

Fabrice Dubois

Digital Product UX Design
Portfolio

Contents

1

City lighting on-site
care, fast and
sustainable

PHILIPS

©ignify

2

City lighting
telemangement,
optimized for
energy and
performance

PHILIPS

©ignify



Interlude

Simple principles
for hard problems

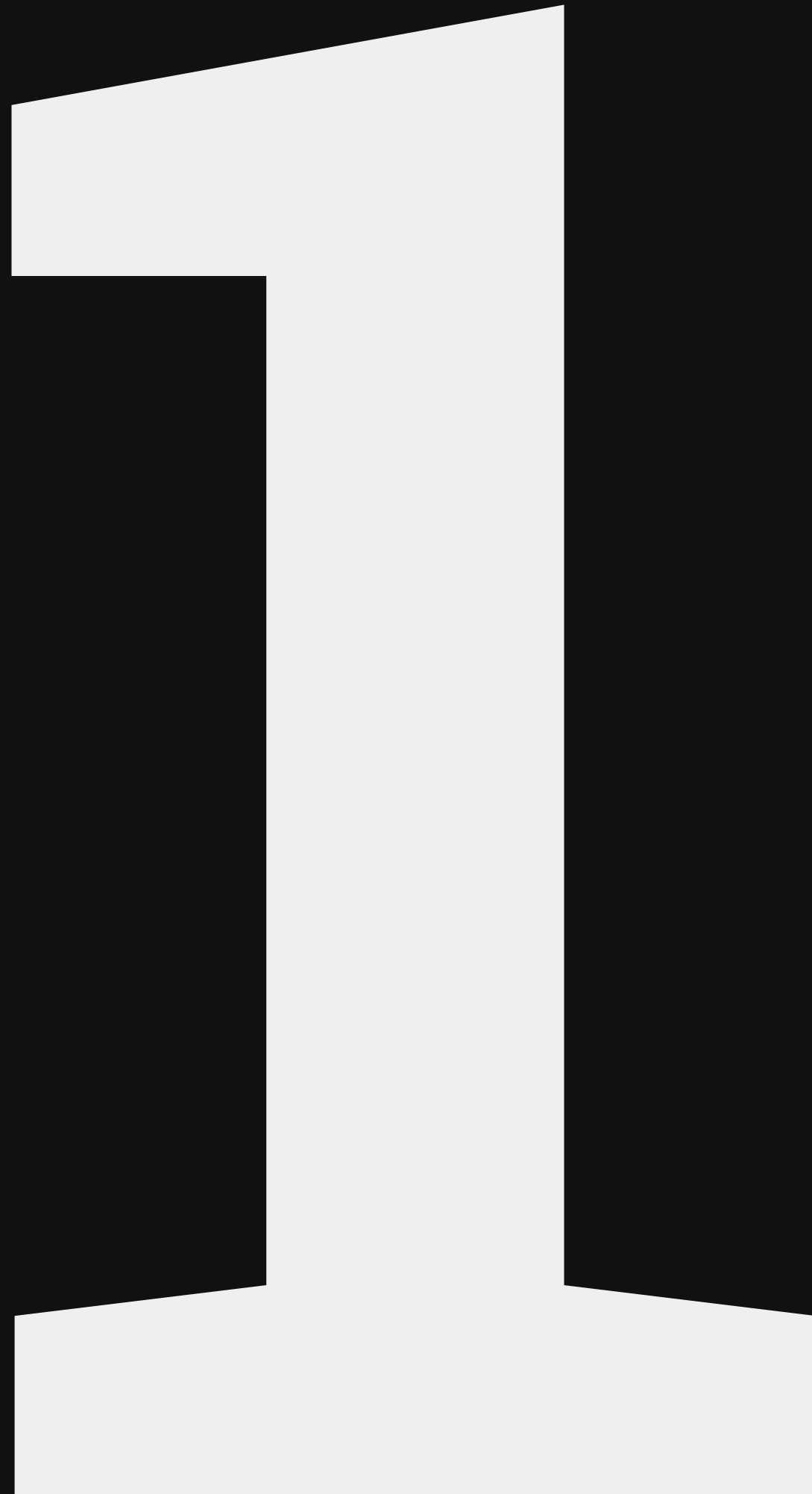
3

Airport staff UX,
boosted for
inspection and
collaboration

SITA



More projects that
helped me grow as
a designer, analyst
and manager



PHILIPS

signify

City lighting on-site
care, fast and
sustainable

Philips Lighting became Signify in 2018

Service Tag

An app that helps reliably install and selectively repair lights in the field. With the right data and tools at hand, engineers reduce waste and quickly get the light on.

Context

- 2018–2024, for Philips Lighting / Signify

My role

- Design lead
- Interaction design

Learnings

- Setting a strong way of working with remote teams
- Tech challenges (eg. NFC, GPS, etc.)
- Obstacles to app adoption
- Responsive design



Supporting every stage of the lifecycle

After identifying the luminaire by scanning the QR code on its pole, the user gets luminaire data, guidance and features to do tasks such as:

- Installing the product
- Registering the product
- Configuring the light behavior
- Troubleshooting
- Replacing a faulty part

Facilitating reuse of parts that work fine

LED luminaires are complex and made of many parts. Being able to identify, then repair or replace only the faulty component means less need to return the whole luminaire to factory.

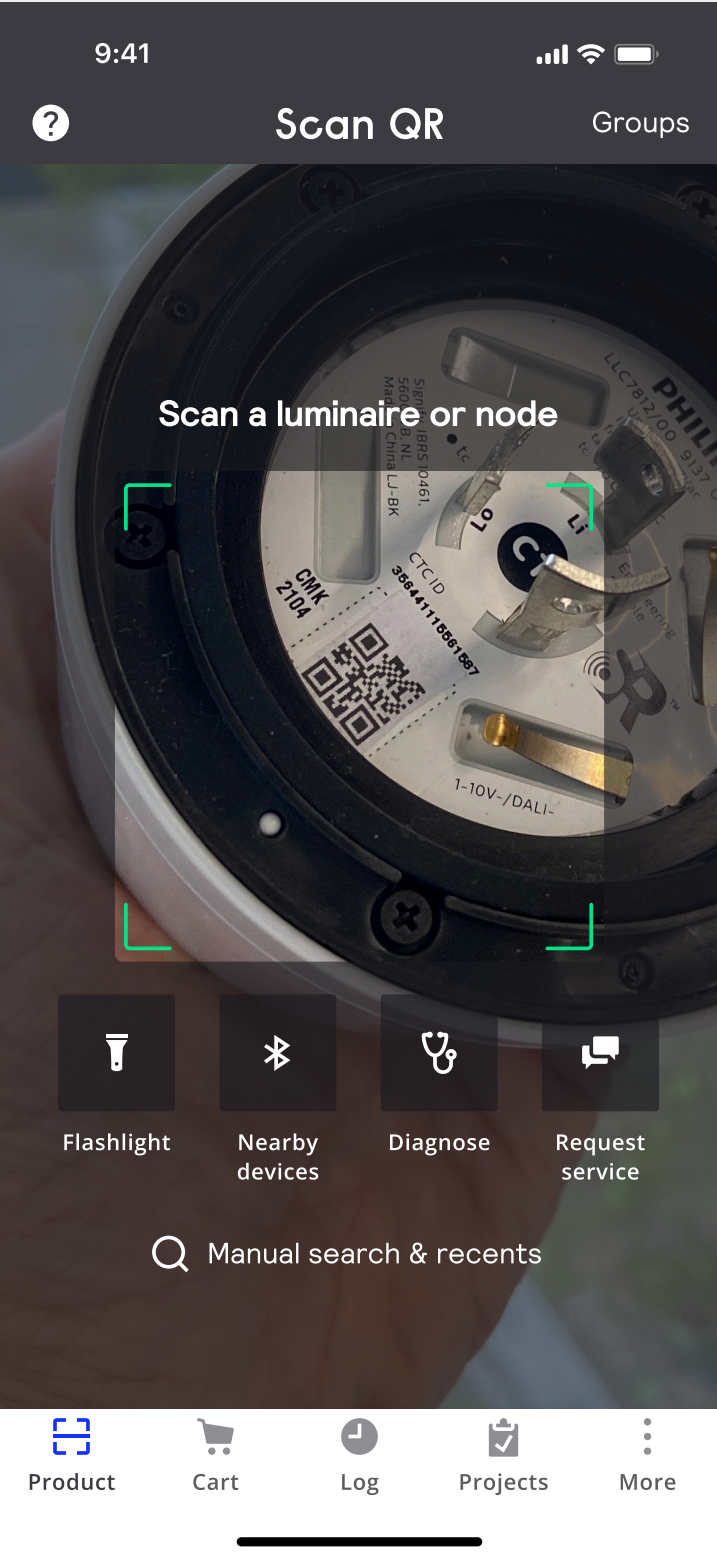
Service Tag supports a more sustainable, circular economy with better planning and reduced waste.

Servicing isolated streetlights

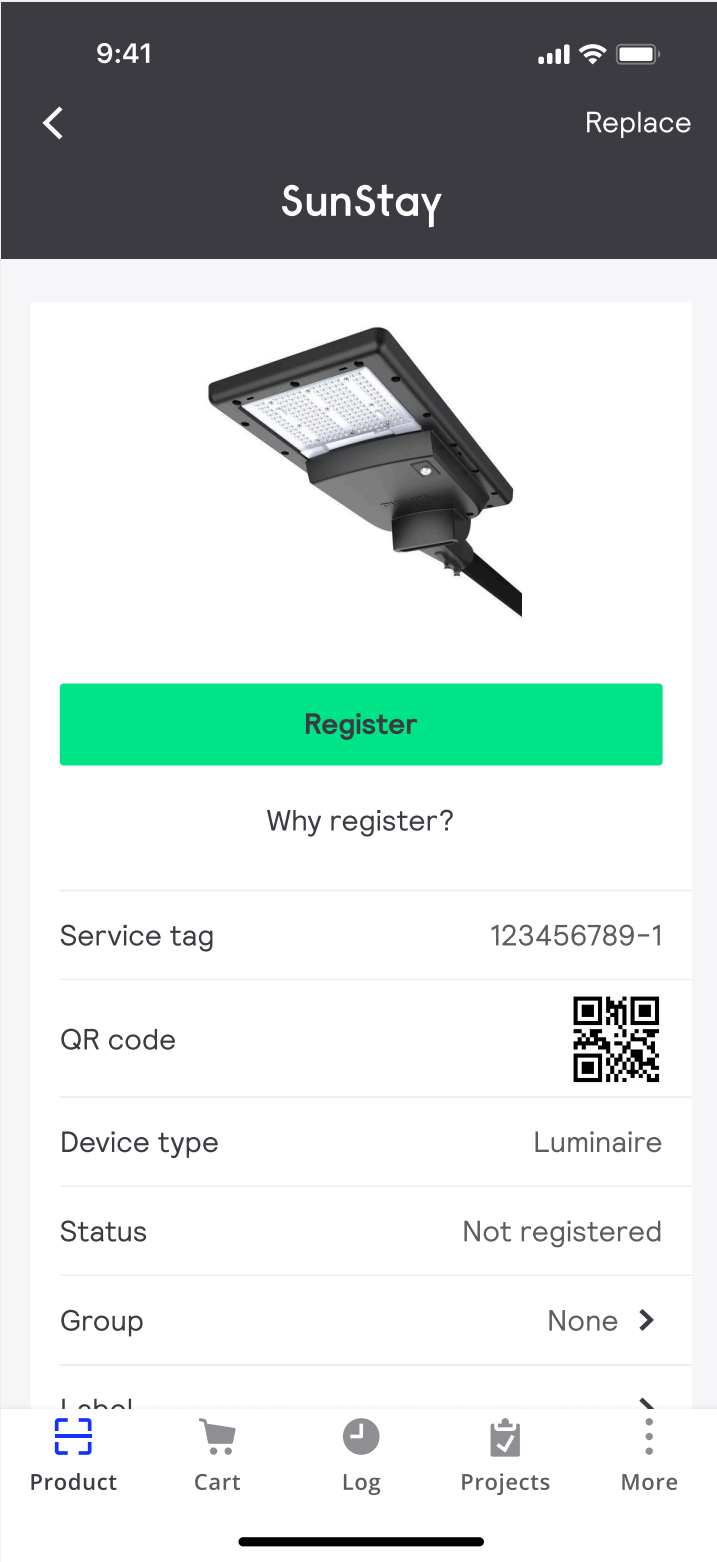
For modern, Internet-connected lights, certain actions can be done remotely. But the majority are still traditional devices without connectivity, and for those, people must go to the field.

On site, technicians want to act fast, typically to avoid disrupting road traffic too much.

