute quickest with the tools and skillset you already have right now — that means without learning anything else now.
- Build the first prototype of your idea, it's minimal, but that doesn't mean not functional. It should do something, either it being a Typeform connected to Zapier or a WordPress landing page or your self-programmed web app or native app. It should have the core functionality working well to be useful for users.

of them adds a little chat box on the bottom right of your app, where you can chat with users directly. And if you're not there, people can leave a message.

I don't suggest being on-call at all. I'd suggest using it just so users can leave a message. The messages you'll get vary. I get a lot of very fun ones. Mostly it's people thanking for making a website and that I should add a new feature. Because they enter their email address, I am usually able to reply quickly. AND many of these feedback messages turn into entire email threads full of feature discussions with random people from the internet! It's pretty amazing.

It's also very useful if you break something. I accidentally break my sites A LOT and many times the moment after I deploy the new version with a bug, I'll get a message within 10 minutes through my feedback box that I broke something. I wouldn't even know what to do if I didn't have that. It's become essential to my workflow even with startups I have running for years already. Things break, and you can't always test for them, so let your users be your 24/7 continuous testing army!

## **Where to launch?**

### **Typical places to launch a startup**

Traditionally, startups have launched in the tech subculture first. That works because tech people are early adopters and open to use new apps. They also care less about bugs than mainstream people will.

Platforms like Product Hunt, Hacker News and even Reddit are geeky but reach a huge mass of people. They're the most obvious places to launch new startups, products and apps.

Should you launch there though?

Many people say you should launch where your customers are. And that's a great point. If you make a food delivery service for pets. Are your then customers really on Hacker News? Or Product Hunt? Or Reddit? I guess, they

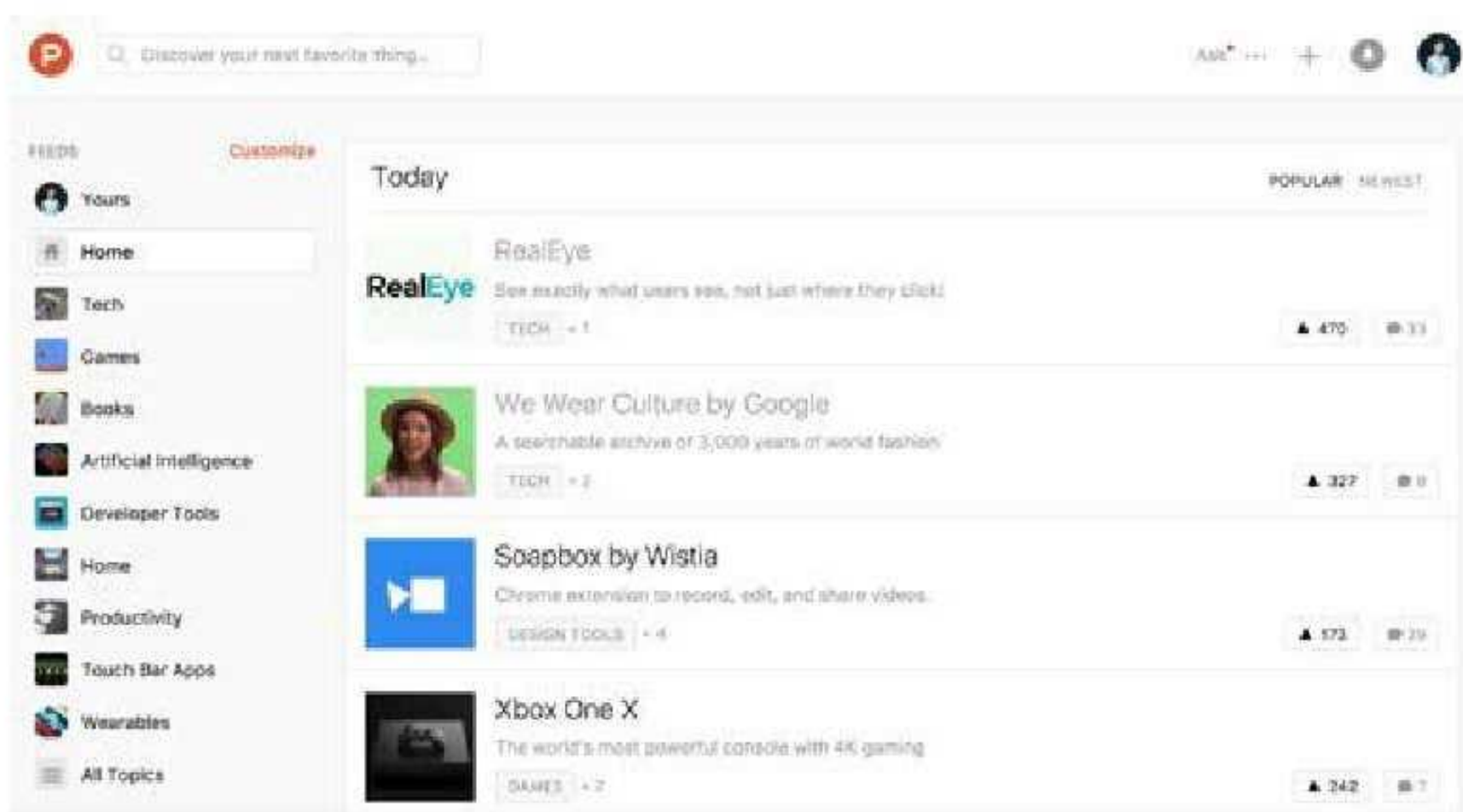


could be on all of those. But it'd be probably smart to find an place where pet owners hang out. That could mean advertising on Facebook targeting pet owners, or reaching out to online forums for pets. I have no idea. The point is, in that case you should target your audience specifically.

So there's a big argument to be made to not launch in the tech subculture. But then more recently this is changing though. Now that the whole world is starting to get smartphones, it seems tech has become mainstream. Everybody is obsessed with apps and startups. And in that sense, those startup platforms are actually a great place to launch. It really depends how you look at it.

## Launching on Product Hunt

Product Hunt was founded in 2014 by [Ryan Hoover](#) as an email newsletter and has since become one of the most important platforms to launch a startup. Product Hunt is a daily leaderboard (kinda like Reddit) where people can submit and upvote new startups. In 2014, Product Hunt single-handedly instigated an entirely new cultural wave of indie makers, a wave of which I'm thankfully a part of. By letting anyone submit their app, it lets indie makers compete with giant VC-backed startups for attention.



It has a community of people that are startup fans, app makers and mostly just people curious to see the next big thing. There's a lot of early adopters that are literally on the site to discover and try new apps.

In general, being on top of Product Hunt will get you around 10,000 people visiting your site, with around 300 simultaneous users, with a percentage of those signing up. Product Hunt traffic may convert less to paid users than normal traffic that arrives on your site by search. That's logical, because those users have a set goal. Instead, Product Hunt users are curious and mostly just visit to "take a peek".