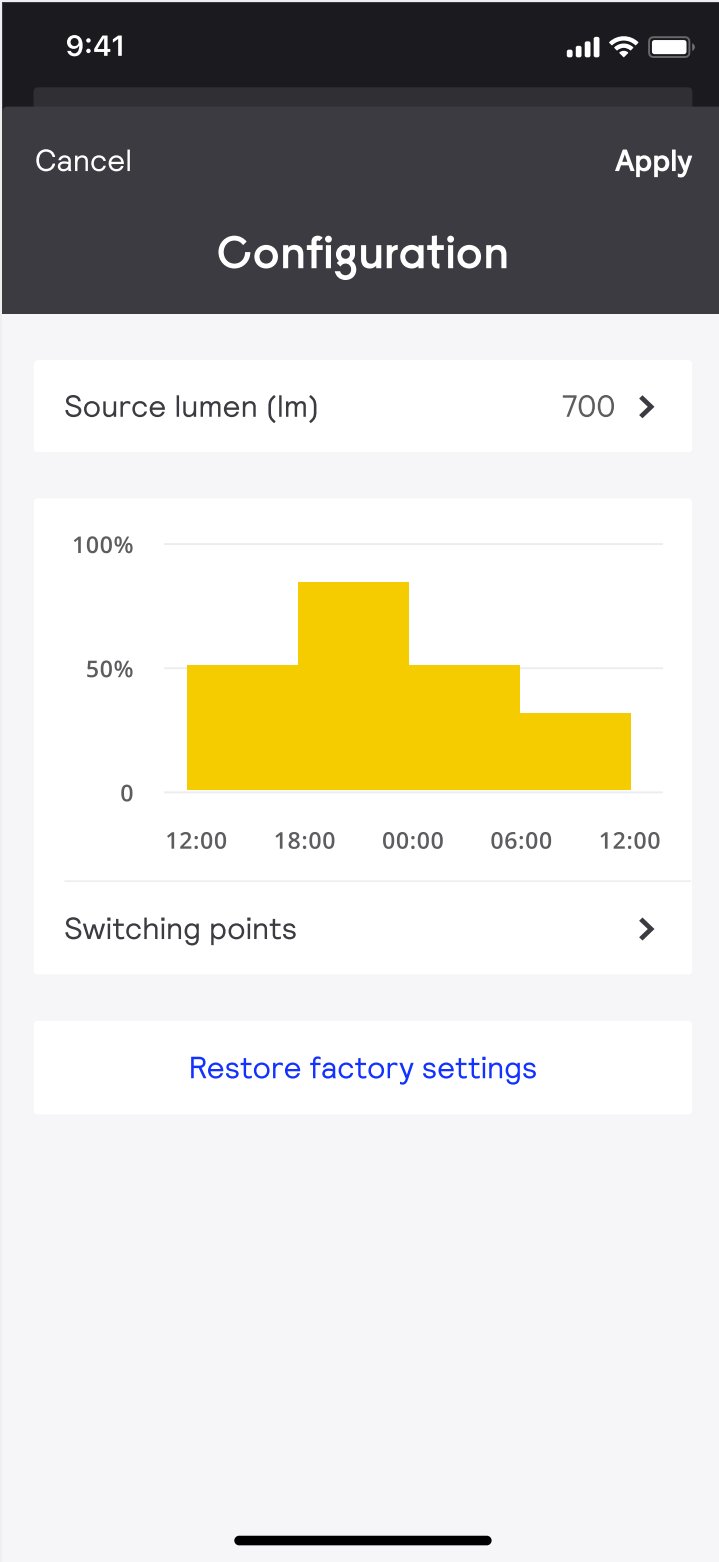
So it was key to provide straight forward ways to diagnose a fault, check instructions, reconfigure a LED driver, etc.



The QR scanner. *The start of most workflows, I made it easy to find (landing view) and added alternative identification methods in case users have trouble scanning.*



The device info view. *Shows up upon scanning. Simple list layout, with general items first, followed by more technical ones.*



The Configuration function. *The user can check the current dimming program (how bright the light must be throughout the night time) and override it with basic changes.*

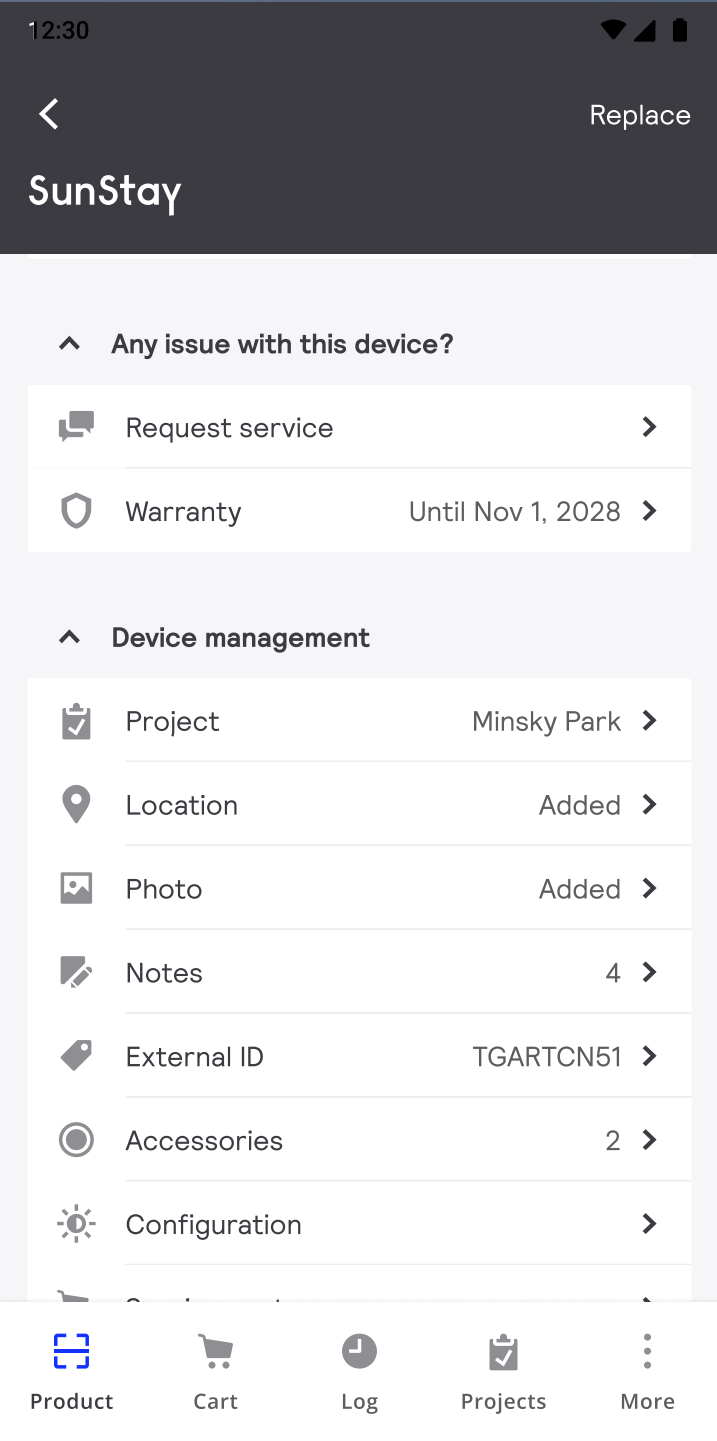
(iOS mockups)

Service Tag (2/5)

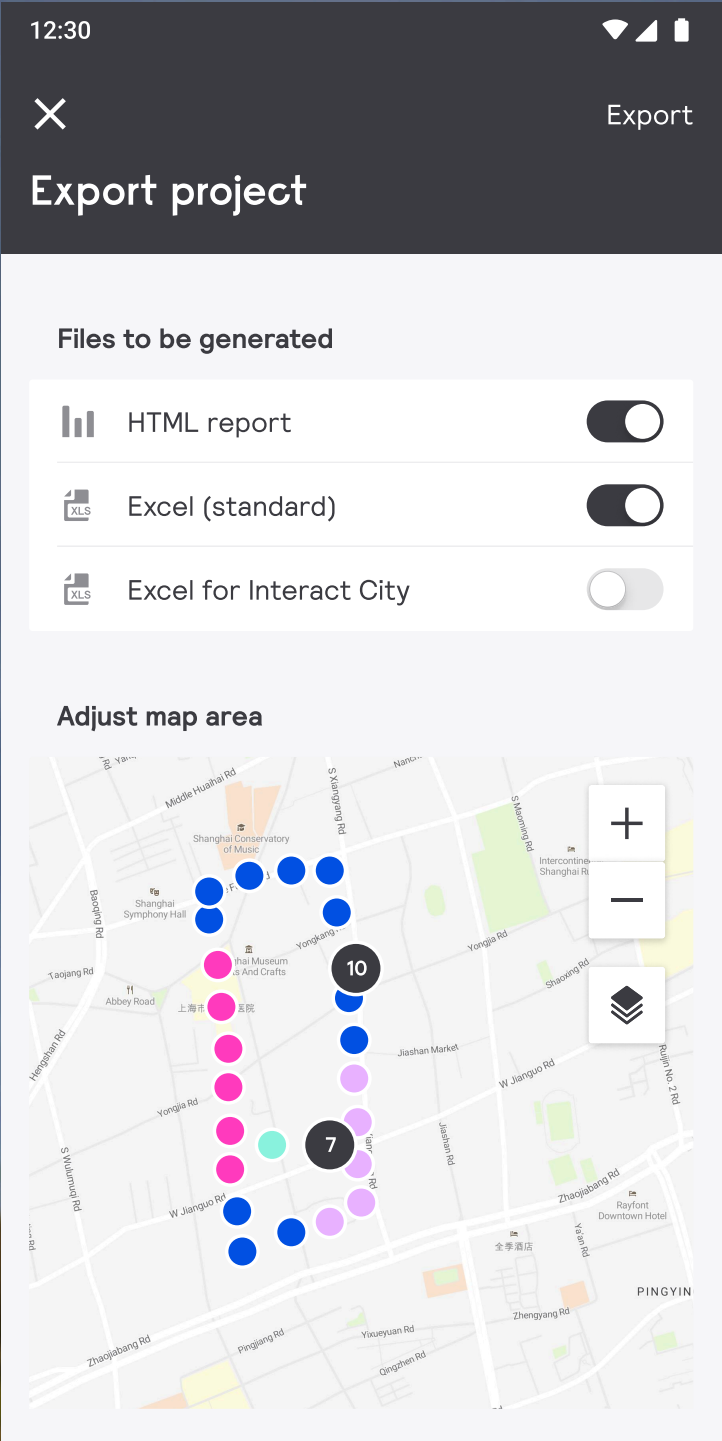
Design choices

Standard navigation and interaction principles make Service Tag intuitive and easy to extend with new features.

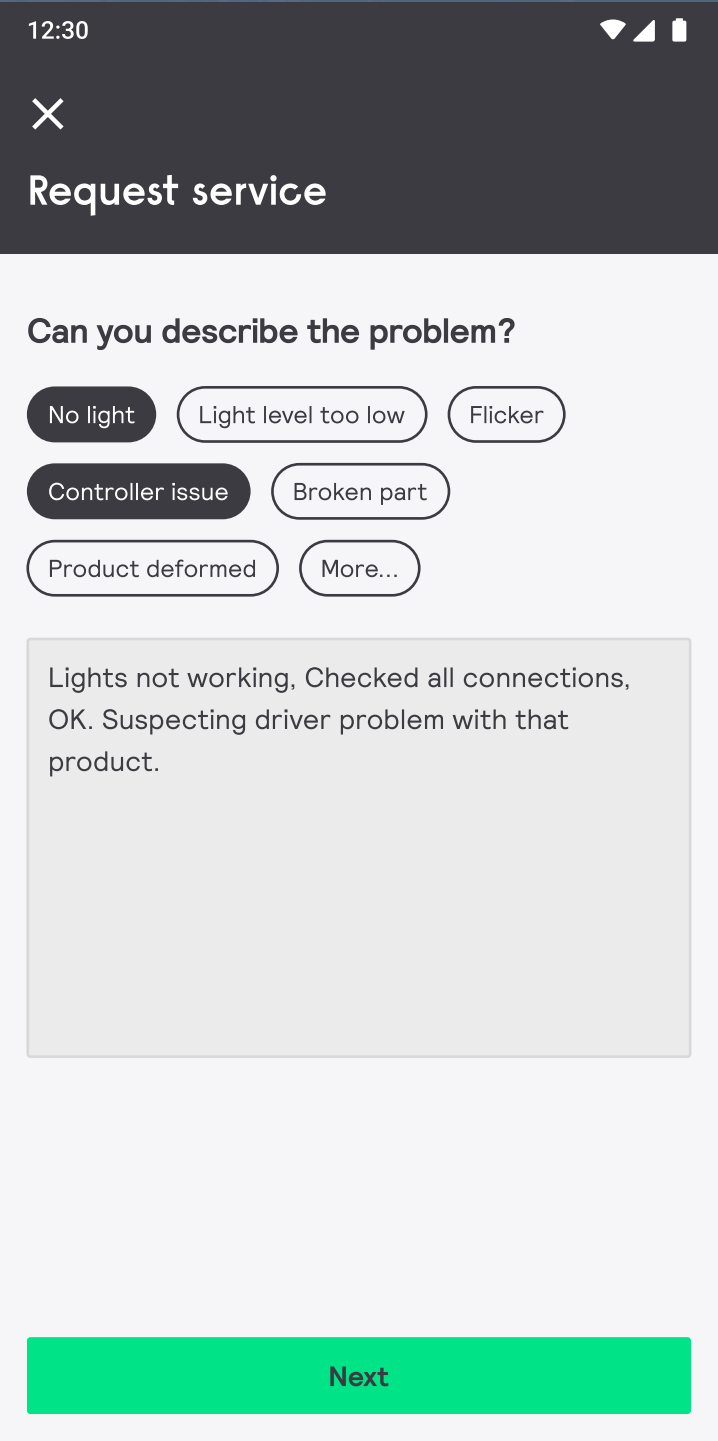
As we prioritized outdoor usage, I chose to limit the number of actions per view, paid attention to visual simplicity, and reused familiar terminology whenever possible.



Clean and logical grouping of sections.



Features well decomposed, no superfluous options. Use of modal view presentation for enhanced focus when relevant.



Limiting the amount of text input by suggesting one-tap frequent choices.

(Android mockups)

Usability issues: the GPS fix example



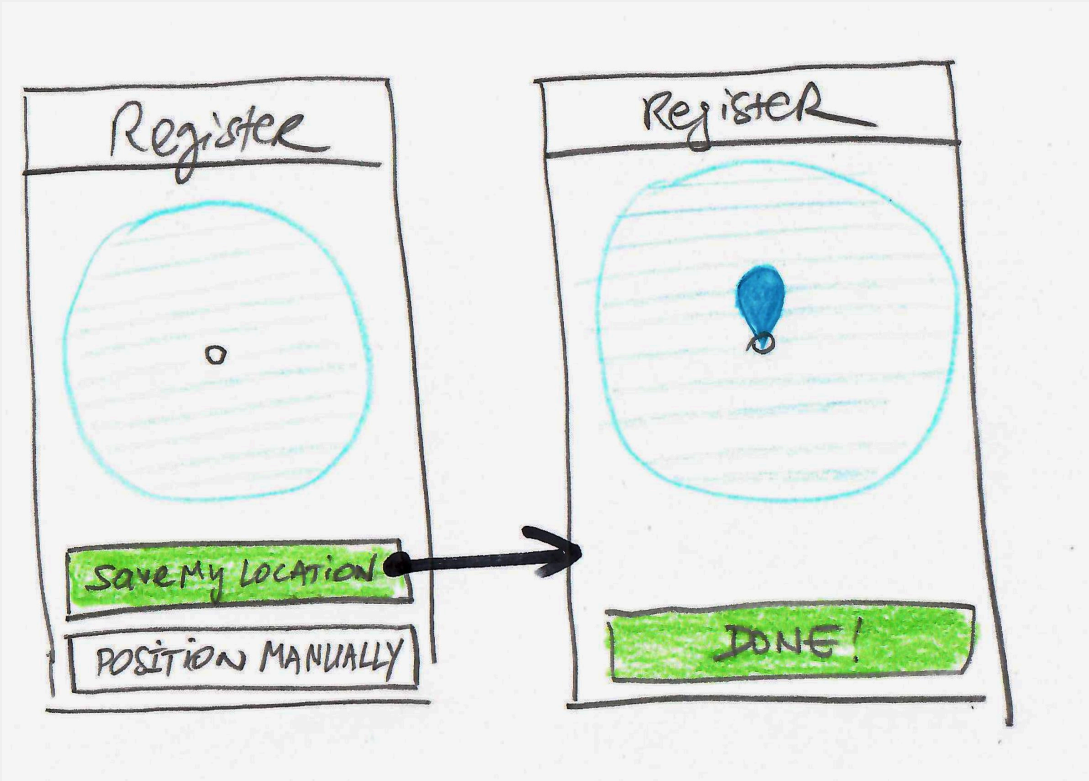
Key data in the system includes the light's GPS coordinates. These are normally set by installers via the mobile app. They do that during product registration, while standing by the pole, by confirming their current location on a map preview.

At first, some people complained that their saved locations were incorrect, sometimes significantly.

By observing how they were working we found they were saving too fast, without noticing the GPS positioning process needed more time to locate them well.

The revised UI improves data quality by stopping users from hitting Save until accuracy's deemed sufficient by the app. If that takes too long, it asks the user to position themselves manually on the map – which they usually do reliably by looking at the surroundings.

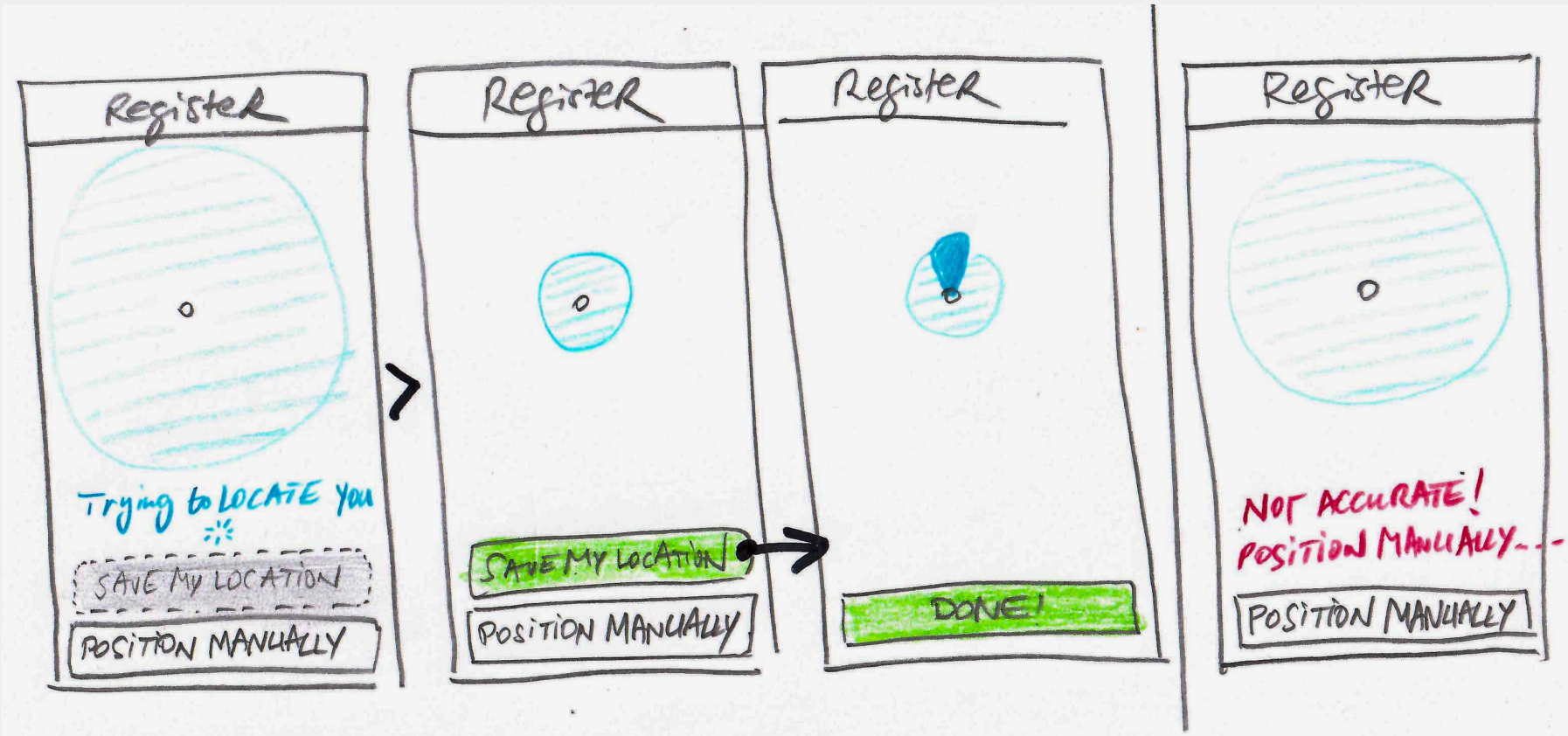
Original:



Accuracy low (big circle), yet Save is enabled

Wrong location saved!

Revised:



Accuracy low, Save is disabled

Accuracy high

Correct location saved

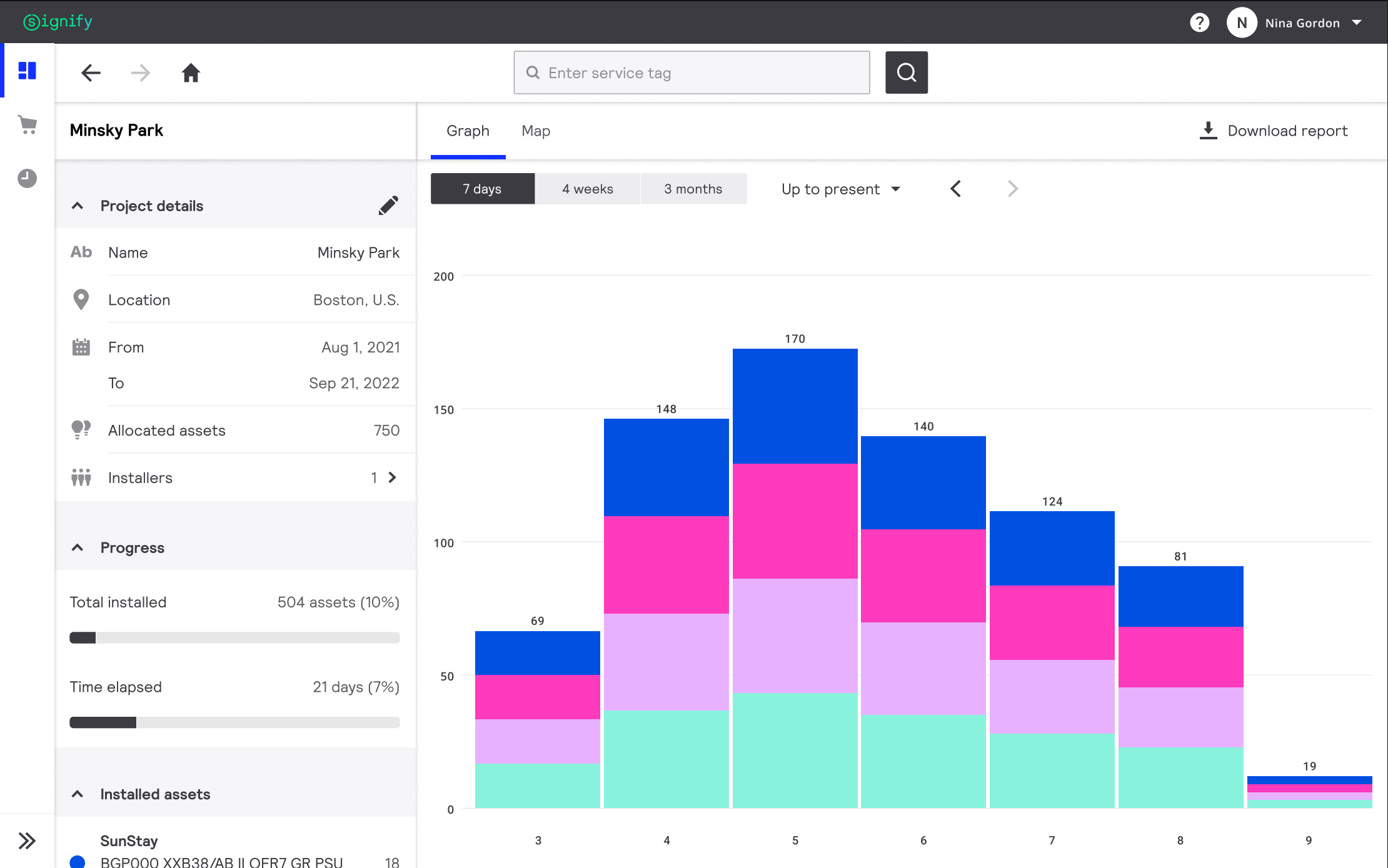
...or on timeout, user asked to place the pin manually