Product Hunt's comments on products are generally very supportive and positive. But they're not radically honest (and painful) like Hacker News. You have to take them with a grain of salt.

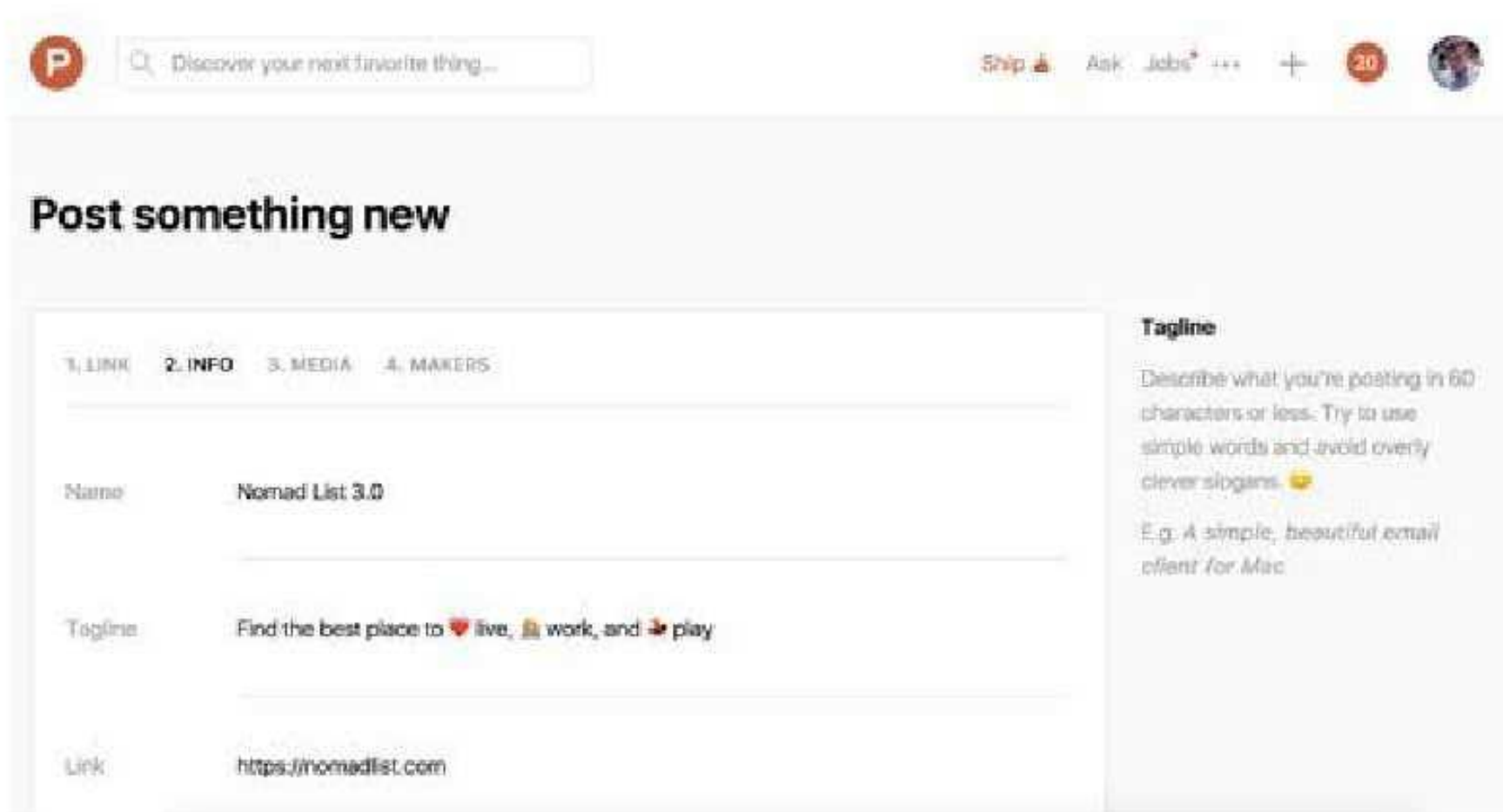
The most important selling point of Product Hunt for me is that it's packed with tech journalists trying to find their next story. Usually what happens just hours or a few days after a Product Hunt launch is that you'll see your app show up in articles in the tech press all around the world. That alone is a great thing and probably gets you another 50,000 people to visit it over the next few weeks, depending on the media outlet.

### **Submitting to Product Hunt**

You only have one opportunity to submit to Product Hunt in a long time (you can't just keep submitting obviously). Therefore, you have to make sure you do things the right way. I've seen too many startups work for months on their product, to then absolutely fail at launching on Product Hunt by underestimating how hard it is.

The timing of a launch on Product Hunt is quite important. Product Hunt's timezone is San Francisco (PST timezone) and its ranking resets at midnight. That means you will need to launch at 00:00:01 or not much later. If you submit at 9pm PST when there's 3 hours left to vote, Product Hunt will take that into account somehow with their vote algorithm, but it still simply puts you at

a disadvantage. Most people know by now to launch at midnight PST, so you won't be the only ones up there then. Due to timezones, I've had to stay up or wake up early several times just to submit to Product Hunt at the right time in PST.

The image shows a screenshot of the Product Hunt website's 'Post something new' form. At the top, there is a search bar with the text 'Discover your next favorite thing...' and a navigation menu with 'Ship', 'Ask', 'Jobs', and a plus sign. The main heading is 'Post something new'. Below this, there are four tabs: '1. LINK', '2. INFO', '3. MEDIA', and '4. MAKERS'. The '2. INFO' tab is selected. The form fields are: 'Name' with the value 'Nomad List 3.0', 'Tagline' with the value 'Find the best place to ❤️ live, 🏡 work, and 🎮 play', and 'Link' with the value 'https://nomadlist.com'. On the right side, there is a 'Tagline' section with instructions: 'Describe what you're posting in 60 characters or less. Try to use simple words and avoid overly clever slogans. 🙅' and an example: 'E.g. A simple, beautiful email client for Mac'.

Set a proper name. If it's the first launch on Product Hunt, just use your app's name. If you already launched (like a year ago) and this is a big new version, use 2.0 or 3.0 etc. Be sure to inform Product Hunt's community managers about this, because sometimes they don't allow you to repost the same product.

The tagline is like a motto or slogan. Make sure you put some thought in to this. The most common problem with bad taglines is that they're too complex to understand for non-users of your product. What does your app do? It uses machine learning to make your photos better?

I see many people For example "An algorithmic application for machine learning applied to photos" may be accurate but is terrible for Product Hunt. Is it the first app to do this? Okay, so what about "The first machine learning photo editor". Then add some emojis: "The first 🧠 machine learning 📷 photo editor".

Thumbnail

https://

or Upload an image

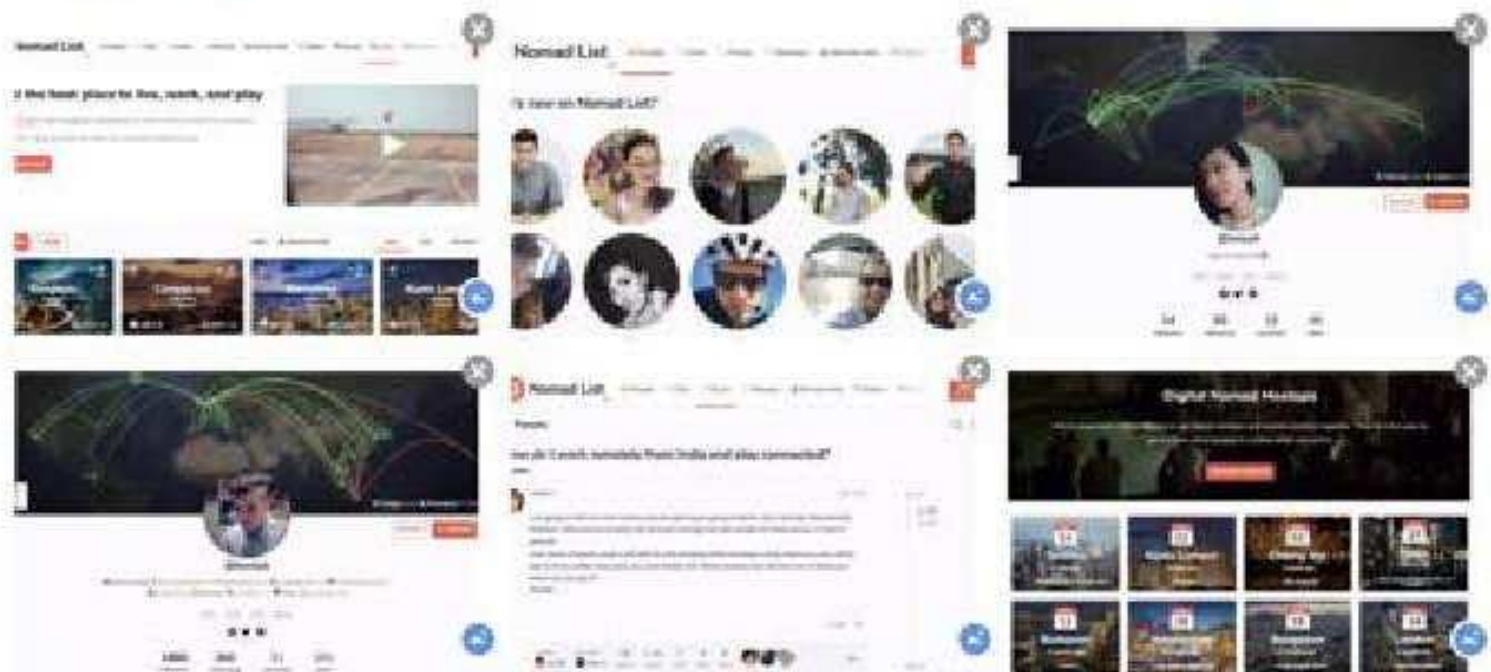


The thumbnail is as important. Product Hunt allows you to upload animated GIFs. This means you can even convert a video to a GIF (as a square box) and use that. Be sure it's only a few MB. A good app to capture videos to GIF I use is **GIPHY Capture**.

Gallery

https://

or Upload an image



Make about 8 to 16 high-res screenshots of your app. Zoom in the browser on your website to make sure the screenshots are crisp. They're displayed scaled down on Product Hunt so it makes sense to zoom in a bit.

Show the core functionalities of your app in the screenshots. Make it eye catching.

Product Hunt also allows you to add a video now. It's auto played (with muted audio) when your product page is opened on desktop. This is a great opportunity to make a short thematic promo or explainer video. Make it 30 seconds or shorter, because people don't have much time. Capture people's attention quickly and make sure the video gets across what your app is about.

### **When it's up there**

I usually send out a tweet on Twitter, share it on Facebook and Instagram showing people that my new app is on Product Hunt. I explicitly do not ask them to upvote it. But they might do if they like it. And that helps getting up in the ranking.

If you have an email list, you can even send a message to them to tell you're on Product Hunt. Again, don't ask people to upvote. They'll do it if they like it any way.

I'll also immediately jump in to the Product Hunt comments and write an introduction post about who I am, why I made this app and what's my future plan with it:



**Pieter Levels** HUNTER

@levelsi0 • 🇳🇱 Serial maker 📍 Nomad List + et al

Hey Product Hunters! I made this new thing called Hoodmaps. Here's the problem I had, and the solution I thought of:

#### 🤔 Problem

The problem is that every time I travel to a new place it's hard to figure out which parts of the city to go. I very often end up in the tourist center. I'm originally from Amsterdam and I know 90% of tourists will never get any idea about the "real" Amsterdam because they just stay in the tourist center. It's a fake area that has nothing to do with Dutch culture.

So what do I want? I want to get a quick overview of what a city is about. What are the cool "hip" areas? Where's the wealthier areas? What areas are more suburban (and maybe boring 🙄 for me?)

#### 🔧 Solution

When my friend @generic\_dreams asked me where to go in Amsterdam I drew a map:

[https://twitter.com/levelsi0/sta...](https://twitter.com/levelsi0/status/1111111111111111111)

I thought, maybe I can make this into an app. So I made a Google Maps map, and you can draw colors on it. And every color represents a different category. I have 🧑 hipsters, 🗺️ tourists, 💰 rich, 🧑 normies, 📄 suits and 🏠 uni area colors. And it uses everyone's drawings as crowdsourced data. So if 5 people draw over an area that it's tourist, but 8 draw over it that it's hipster, it'll show up as hipster.

I then stick around in the comment section to keep answering any questions people have. And see if people have feature suggestions.

It's important to be polite and humble when you're in the comments section. I see a lot of people pose, brag and market themselves when they're up there. And people don't like it. It looks sleazy. You're there to take people in as guests into your app and treat them well and hear their feedback and improve based on that. Not to sell!

Different platforms have different degrees of hate. Product Hunt is famous as one of the more positive platforms to launch apps on as it's fiercely curated. That might mean you'll get nicer people but less real feedback than e.g. Hacker News.

## Launching on Hacker News

**Hacker News** is probably the hard "core" of the tech and startup scene.

Just like Product Hunt, it's a leaderboard. But it doesn't just contain startups. It also has blog posts about new technology, startups, funding rounds and venture capital.