Mobile and web

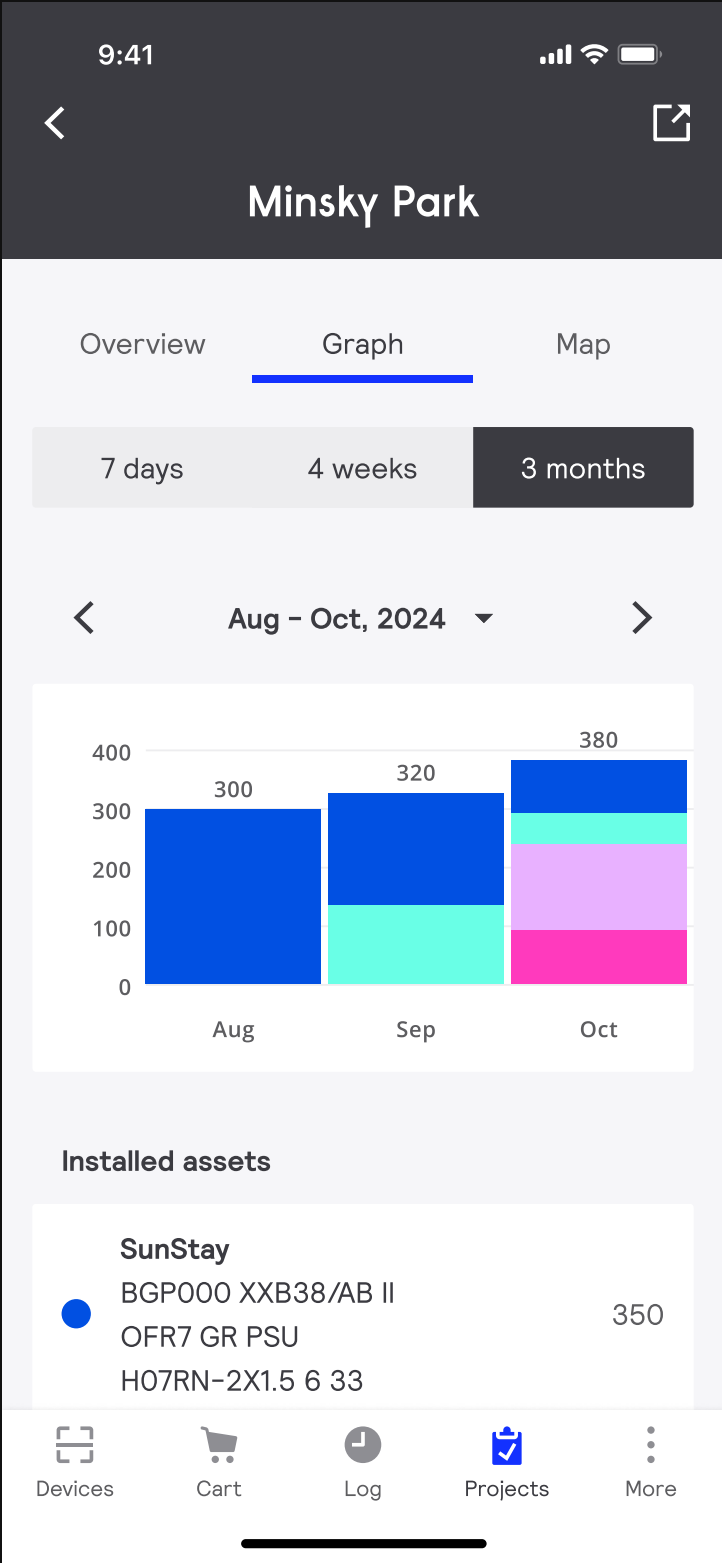
The Android and iOS apps have much in common, being both based on the same design system, Signify Photon. Branding was a must, but what mattered most to me was that the app felt at home on the user’s device.

An opportunity to grow the business drove the web adaptation

I later designed a web variant of the product, for a new category of users the PM discovered: office-based managers, they wanted a better integration with their desktop workflow, and being able to use a feature called *Installation monitoring* on large displays. This premium feature shows the realtime status of an installation project: what lights are being installed (registered) every day, how many, where, and whether delays are to be expected.



The app makes the most of each platform. On desktop it is more map (and chart) centric, has a flatter hierarchy and more advanced responsive layouts.



On mobile, it’s more decomposed and has features that are only used in the field – typically involving NFC or Bluetooth interactions with the hardware.

How we worked

Communication with PM and devs (based in India) was pragmatic and result-oriented. They would regularly brief me, voice their concerns, and I strived to deliver fitting solutions that I commented weekly. Over time we got to know each other, which led to efficient and realistic planning.

The PM valued the interactive prototypes I kept at their disposal for customer meetings, as they helped reach a shared understanding of the business needs.

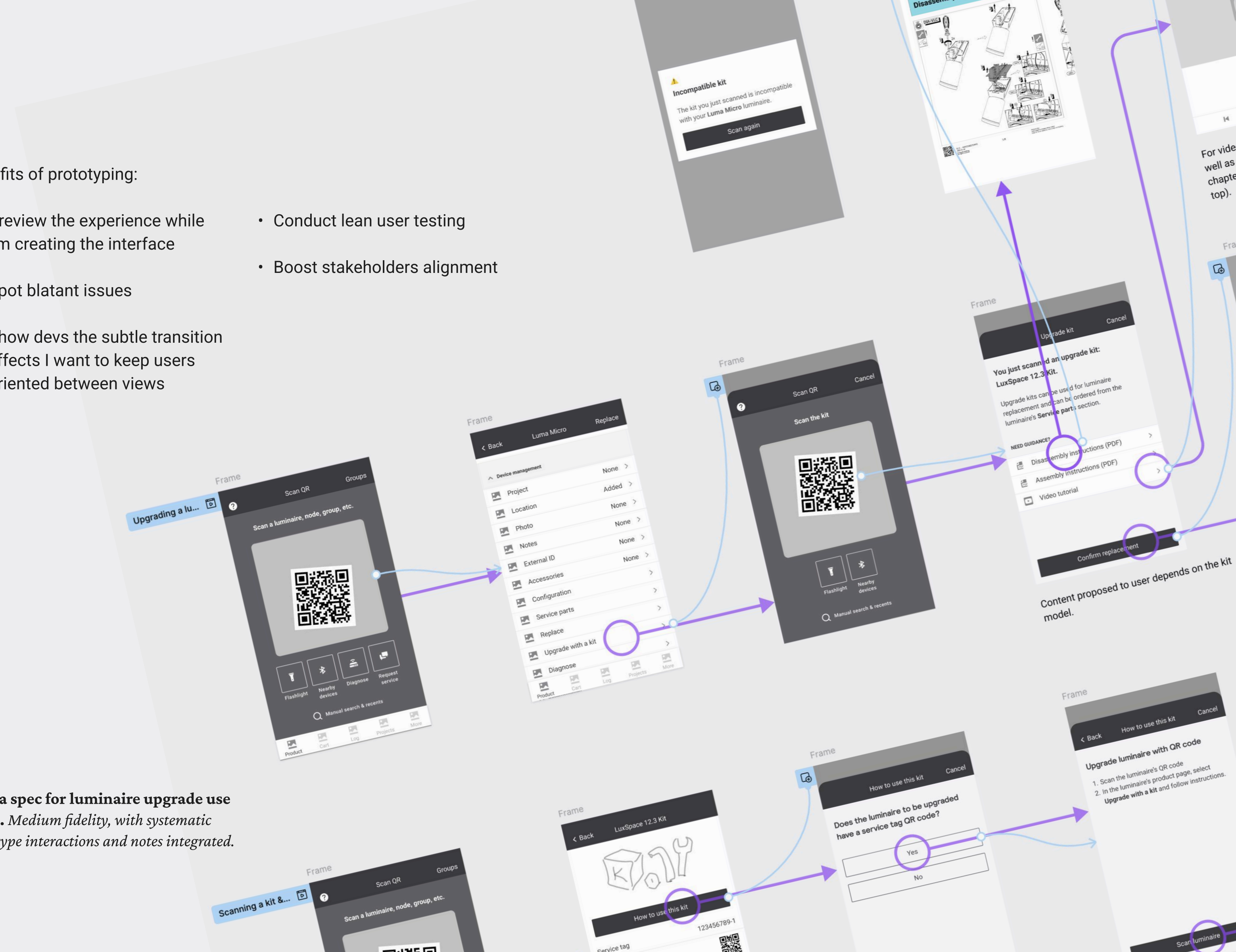
Specs and prototypes are complementary and best done at the same time

I tend to build prototypes as I go along, not as an afterthought. This became a habit when Figma made it expressive enough, and seamless to switch between canvas and preview.

Benefits of prototyping:

- Preview the experience while I'm creating the interface
- Conduct lean user testing
- Spot blatant issues
- Boost stakeholders alignment
- Show devs the subtle transition effects I want to keep users oriented between views

Figma spec for luminaire upgrade use cases. *Medium fidelity, with systematic prototype interactions and notes integrated.*





City lighting
telemangement,
optimized for energy
and performance

PHILIPS

@signify

Philips Lighting became Signify in 2018

Interact City

A web app for cities to remotely manage LED streetlights at scale. Better light with less energy, optimized operations and happier citizens.

Context

- 2018–2024, for Philips Lighting / Signify

My role

- Design lead
- Interaction design
- Mentoring of new contributors

Learnings

- UX research
- Design for web & SaaS
- Information architecture
- Long-lived platforms
- Stakeholder & priority management
- Team planning



LED telemanagement benefits

Source:
Design for sustainable smart cities
Signify 2024

Energy savings

Up to

80%

Compared to conventional lighting.
Achieved by using LED, dimming control, and remote management.

Field trip savings

Up to

50%

Thanks to preventive maintenance, accurate asset management, and timely fault detection.

Crime reduction

Up to

20%

Because better light management means safer streets. Also, road accidents reduced by up to 30%.



The application

A modern interface to unify legacy and new systems

Over the years, Signify has offered various city lighting and remote management solutions, each with their own hardware, software, pros and cons.

With Interact City, our goal was to create a unified and modern web platform that integrates these systems together and gracefully.

This required me to understand each system well, recognizing common elements while preserving their unique features.

Aligning things could be challenging. Changing workflows too much would confuse people used to the older products.

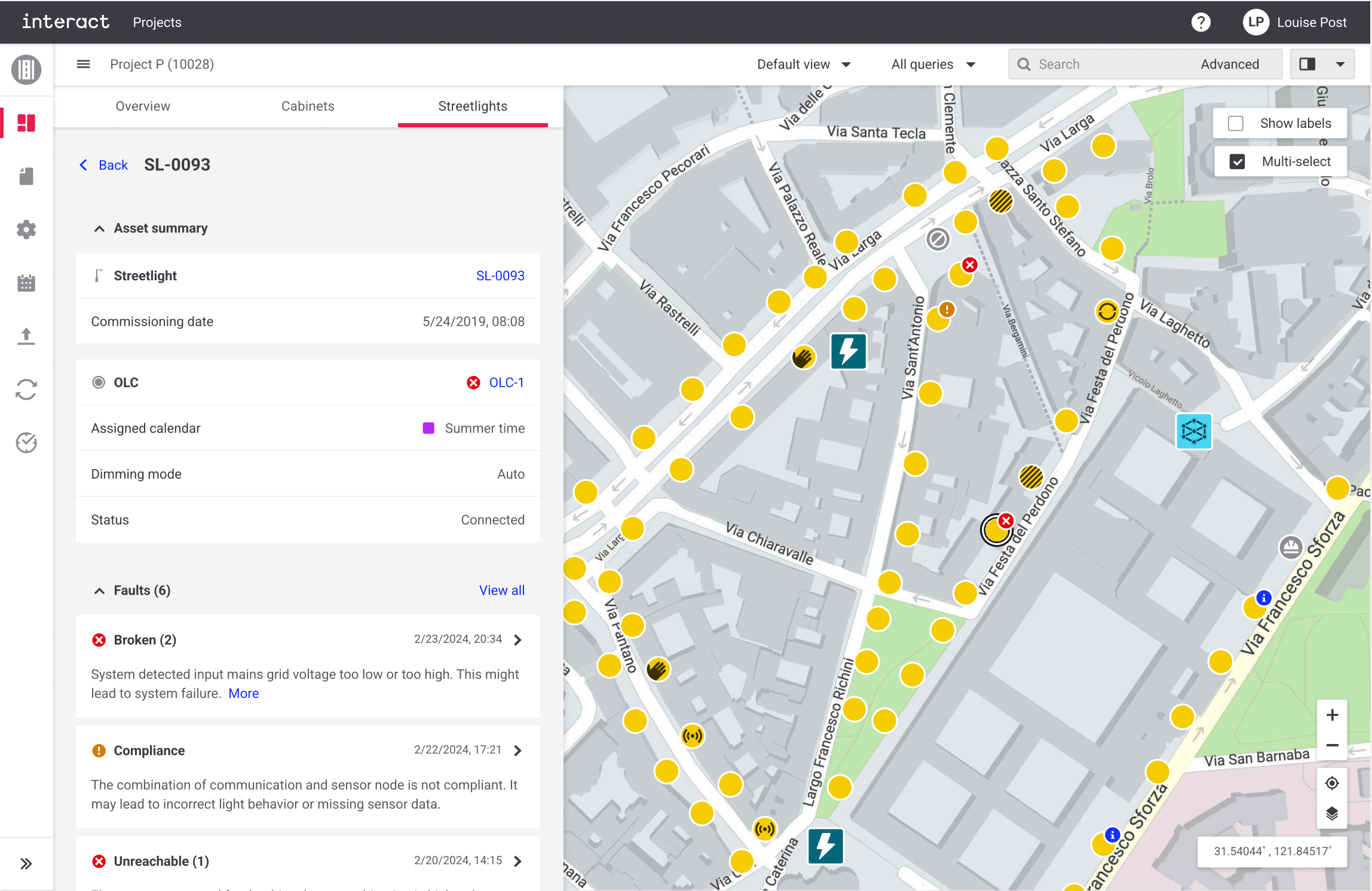
Besides, we gradually added features and support for new technologies, such as sensors, solar-powered lights, or simplified commissioning.