Hacker News is open to anyone's submissions, but that makes it extremely competitive though. It also has lots of protections against spam, marketing, voting rings and even controversy. For example, if a post receives more comments than upvotes, it will quickly be flagged (automatically) and disappear from the frontpage, that's the controversy filter at work. If Hacker News detects you're forwarding your post to your friends to upvote, it'll discount its votes and it'll just drop it to a few pages further. ATherefore, it takes quite some work and luck to get on Hacker News' frontpage.

Where Getting on Hacker News is already hard, staying there is a war in itself. Hacker News has aggressive moderating, which means if a mod doesn't like your post, it'll simply discount the upvotes too and it'll fall off the frontpage. The whole goal of HN is to keep the content very specifically relevant to its audience and the current zeitgeist. You need to consider that. Many makers make things specifically for the Hacker News zeitgeist. For example, now that I'm writing this, there's a lot of pushback about Facebook's ruining of society and addicting its users to newsfeeds. Making an app around this that solves this will probably get high on Hacker News (if someone else didn't already)≥

As with Product Hunt, the title is very important here. Let's say we're making that food delivery app for pets. We call it Petsy.com. We submit it to HN like this:



Petsy.com - The best food delivery for pets

It might work. But I think it probably won't.

A better way is to make it personal like:

The same applies to buying fake virality. It won't work. It looks good on paper but it's not actual humans. And people are less and less impressed with seeing big numbers on somebody's social media account. More important is, again, what your app does.

### **So why doesn't this work**

Well, it actually does work. For a while. You can create artificial traction. The problem is that it doesn't stick. You'll have to keep adding more fake virality and the odds of it being picked up by actual REAL humans is quite small.

Because if your app was good from the start, it wouldn't NEED any artificial following. It could push itself just by being great. Because if people like something, they'll share it themselves, they'll make it big. Especially if it's something unique and original that does something better than it has been done before.

Because we're sick and tired of fake stuff. This goes much deeper than just apps and sites. The whole Zeitgeist is now about people looking for authenticity, because everything around us is fake. We like people that are open, honest and real. Why? We are bombarded by bullshit marketing messages. Why do you think we're all installing ad blockers? We want purity. We want stuff to be authentic and organic! So what's the best way to get those people to follow you? Be real!

### **Be organic**

So the answer is, be organic! Be real. Create traction by making a great product that is considerably better than the competition, easier to use and more original.

The big advantage by launching organically is that you will see if your app doesn't work. If it doesn't get traction, there's a good chance it's simply not good enough. The idea might suck. The interface might be bad. Maybe the idea isn't original enough. You'd never be able to find that out by faking it!



## Telling your story

### Blog

People might call this "content marketing", but I feel this is a sleazy term. It means that you're writing something with the goal of selling something. I don't like that. It should be real.

I started blogging years ago and pretty much the preface of [Nomad List](#) was me writing about traveling for 1.5 years. And randomly, because of that I had a small audience of people that knew me, and that just grew a lot and helped Nomad List's launch a lot.

Blogging in that way gives you the opportunity to build up an audience even before you launch.

I have a hunch though that text is becoming less important (lol writing this in a book). But I think it's true. Look at the rise of Instagram and Snapchat vs. Twitter and Facebook. People are getting too lazy to read lots of text. Probably only a few people will actually read this entire book. I wonder how many actually will read this sentence!

So there's a case to be made that telling your story should happen in a different, or at least, in more media than just text alone.

### Press

#### Why press matters

The importance of press, is that it gets you lots of mainstream users outside of your niche bubble. Especially non-techy people that are hard to reach through platforms like Product Hunt.

#### Why press increasingly matters less

Just like I wrote before about blogging, I think people read press less and less. Again that's a hunch but I think I'm right. The rise of Product Hunt is an exam-

ple of a platform that pretty much took a big bite out of the tech press pie by automating it.

Then there's the issues with the deep mistrust people have of press these days. Readers aren't sure if what they're reading is a paid advertorial or actually a real article about a company. Startup founders are scared to talk to journalists in fear of becoming part of clickbait hit pieces killing their careers.

You might need press to kickstart traction, but you definitely need it less and less when you have traction already. Especially in these times.

If you do use the press, make sure you control the conversation. You set the angle of the story. Check what type of articles journalists write before you talk to them. Fact check the articles they post about you or your product after and ask for corrections if necessary. Press can easily flip a story to become negative about you and your product in favor of page views. The press is not necessarily your friend, as much as they like to make it sound like it when they talk to you.

## **How to get press**

### **Make a press list**

The reason most people don't get press is because they simply contact the main email address of a press outlet like *tips@thenextweb.com* with their app's pitch and URL. Here's a little secret. Those email addresses are the industry's black holes. A lot of stuff goes in them. Not much comes out.

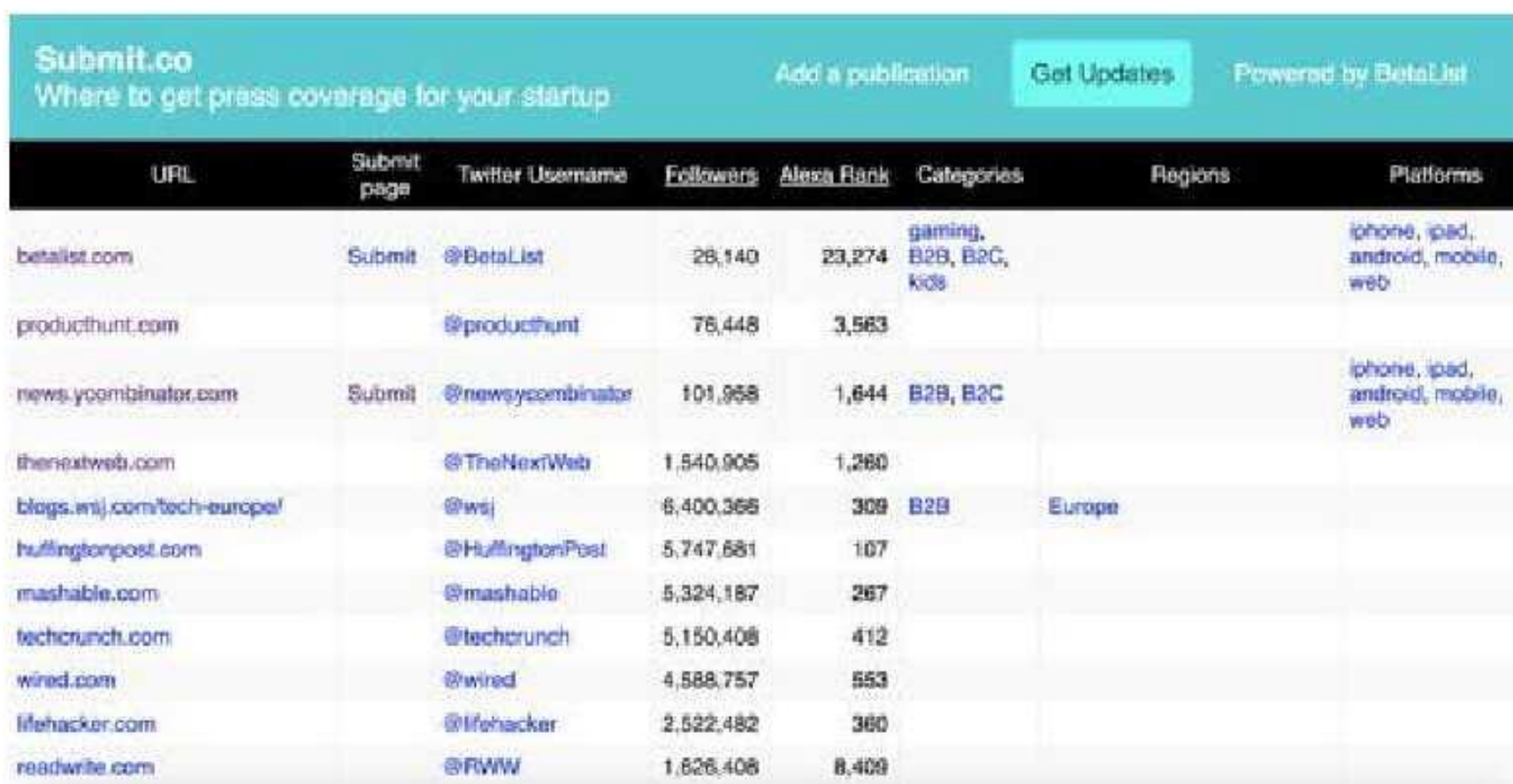
The way to get a journalist to write about you is to make it personal. How do you do that? Well. First figure out which journalist would actually be interested in your app and find it useful. Every journalist has their own style, taste and personality. And niche.

There's (tech) journalists who mostly write about cryptocurrencies, or only about travel apps, or only about bootstrapped companies. Find journalists that are relevant for your app by searching who writes about your competitors or apps in your industry. Just doing that, already gets you further than 99% of

people who don't take the time. Shooting 1 targeted email can result in more success than 1,000 untargeted emails to every single journalist out there!

If you know a journalist that already wrote about a previous app you made, that's a warm connection and it's *very* valuable. Keep them in the loop about your projects but don't bug them. They'll be happy to write about you again if they like what you're doing. Update them about significant stuff, and give them scoops, that means exclusive news you don't give anybody else. That's currency for journalists.

Based on all of this make a press list of journalist you want to approach. A fair estimate is that less than 5% will write about you. Even if you target it well. So to get 5 press features, you need to contact 100 journalists. That might even be a positive estimate! It might take hundreds of targeted (!) emails.



URL	Submit page	Twitter Username	Followers	Alexa Rank	Categories	Regions	Platforms
<a href="http://betalist.com">betalist.com</a>	Submit	@BetaList	26,140	23,274	gaming, B2B, B2C, kids		iphone, ipad, android, mobile, web
<a href="http://producthunt.com">producthunt.com</a>		@producthunt	76,448	3,563			
<a href="http://news.ycombinator.com">news.ycombinator.com</a>	Submit	@newsycombinator	101,958	1,644	B2B, B2C		iphone, ipad, android, mobile, web
<a href="http://thenextweb.com">thenextweb.com</a>		@TheNextWeb	1,540,905	1,260			
<a href="http://blogs.wsj.com/tech-europe/">blogs.wsj.com/tech-europe/</a>		@wsj	6,400,366	309	B2B	Europe	
<a href="http://huffingtonpost.com">huffingtonpost.com</a>		@HuffingtonPost	5,747,681	107			
<a href="http://mashable.com">mashable.com</a>		@mashable	5,324,187	267			
<a href="http://techcrunch.com">techcrunch.com</a>		@techcrunch	5,150,408	412			
<a href="http://wired.com">wired.com</a>		@wired	4,588,757	553			
<a href="http://lifehacker.com">lifehacker.com</a>		@lifehacker	2,522,482	360			
<a href="http://readwrite.com">readwrite.com</a>		@RWW	1,626,406	8,409			

How do you find these journalists? Luckily here's a few directories to find contacts for journalists. One website to find tech journalists in specific is **Submit.co** lets you find a major amount of press outlets. From there you can go on to find specific journalists.

Put every journalist in a spreadsheet and personally contact them. Don't bug them! In the list, write down how far you are with them. For example,

"emailed?", then "replied?", then "will feature?" and keep track. Don't underestimate this. This takes work especially if you're new, nobody knows you and you don't know anybody either.

### **How to approach journalists**

Journalists generally hate if you approach them through Twitter, Facebook or LinkedIn. They still use and read email, just not in the way you do. They skim through it very fast. The best way to get them to stick is to keep your approach concise and powerful.

I see so many people write lengthy emails about why their app is the best thing ever, but hardly anybody is able to argue why it's great. What's the value for people? And they're all lengthy. Journalists don't have time for that stuff!

Here's an example of a bad email that is very typical:

Subject: New startup Petsy.com

Hi TechCrunch!

We just launched our new startup Petsy. It's funded with a Series A round led by Lion Capital.

We're growing like crazy, crushing it right now and we'd love to tell you more about our launch this week.

Petsy connects pet owners and pet food. It works like this, you go to the app, then you select which food you want. Then you tap order. It then delivers the food to your house. We have about 5,000 people in our app and growing 40%. Thanks for covering us!

-Rick CEO, Founder Petsy.com

*This e-mail is intended for the addressee shown. It contains information that is confidential and protected from disclosure. Any review, dissemination or use of this transmission or its contents by persons or unauthorized employees of the intended organisations is strictly prohibited. The contents of this email do not necessarily represent the views or policies of Rick Andrews.*

Why is it bad?

- It's way, way, way too long, ain't nobody got time for this
- It gets to the point way too late
- Subject is vague
- It's not personally written towards the journalist

- It's full of vague bragging instead of facts (e.g. "we're growing like crazy")
- It uses buzzwords like "crushing it" and jargon like "connecting pet owners"
- It's full of ego and business lingo (like CEO, Founder, of what?)
- There is no clickable link to your startup
- It has a ridiculous disclaimer
- If you look at it from a distance, it's just a block of text

This is not good. But this is what journalists get by default.

So how *should you* approach a journalist?

Most importantly: be concise. Keep your email to one or two sentences max. Here's a good pitch to Susan from TechCrunch, who you know is interested in food or pet industry startups:

Subject: Food delivery startup for pets

Hi Jody! I made a site that lets you subscribe to food delivery for your pet. Let me know if you need more info :)

**<https://petsy.com>**

It's friendly, fast and clear. The journalist has the opportunity to check it out themselves without you selling it to them. And yes, if they like it, they'll reply and ask you for more info! That's when you can give all your data and info.