The main challenge was to keep the design adaptable for future developments, but the overall complexity manageable.

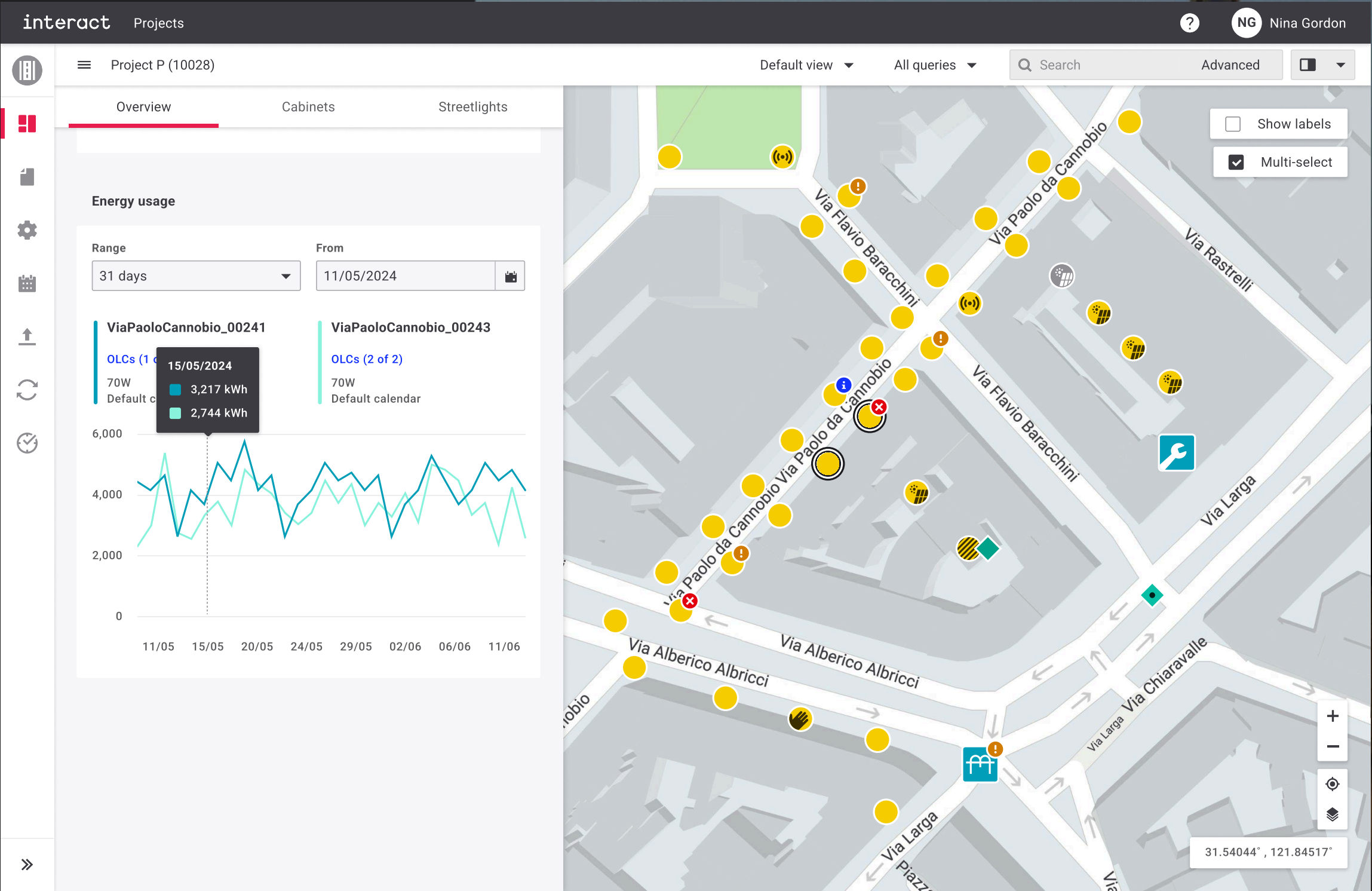
One can’t evolve a product they no longer understand.

Keeping the design adaptable and complexity under control

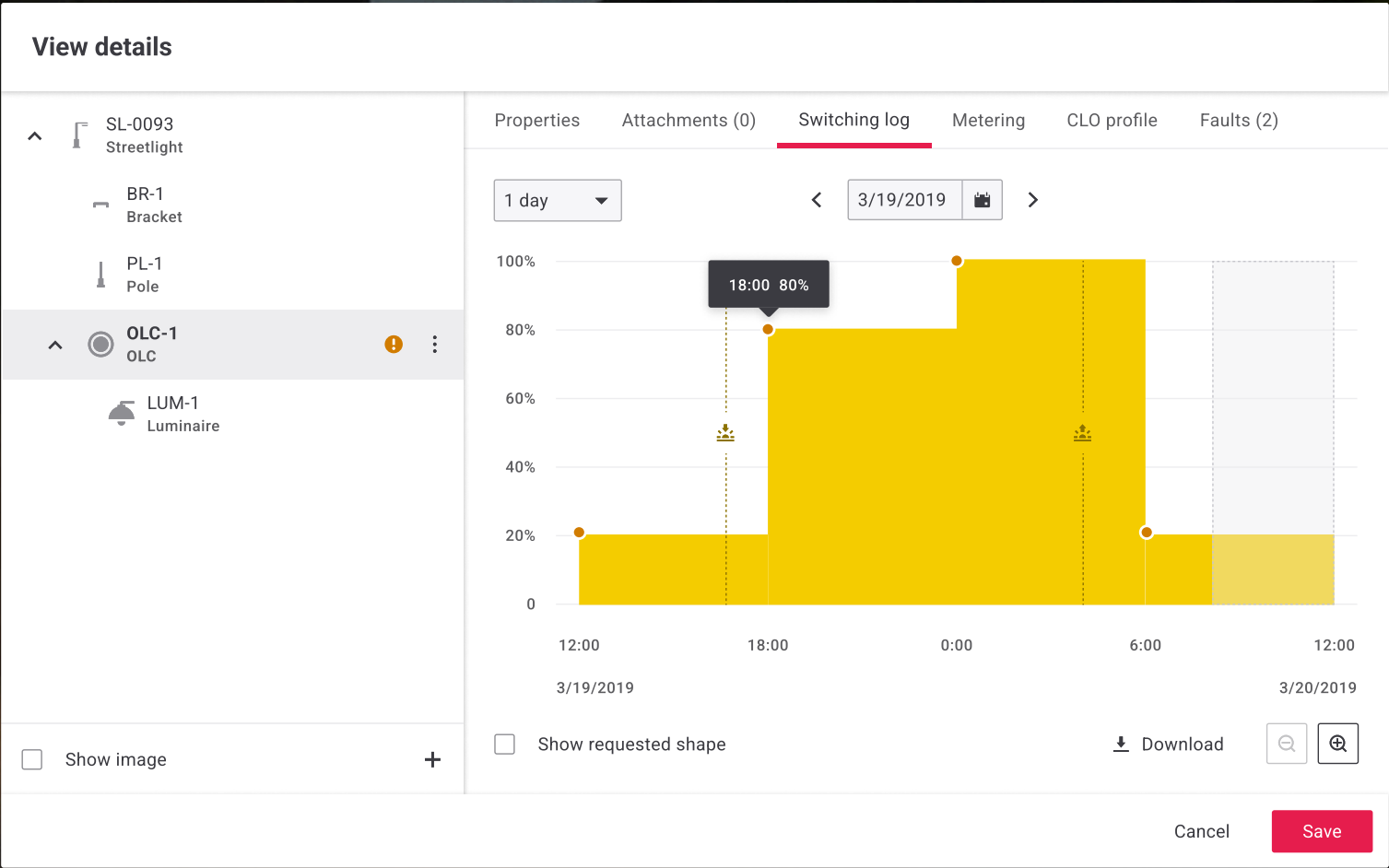
Style wise, we learned how to apply our design system consistently and, when relevant, deviate thoughtfully.



The main UI. Users expect to see a lot of data on a map, but it must remain intuitive. Showing a growing number of objects, in different states and of different types, required a rigorous icon system that prioritizes information.



The quick comparison tool. When they troubleshoot a light, engineers like to start by comparing its energy usage with that of a neighboring light. A call with a user made me realize this action’s UX could be improved, and all it requires now is 2 clicks on the map.



The View details UI. Initially designed to inspect streetlight properties, this evolved as a general tool that proved helpful for many workflows. It combines tree and tab views to organize function and information efficiently.

Approaching deeper changes

A case study

Example of process I followed to address a significant redesign.

At some point in the project I saw a need to rethink part of the UI. More and more issues reported by users were hard to tackle individually, as I knew they were symptoms of deeper limitations.

I argued we had an architecture problem that demanded to be addressed globally rather than with local patches.

Ideas were in the air but I didn't see implications for users or on the rest of the product. One day I decided it was time to **go to the bottom of the problem and come up with a strategy.**

Understanding →

The multiple issues seemed caused by the same flaws:

Unclear navigation logic.
Hard for users to build a mental model and predict UI reactions.

No uniform way to handle objects.
Mostly designed for streetlights and cabinets, the UI is too rigid to adapt to new types of objects.

Began a long journey of juggling:

- Deconstructing the current design (what ingredients, how they're connected, fundamental or artificial, etc.)
- Securing alone time outside business as usual
- First thoughts on user observation, how to ace it

Interviewing →

6 or 7 user interviews, in open format as I wanted a discovery approach to:

- Understand how people think
- See what matters to them
- See where they struggle
- List down problems to solve

Users seemed to work like this:

1. View a certain set of objects (eg. *Lights on Main Street*)
2. Apply actions
3. Repeat with a new view, or just go back

In short they needed to quickly jump between arbitrary views. *But:*

- Existing navigation was too hierarchical and conflicted with the expectation to move freely
- I had no proper definition of the concept of a 'view'

Shaping →

The hardest part. Turning intuitions into key ideas, and ensuring they work, *especially when put together:*

- What do I introduce or change?
- What do I lose, how do I mitigate?
- What new questions arise?

Key idea examples:

Users should access their favorite views as easily as bookmarks in a web browser.

Navigation should be more historical, less burdened by 'spatial' controls like tabs or breadcrumbs.

Data visualization features interfere with navigation and should be relocated.

Etc.

Testing, Planning, Detailed design

With focused prototypes showing off each key idea, we met again with our interviewees. Integrating their feedback took a few rounds but the direction seemed solid.

Time to share more! Demos to local PM and dev lead, then to the whole project to get support.

Once we had a clear target and buy-in, detailed design could start. I mapped out dependencies to see what we had to do first, which helped us proceed incrementally and with confidence.

The big idea was easy to split into small, digestible demos: a good sign!

Interact City (5/5)

How we worked

My main project contacts were PMs, software architects and customer-facing engineers.

We were up to 3 contributing designers, plus a technical writer, while system experts (internal users) informed our work with frequent inputs and feedback.

We used the SAFe agile method. Every quarter I scheduled design work for my team so as to stay ahead of R&D and deliver solid solutions. Over the years, I produced most of the interaction design and associated documentation – but my team was very supportive and the early version of the UI came from an agency.

I encouraged the team to join global meetings such as the Change Control Board (CCB), to connect with project members in diverse roles. Good for socializing, motivation, and showing that designers don't mind getting their hands dirty when needed.

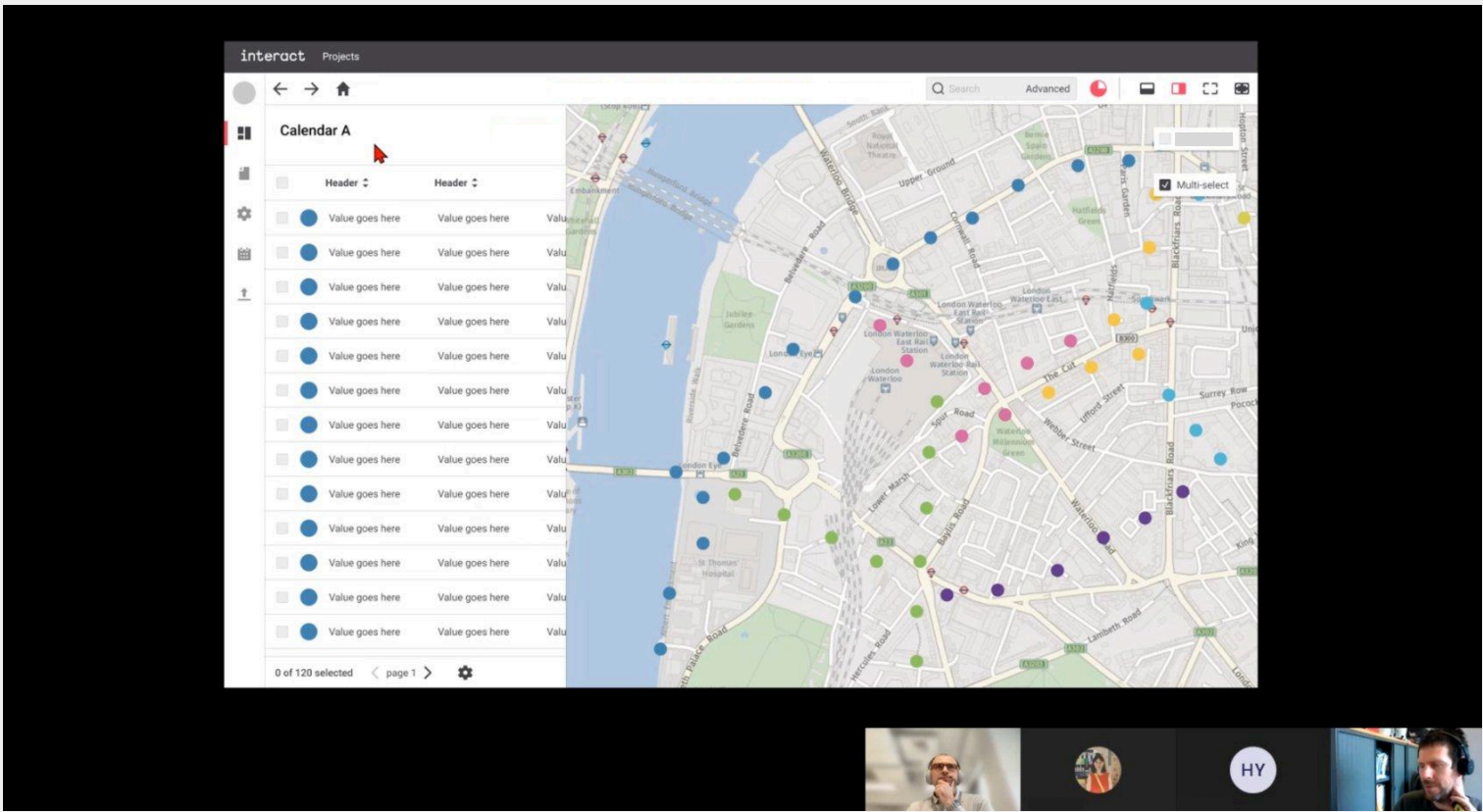


Meeting in person. *Here on a visit to system experts in Bangalore. These trips were very useful to establish trust and team spirit. Hearing diverse voices allowed me to grasp problems faster and generate fresh ideas.*

To save everyone's time, prep work is best, but UX research doesn't have to be complicated. Our global UX lead once said to me: "Stay pragmatic. It's only about getting answers, in the end."

Watching people use the product reveals more than their mere words ever could

Usability testing. *A user from the Netherlands plays with a potential solution to one of his pain points. If people agree, I record the session and invite other designers. This is to catch details I could miss during the call, and to better focus on the respondent without having to take notes live.*



A thick, bright pink diagonal line runs from the bottom-left towards the top-right of the slide, positioned to the left of the text.

Interlude

Simple principles for
hard problems

Orthogonality

Orthogonality is about ensuring that the parts of a system don't depend on one another without a good reason.

In a well designed system, the parts tend to be orthogonal to each other. It means you can manipulate one part without worrying about the other parts, as each part is well-separated and understandable on its own.

In tools like Photoshop or Figma, the layer system is very orthogonal, which largely explains its power and universal adoption: the user can reorder layers regardless of their visibility attribute, and toggle their visibility without having to think about how they're ordered. Manipulating a feature does not impact the other.

The Mikado game, pictured here, is a metaphor for very poor orthogonality: grabbing any stick will very likely move sticks nearby. Even *looking* at a particular stick – finding where it starts and where it ends – is tricky. Products built like that are mentally exhausting, difficult to explain, painful to work with and work on.

Designers sometimes get briefs that look like a bowl of sticky noodles, a bit like the Mikado. Their first job is to disentangle the mess, clarify intentional dependencies, cut accidental ones, and if necessary, restate definitions.

One thing I noticed, for a healthy start: long list of short requirements beats short list of long ones.

“Don't link what's independent.”
– Fred Brooks



Emergence

Emergence is when a system shows interesting new characteristics and behaviors that weren't designed, but that just *happen* at runtime. Classic example: surprising shapes created by flocks of birds. Birds are not programmed to do shapes. Shapes simply result from a few basic rules the birds follow.

When it's hard to tell what the users of a system will do in the future, when too many scenarios seem relevant, designers may consider focusing on the seeds rather than the fruits; on the DNA rather than the flesh. In other words, on the basic rules, so as to grant users space for creativity and adaptation.

I had eureka moments related to emergence before, though I didn't understand it right away. Once, I was trying to prototype a telecom session involving multiple users; another time, on location sharing for teams with unclear policy requirements.

Every time, the emergence-based solution felt simpler, and at the same time infinitely flexible.

“One of the founding principles of agility, and the closest one to pure magic.”
– Andrew Hunt



Informed simplicity

This is CCTV headquarters in Beijing.
A gigantic engineering challenge, due
to its impossible shape.

For a long time I was scratching my
head, not sure how to think of it.
Donut beaten badly, origami-folded?
Escaped from an MC Escher picture?

Now consider this: a cube minus two
other cubes. Carved out at diagonally
opposite corners. Two large notches
leaving a hole where they intersect.

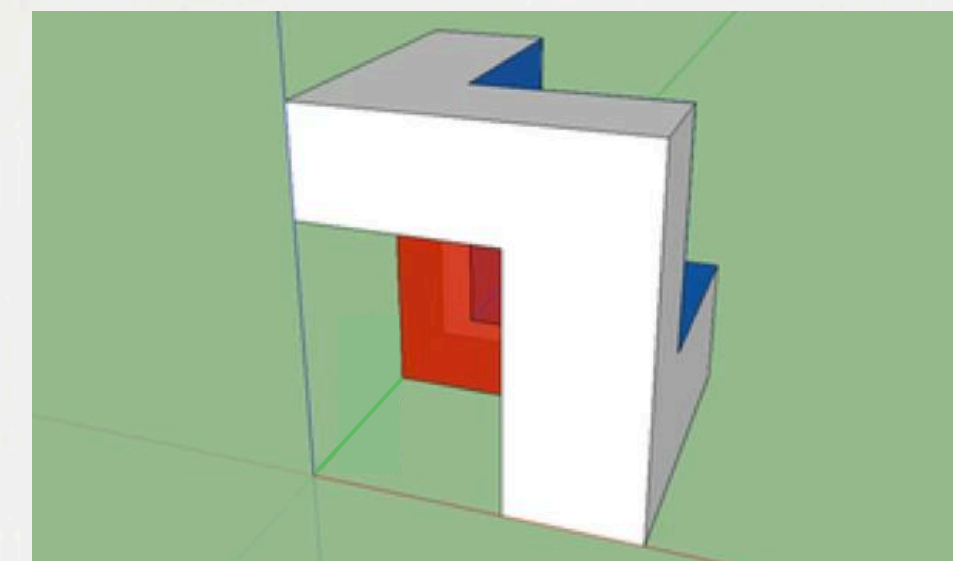
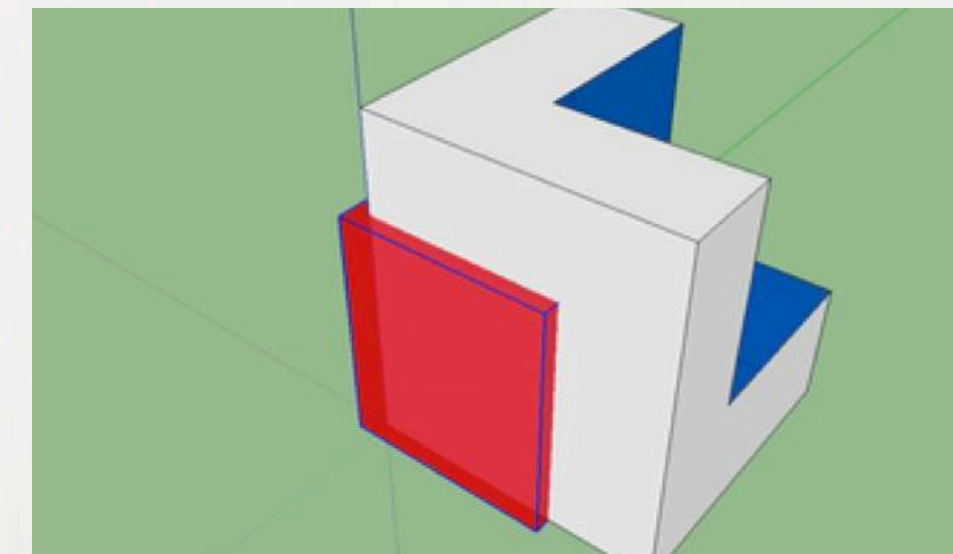
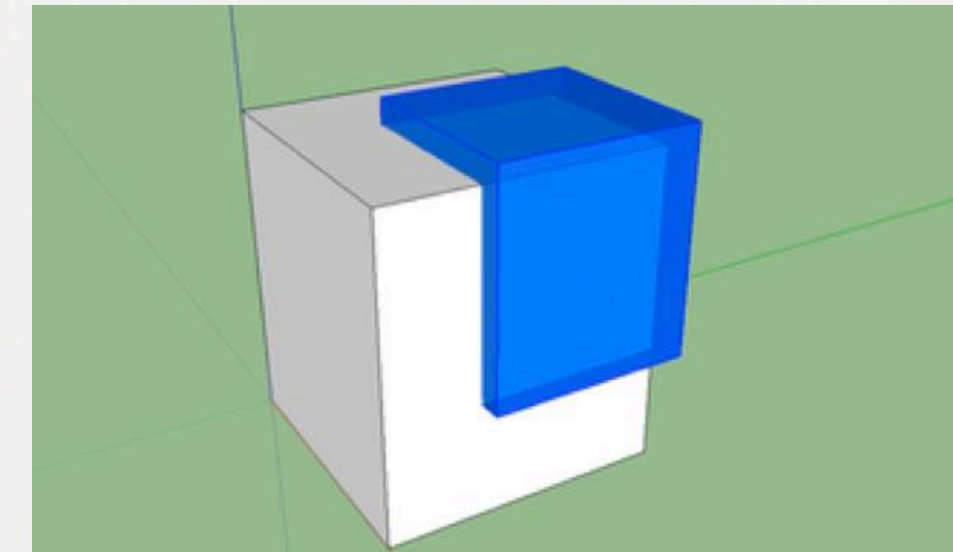
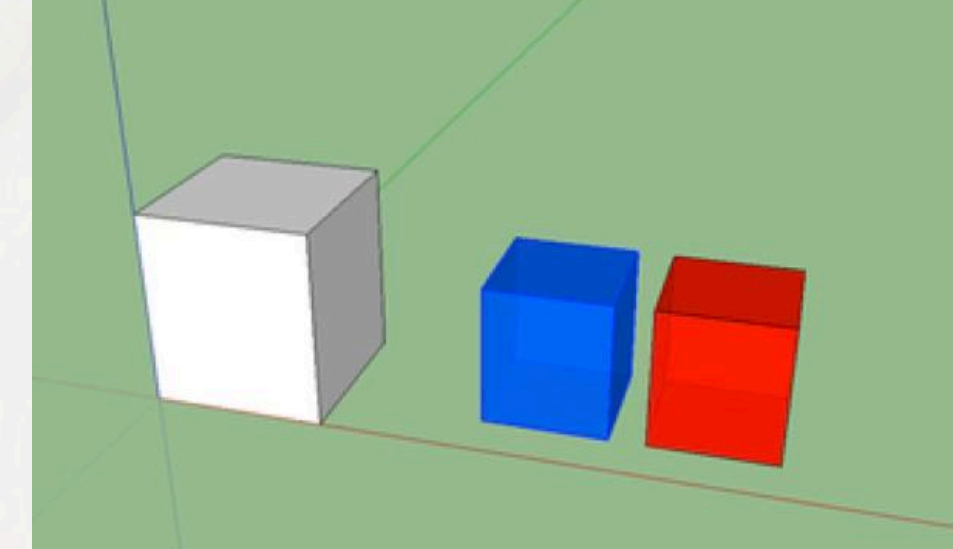
Systems seem tricky until you look
hard enough and discover ways to
'rephrase' them in simpler terms.