Be sure to reply to them ASAP as they're impatient. They might be waiting to write an article on you and if you reply immediately, they can get on with it. If

and press outlets. Most importantly, a launch is not finite. If you'll have a product running, you'll keep launching repeatedly every few months or years when you have specific big new feature developments.

Nowadays, the launch is perpetual.

## **Resources mentioned**

- **MailChimp**
- **Google Analytics**
- **MixPanel**
- **Amplitude**
- **Hotjar**
- **Product Hunt**
- **Hacker News**
- **Reddit**



## **Your homework**

- Make a list of places you will launch your first product, these can be the typical ones like Product Hunt, Hacker News and Reddit, but make sure there's more niche platforms on there too.
- Write a title and description for each platform you will launch too. Personalize it for each platform depending on its audience. Make sure it doesn't look spammy.
- Do a final check that your product actually works. Can a new user immediately see what it is about and start using it? Go through the process multiple times to make sure every little bug is gone.
- Pick a day and time and launch!



**Grow**

	Single payment \$75	Subscription \$75/year
Year 1	\$75,000	\$75,000
Year 2	\$93,750	\$168,750
Year 3	\$117,187	\$379,687
Year 4	\$146,484	\$854,296
Year 5	\$183,105	\$1,922,167

See how fast recurring revenue grows? Obviously, this is without people canceling a subscription (that's called churn). Let's be pedantic and add churn of 7% per year. That means 7% of users cancel their subscription yearly.

	Single payment \$75	Subscription \$75/year
Year 1	\$75,000	\$75,000
Year 2	\$93,750	\$163,500
Year 3	\$117,187	\$356,430
Year 4	\$146,484	\$777,017
Year 5	\$183,105	\$1,693,897

See? Even with a moderate churn, it'll still grow much faster than a single payment does. After 5 years of single payments, 25% annual user growth and 7% churn you'll be making \$1,693,897 of revenue! Versus only \$183,105 if you use single payments. The difference is \$1,510,792. Recurring revenue is *very* powerful.

There's a very good reason why entrepreneurs like this. It's easy to sell a company like this. Acquiring companies love subscription revenue as it's highly predictable and relatively stable. If you know the growth rate and the amount of people canceling, you can evaluate the company easily and in turn, value it with a good price.

From the customer's perspective, subscriptions might not be that great though. People spend much more money generally. They often forget to cancel their subscriptions and then they renew for another month or year, meaning they overpay. It's hard to get that money back then. Actually, "forgotten subscriptions" are the elephant in the room of subscription-based businesses, with some assuming up to 50% of subscription revenue are by customers that don't even use the product anymore. Obviously, subscription business owners are the last to discuss this. It's their cash cow. But it's dark. A solution to this would be to specifically ask people to renew their subscription or not after you detect they're not actually using your product but are still paying for it. Nobody is doing this though that I know of yet. Obviously. It'd instantly make the subscription model more healthy for users, and still beneficial for businesses.

What should you offer as a subscription though? Well, anything really. Your app. A service. A forum. A chat group. A social network for a specific niche (like I did). Most apps people repeatedly use work with subscription revenue. You might see a drop in sales though when switching to subscription revenue because subscriptions require commitment. And not everyone wants to commit to another subscription in their lives (besides Netflix). This depends heavily on your audience. Personally, I have become a bit tired of signing up for another subscription.

A good middle ground (that I use) is to offer besides subscription, also a one-time-payment lifetime memberships for the predicted lifetime value (LTV) of a customer. That means if you know on average your users remain members for 2 years and you charge \$100/year. Your lifetime membership will be \$200. That means you'll make the same amount of money as with subscription, they just pay the entire amount up front.

## **Community model**

Subscription payments work especially well with the business model of building a community. What's a community? It can be a discussion board, forum,

chat group, social network to physical meetups. I do all of these with Nomad List.

Communities are attractive business models because by default they're niche. You make a community for travelers, or motorbike enthusiasts, people who have horses, fans of Ariana Grande. Whatever. There's endless niches to serve. They're also attractive because if you become the dominant community in your niche, you have a monopoly and you can charge high prices. As with social networks, communities have the famous network effect. The more people you have in a community, the more valuable it becomes (and the more money you can charge). Each user usually has a profile like this:



Operationally communities can be run quite lean because community software like forums is quite maintenance free, you might need a moderator but that's it. On the other hand, you can go as big as you want. I started with a chat, but now have a forum, a location-based social network and physical meetups all over the world. The model is also highly relevant to current time. People don't like to pay for content anymore, but they do like to pay for connections, e.g. communities. Tech makes people more lonely than ever, but it can also connect them to more people than ever. Your community can enable that!

## **The story of Nomad List's paid membership community**

- Browse 100+ job boards for jobs that can be done remotely, and parse the job position, description and company to see if it's a remote job or not, if it is, post it to **Remote OK** and share it on Twitter and Facebook and send it out to people subscribing to email alerts for that specific job tag (like JavaScript).
- Browse my city pages and screenshot part of them to make city thumbnails so that if people share a page on Twitter or Facebook it shows a picture of the city with data printed on it.
- Check every day if there are deadlines passed on GoFuckingDoIt.com and email the supervisor of the goal set if the person really passed their goal, give them a link to click if they didn't pass the goal which if pressed lets the robot charge their creditcard for money.
- More personal: Login to my smartphone telecom provider, download the invoice, print it as PDF and email it to me. Send a copy to my spreadsheet robot who automatically enters it into my bookkeeping.

## What's a robot really?

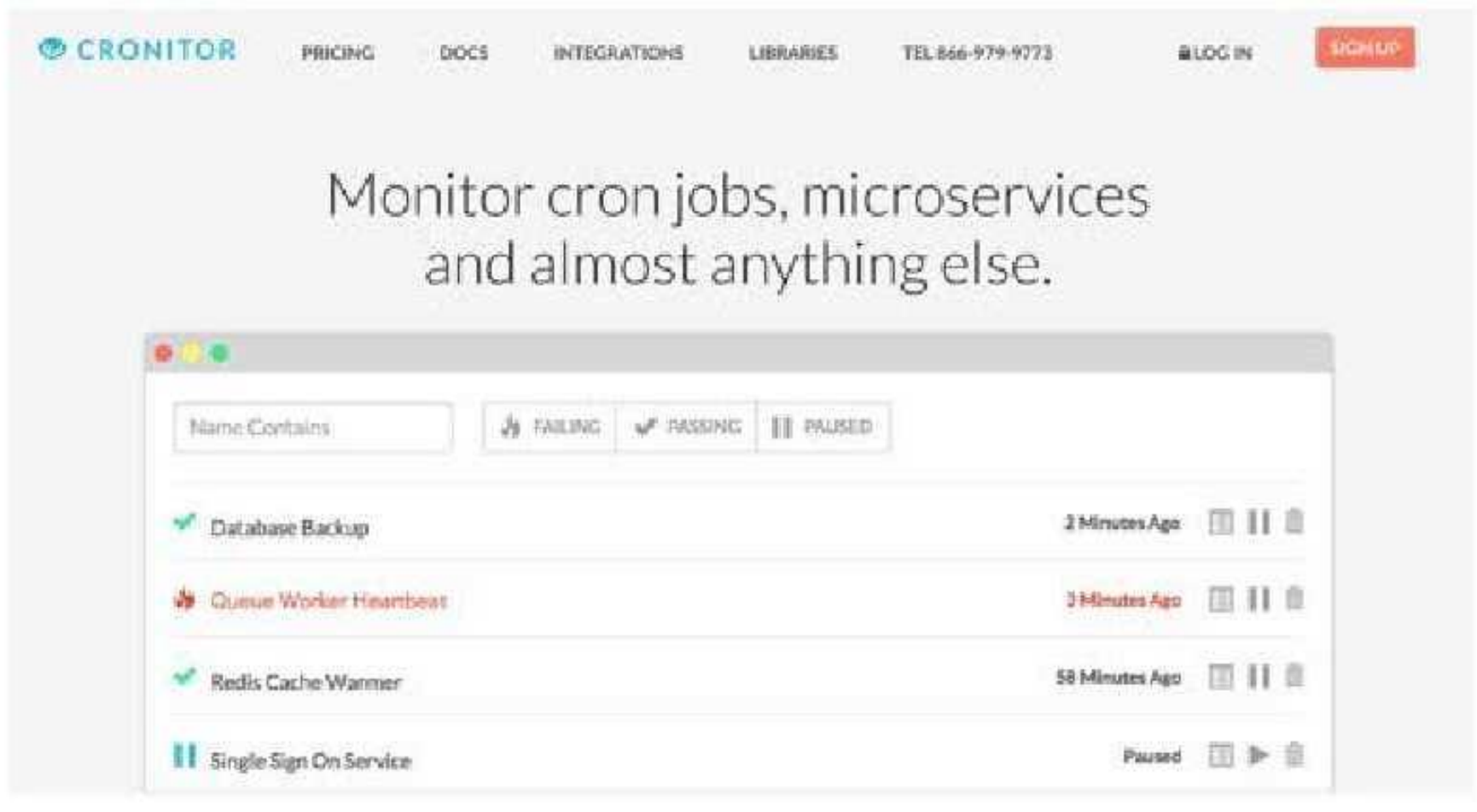
A robot for me simply means a task I used to do myself, that I wrote a script for, then scheduled to run every second, minute, hour, day or week. My robots are written usually as simple PHP scripts that are run in the shell, but I also sometimes use JavaScript on the server for it (with ExpressJS or PhantomJS). It doesn't really matter how you do it. If you can make a script that does something and saves you from doing that task, that's a robot. Here's my server dashboard (called cron) with some of my robots scheduled (the /srv/cronlogger is simply an app to log the result of the scheduled job, the stuff after php is actually the script being run):

```
+*/18 * * * * /srv/cronlogger rinsepodcast "php -f /srv/http/etc.levels.io/rinsepodcast.php -- 'refreshCache=true'"
@ 8 1 * * * /srv/cronlogger sleyerobot "php -f /srv/workers/sleyerobot/sleyerobot.php"
@hourly /srv/cronlogger gfdi-deadlinespassed "php -f /srv/http/gofuckingdoit.com/workers/checkDeadlinesPassed.php"
@hourly /srv/cronlogger kyfr-deadlinespassed "php -f /srv/http/keepyourfuckingresolutions.com/workers/checkDeadlinesPassed.php"
@daily /srv/cronlogger gfdi-deadlinesexpired "php -f /srv/http/gofuckingdoit.com/workers/checkDeadlinesExpired.php"
@daily /srv/cronlogger kyfr-deadlinesexpired "php -f /srv/http/keepyourfuckingresolutions.com/workers/checkDeadlinesExpired.php"
@daily /srv/cronlogger gfdi-staticpage "php -f /srv/http/gofuckingdoit.com/app/index.php > /srv/http/gofuckingdoit.com/public/3"

# REMOTEOK
@hourly sh /srv/http/remoteok.io/workers/sources/run.sh
@hourly /srv/cronlogger remoteokio-staticpage "php -f /srv/http/remoteok.io/app/index.php -- 'HTTP_HOST=remoteok.io' 'page=front'"
@daily /srv/cronlogger remoteokio-refagjobs "php -f /srv/http/remoteok.io/tools/refagAllJobs.php"
@daily /srv/cronlogger remoteokio-downloadpics "php -f /srv/http/remoteok.io/tools/downloadPicsForAllJobs.php"
30 * * * * /srv/cronlogger remoteokio-queueemails "php -f /srv/http/remoteok.io/workers/queueEmails.php"
@ * * * * * /srv/cronlogger remoteokio-sendmails "php -f /srv/http/remoteok.io/workers/sendMails.php"
*/4 * * * * /srv/cronlogger remoteokio-twitterfirerhose "php -f /srv/http/remoteok.io/workers/twitterfirerhose.php"

# NOMADLIST
@hourly phantomjs --ignore-ssl-errors=true --ssl-protocol=tlsv1 /srv/http/nomadlist.com/workers/getApiKeyFromSlack.js
@hourly /srv/cronlogger nomadlist-setcurrentlocation "php -f /srv/http/nomadlist.com/workers/setCurrentLocationForUsers.php"
@hourly /srv/cronlogger nomadlist-preparecitydata "php -f /srv/http/nomadlist.com/workers/prepareCityData.php"
@hourly /srv/cronlogger nomadlist-nomadslackuserdata "php -f /srv/http/nomadlist.com/workers/syncUserSlackData.php"
@hourly /srv/cronlogger nomadlist-postslacktrips "php -f /srv/http/nomadlist.com/workers/postTripsToSlack.php"
@hourly /srv/cronlogger nomadlist-airqualityyaq "php -f /srv/http/nomadlist.com/sources/air_quality_opensaq.php"
@hourly /srv/cronlogger nomadlist-scaleuserpics "php -f /srv/http/nomadlist.com/workers/scaleUserPics.php"
@hourly /srv/cronlogger nomadlist-tripstweet "php -f /srv/http/nomadlist.com/workers/postTripsToSocial.php"
@hourly /srv/cronlogger nomadlist-tripnotifications "php -f /srv/http/nomadlist.com/workers/emailTripNotifications.php"
@hourly /srv/cronlogger nomadlist-getfbmeetups "php -f /srv/http/nomadlist.com/workers/getMeetupsFromFacebook.php"
@hourly /srv/cronlogger nomadlist-exchangerates "php -f /srv/http/nomadlist.com/sources/exchangeRates.php"
@daily /srv/cronlogger nomadlist-joincache "php -f /srv/http/nomadlist.com/app/join.php -- 'refreshCache=true'"
@daily /srv/cronlogger nomadlist-synctwitterdata "php -f /srv/http/nomadlist.com/workers/syncUserTwitterData.php"
@daily /srv/cronlogger nomadlist-maketripsmaps "php -f /srv/http/nomadlist.com/workers/makeTripMaps.php"
```

There's even services that will monitor if your scheduled cron jobs actually run like **Cronitor**:



## Don't automate if it's not worth to automate it

As much as I believe robots are quite unlimited, how much effort you want to put in to make them advanced like humans is your decision based on how



much time you have. There's a famous trope of programmers automating everything but then programming the automation script takes longer than the time it saves. That's bad! You want to automate parts that take quite some time for a human, but don't take too much time to script.