*“An enlightened view of reality,
founded upon an ability to discern
or create clarifying patterns within
complex mixtures.”*

– Matthew Frederick



Photo & 3D: Fabrice Dubois



3



Airport staff UX,
boosted for inspection
and collaboration

SITA provides telecom services to the air transport industry



1. Inspection

A tablet app to simplify how airport staff access and fill in all kinds of forms as part of daily inspection and reporting procedures.

Context

2015–2016, for SITA

My role

- Design lead
- Interaction design
- Visual design

Learnings

- Challenges of airport workforce and how digital tools help
- Focusing on high value use cases
- Design for iPad

Overview

Airport staff must routinely fill in various forms, usually on paper.

In this first project I delivered a framework for digitized forms to improve people’s lives thanks to:

- Less paper to carry (and lose)
- Always up-to-date templates
- Easy data input and validation
- Access to personal history

I worked on two aspects:

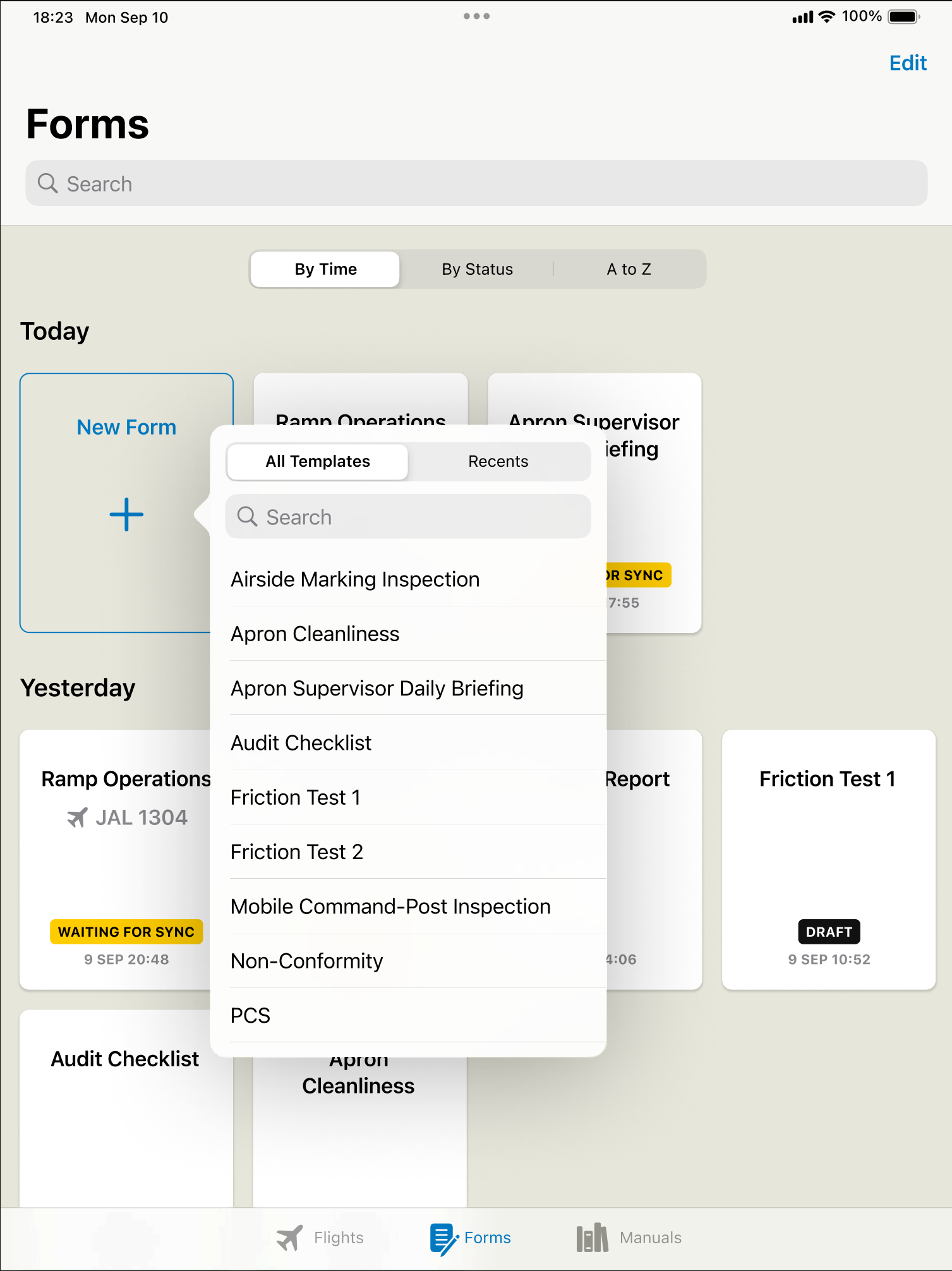
- **A form library**, where users browse their recently submitted forms and start filling in new ones based on templates.
- **Design of specific forms.** For simple routines, we decided to reuse existing HTML versions of the forms, so I could focus effort on more complex procedures like cargo loading, passenger list, or ramp operations (see next page).

The form library adopts the familiar document collection model. A decision driven by the fact that users sometimes need to go back to a particular form, check a form’s status, and more.

Hiding complexity if it’s irrelevant to the end user

Despite a technical constraint that prevented auto-sync of HTML forms to the cloud, I ensured the library presented all forms in the same way, be they auto-synced and manually submitted. An effort to downplay implementation constraints users don’t need to be aware of.

The form library. *Makes it easy for the user to find forms they worked on in the past, or a specific template to create a new form.*



Forms for complex procedures

For some forms, investing on a custom UI really unlocked game-changing user experience.

The **Ramp operations form**, for instance, assisted an operator in supervising the actions required on an airplane during turnaround (the period between the aircraft’s arrival and its next departure).

With a visualization of progress, the user directs their team to best meet deadlines and work in the right order. For reporting purposes, they also capture the start and completion times of each action.

The longer the airplane stays parked, the more the airport charges the airline, so time efficiency is crucial.

Bad timing cost the airline a lot

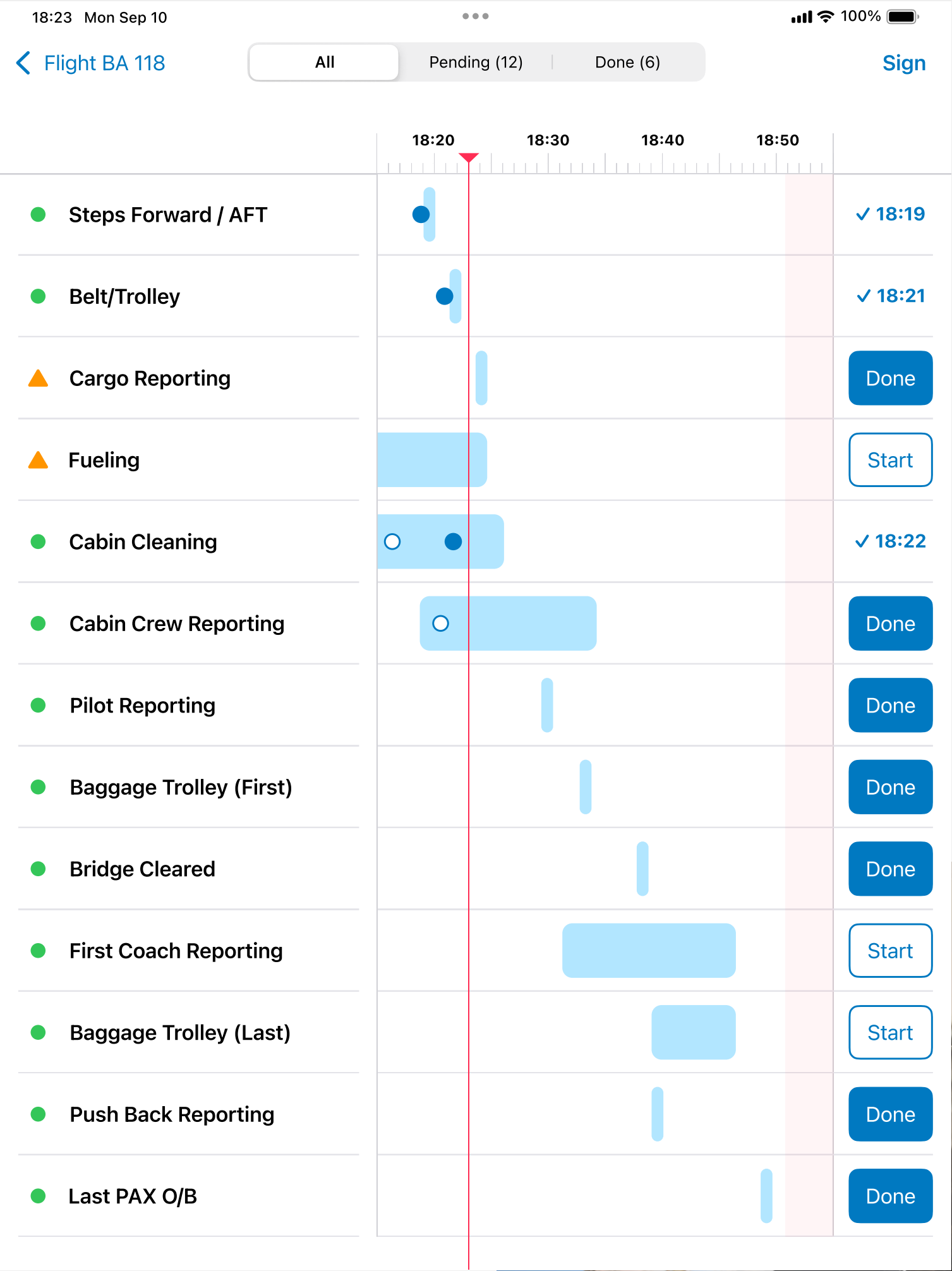
Inspired by multi-track music sequencers, the UI shows how long operations are expected to take (light blue bars), warns if some become urgent or run late, and can be filtered by action status.

The operator simply records times by tapping a button when an action starts, and re-tapping when it’s done. The forgiving UI supports editing after the initial capture.

A reusable design

At some point I saw we could generalize and make the design work for any procedure that involved a similar *checklist of timed items*.

Top right: An original paper form.
Static, poorly adaptable, error-prone.



RAMP ACTIVITY CARD [T]

JET AIRWAYS

DATE	FLT NO	DSTN	CAPT
REGN	BAY NO	ORG	SFS
STD	ATD	DLY	RED CAP
STA	ETA	ATA	AGENT

ACTIVITY	BOEING	ATR 500	START	STOP	REMARKS
CATERING	D-38	D-33			
STEPS FWD / AFT	D-38				
BELT/TROLLEY	D-38	D-33			
CGO RPTG	D-42	D-41			
FUELLING	D-38	D-30			
CABIN CLEANING	D-31	D-30			
CABIN CREW RPTG	D-35	D-30			
PILOT RPTG	D-25	D-30			
BAG. TRLY(FIRST)	D-22	D-24			
BRDG CLEARED	D-28	D-20			
SVC DOOR CLOSE		D-18			
1ST COACH RPTG	D-23	D-17			
BAG. TRLY (LAST)	D-10	D-10			
PUSH BACK RPTG	D-10				
LAST PAX O/B	D-10	D-07			
LOADSHEET O/B	D-10	D-07			
GATE NO-SHOW	D-10	D-07			
H/BAG RTRVD					
HOLDS CLOSED	D-05	D-05			
DOOR CLOSED	D-05	D-05			
CGO DOX			(Y)	(N)	LOADING ASST.
HOLD LOCK			(Y)	(N)	

DEAD LOAD(ARR)	CGO		BAGG.		TOB	
	MAIL		COURIER		TRANSIT	
DEAD LOAD(DEP)	CGO		BAGG.		TOB	
	MAIL		COURIER			



Photo: Olena Yakobchuk / iStock

2. Collaboration

A mobile app empowering airport staff to easily find each other, team up to address problems, and share information in a targeted way.

Context

2015–2016, for SITA

My role

- Functional requirements
- Interaction & visual design
- Styleguide for SITA iOS apps
- Admin app design support

Learnings

- Systems design
- Dynamics of people groups
- Conversational interfaces
- Benefits of separating business logic from UI design

Overview

In this second project we addressed the more general problem of efficient communication between potentially any groups of people at the airport.