## Where do humans fit in here?

For the things that take too much time to program automation for, it might be useful to just get a human to do it. In my case, there were a few things I really couldn't automate away. I've made most of my customer support automatic: a user can sign up, cancel their account, get a refund, change their subscription plan, change their credit card etc. all with a self-help dashboard. But sometimes you have users falling in between the ship and the land (that's a Dutch expression). For these edge cases you want a human on standby to solve their issue. This is how a typical white hipster human looks in 2017:



This human needs basic skills like communicating with a customer, going into your users database to fix things and if they really can't resolve it contact you to solve the underlying problem. If you have enough stuff happening, you can also pay them a monthly lump sum like \$2,500/m (that's what I do with my customer support) and just be done with it. I don't like to manage humans, so I like to work with people who are highly autonomous.

Another way is paying a human to be somewhat on standby based on when something happens and they have to deal with it and they log their hours and just invoice you through PayPal: that's what I do with my developer on standby. They get an alert when something doesn't work, and if they fix it first (before me), they invoice me for the hour worked (I pay \$50/hour and up).

You're laughing about how I talk like humans as just modules in your business. But anyone who portrays they're not is lying. If the goal of your business is revenue, then a human equals a robot. You put energy/money in it, and it gives work as output. Easy. (the social implications of humans being used as modules and becoming unemployed by automation will require basic income which I'm a big proponent of, but that's a story for another book).

Where do you find humans? I usually hire my friends as temporary contractors, but you can also try sites like **Upwork**.

## **What if the robots can't fix things themselves. How do the robots ask the humans for help?**

If you have humans on standby, how do you make sure the robots can figure out something is not right and call in human support troops? There's lots of ways. You can build your own robots that check if things are running properly. For example, a script that opens your website, checks for a certain keyword and if it's not there (which means the site is broken maybe?) emails you.

What if you want it to alert you by SMS? Twitter DM? Facebook? There's a fabulous app called **UptimeRobot.com** that does just that. Here's my dashboard there:



These are all little alerts that check something every minute or hour depending on importance. Like one alert visits the page on **Nomad List** for "safe cities in Amsterdam with good health care". I know that Amsterdam should probably be on there, so if Amsterdam is missing it means the filters are broken or something more severe is going on (like the page not loading).

## So this is like passive income, right? No!

By fully automating the parts that are worth to automate and getting contractors to do the stuff that are not worth to or can't automate, you now have built a machine that just runs and pays out money without you having to work on it. People would call this the magic "passive income" but I think that's a ridiculous term. You probably had to work 3 years on a crazy hardcore level to get to the point where you can automate and not work and make money. Then the question is how long will that last? Maybe another 3 years? So it's more like the first 3 years you just worked as hard as someone working 6 years normally. That's not passive income, that's just compressed income. Passive income is a myth. You can minimize work a lot, but until you sell your company, there will probably always be at least monthly situations where you need to step in.

If you've come this far with your side project, which is now a real business, it means you're near to having an autonomous organization. With Bitcoin-type blockchain technology like **Ethereum**, we can build so-called DAO's (or De-

centralized Autonomous Organizations). This is very new tech but the philosophical concept behind it is very interesting. You have companies that run virtually in the cloud, are owned by people that can trade stock in them and can get paid out dividends. Just like real publically traded stocks.

## The "bus test"

Going completely decentralized is hard and I don't completely know how to that yet. But I do know the way towards an almost fully autonomous organization. You'll need a mix of robots and humans to run it. If this works it means you pass the "bus test":

"A thought experiment which explores the impact of losing a person: If a particularly empowered individual in an organization is hit by a bus, will the organization suffer greatly? If yes, fail. If no, pass."

### — Urban Dictionary: "bus test"

Ask your self, would you pass the bus test? Transforming your product into an autonomous organization means you will.

Here's a rough set up to make your product into an autonomous organization:

- Automate repetitive work with robot scripts
- Hire a dev ops (or sys admin) person on a contract-basis (preferably hire 2 so you have a backup). They'll be the manager of the bots, kind of

- Set up alerts (like with Uptime Robot) so your devs are alerted if your product breaks
- Hire one part-time or full-time contractor who has access to your PayPal or bank and can manage and pay your devs (doesn't have to be technical necessarily). They're like the executive.
- Find one or two people that can be somewhat like "the board". They check if the executive runs your company as you would have wished.
- If you'd like, you can prepay balance to your hosting company (as I do with Linode) and domain name company (as I do with NameCheap), to prepay the costs for a few years. This will likely mean depositing \$1,000's with them, but it means you're sure keeping your product alive isn't dependant on your credit cards existing.

## Conclusion

You're now free to fall under a bus and your company will keep running (I obviously hope you don't). Even better, if you don't fall under a bus you now have a lot of free time to do other stuff! You've reached the holy grail of building a product: nearly passive income.

Nearly, because again, passive income doesn't really exist. There will always be small things. But luckily, you can get other people to manage most things for you. And in turn they can get bots to manage most things for them. Only in the most extreme cases, you're alerted for issues. That means you can focus on

spending more leisure time with your friends and family. Or making new projects and bankrolling them with the cashflow from this project.

The benefits of a clear mind are underestimated. Running a business is stressful. An entrepreneur's mind generally doesn't stop after work hours. New ideas, new solutions to challenges, they'll come up any time. That means they're not the nicest people to be with. But automation gives us a chance here to change that. Stepping away from the business when it's mature and having robots run it is one option. And it's an option I chose, and I think will be increasingly the norm in the future. Automation seems very, very obvious.

## **Resources mentioned**

- **Cron**
- **Uptime Robot**
- **Upwork**
- **Decentralized Autonomous Organizations**
- **Ethereum**

## **Your homework**

- From the business models mentioned in this chapter, make a list of which ones might work for your product.
- Pick one to start with and implement it. See if you're getting payments after a few days. Keep adjusting until the model works. If you're certain this model doesn't work for you product, ditch it and try a different one from here, repeat until you're generating revenue.