Problem:
When facing an issue at the airport, it can be hard to find the right people and right level of information

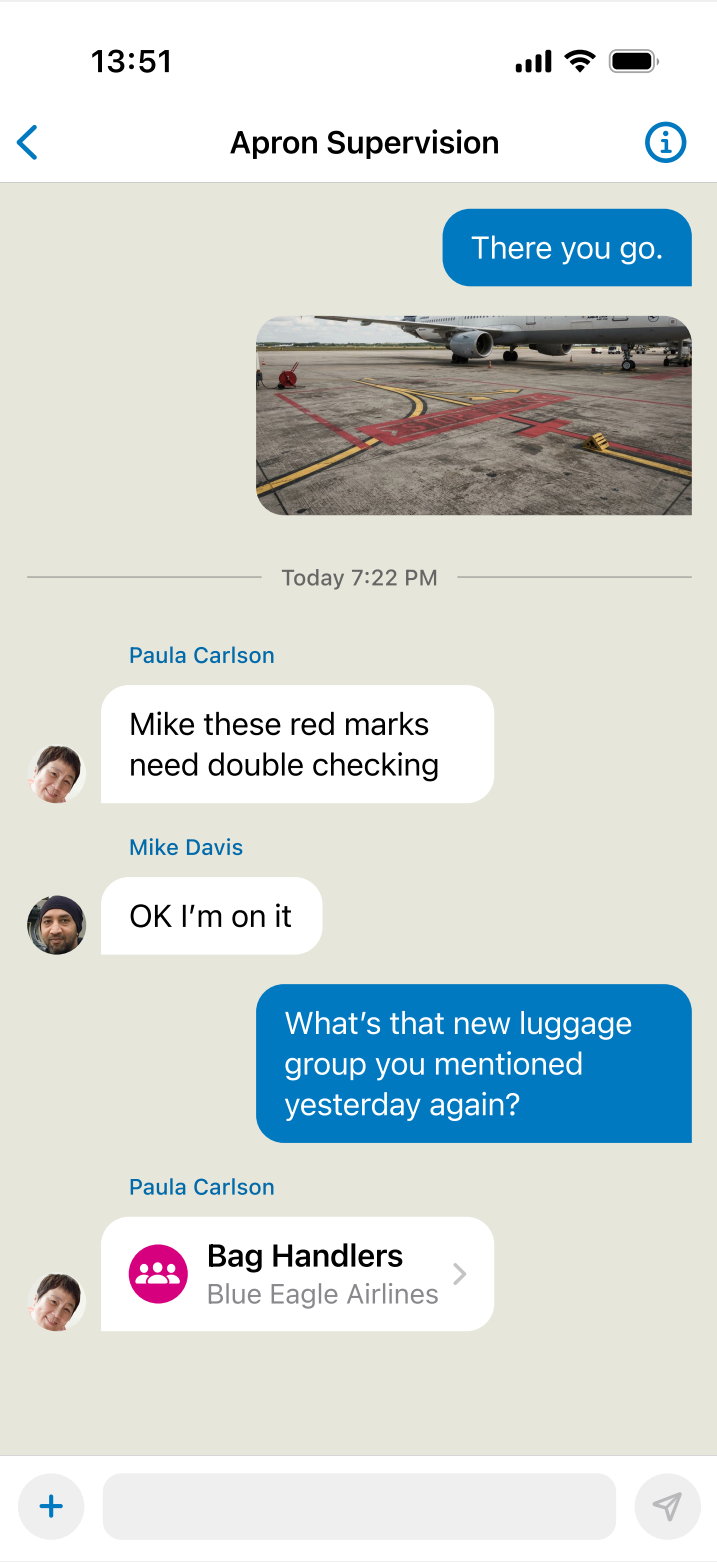
We made a mobile app for people to effortlessly gather to solve problems (planned tasks or unexpected issues) and communicate efficiently.

I based the design on apps like WhatsApp, but framed it to match the way airports work and to include features they needed.

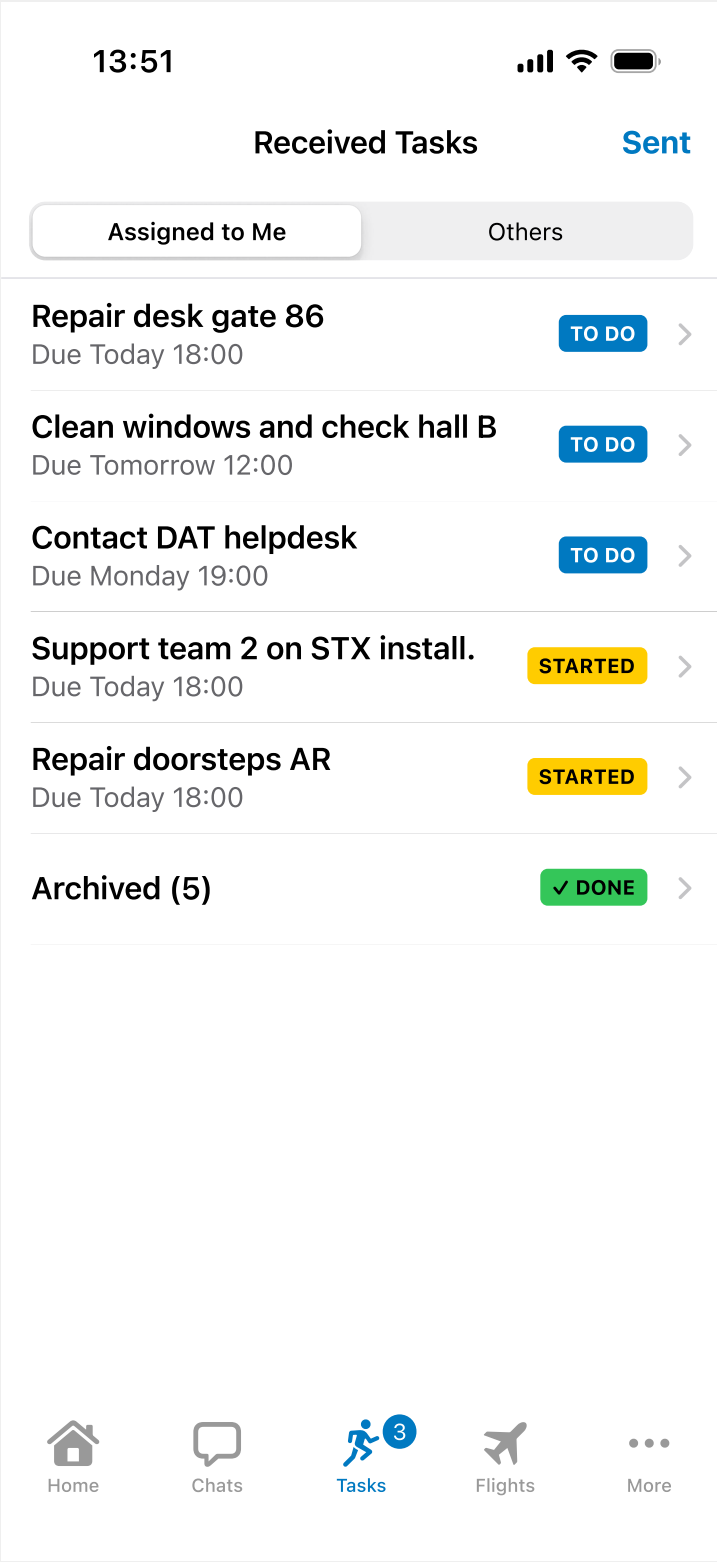
One challenge was to support both official and user-defined groups. This required careful design of admin rules like who controls the groups, who can, can't, or must join them, etc.

A central idea, simply put, was that the airport authority defined the hierarchy of all its tenants (all the companies present in the airport, from airlines to ground operators, to duty free shops), then each tenant could create the user groups they wanted in their domain.

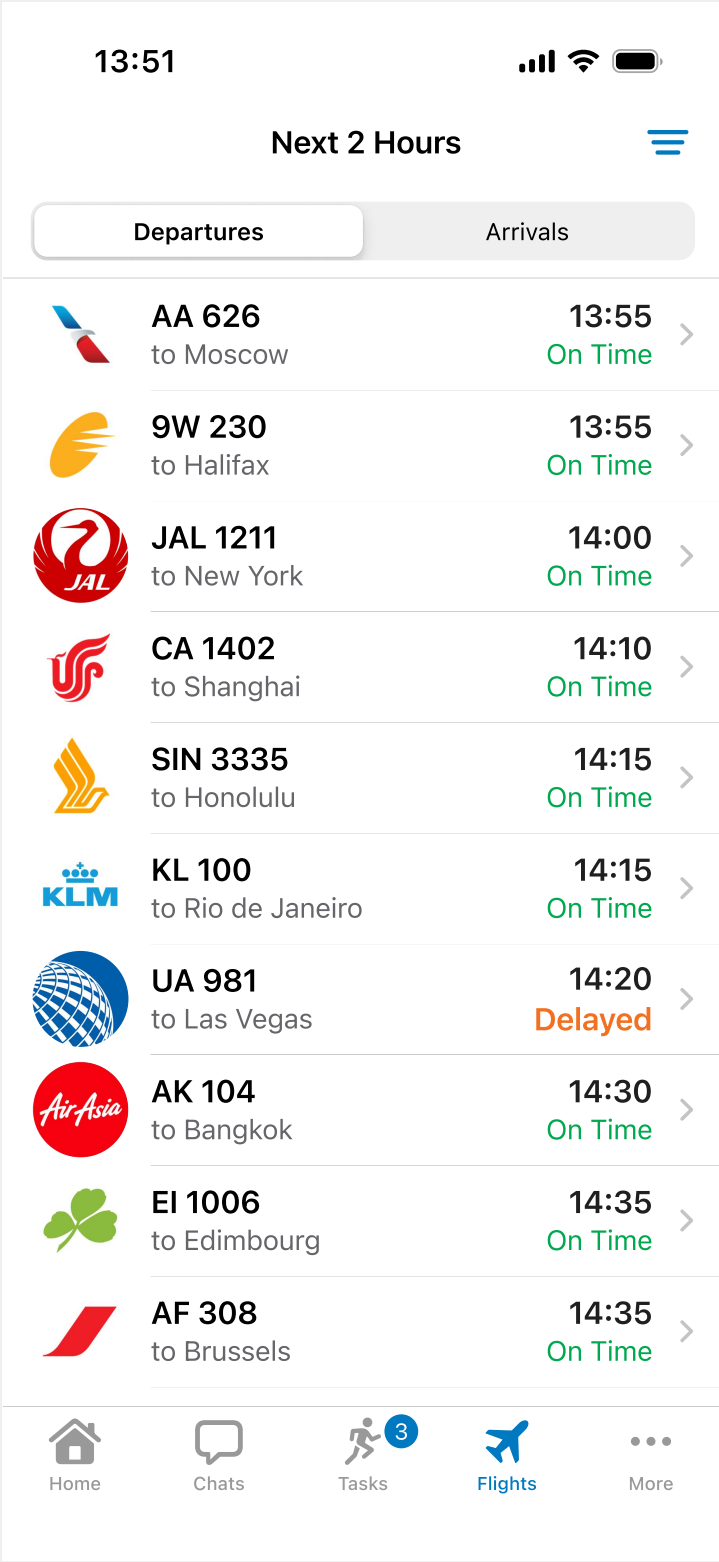
The hierarchy influenced who could see which groups, and this alone already gave shape to the main communication flows.



The chat view. Simple on the surface, but designed to feel familiar and offer useful share options.



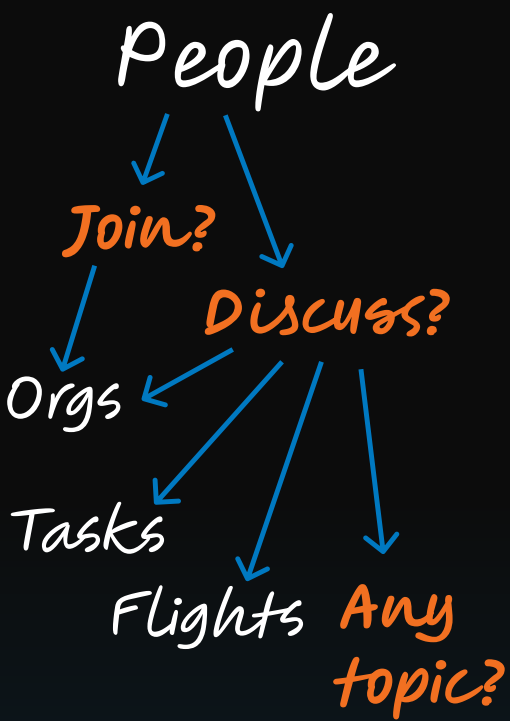
The task feature. Shows tasks assigned to the user by default. In addition the user can always check the list of other tasks and pick one there. A way to encourage people to help proactively and prevent delays.



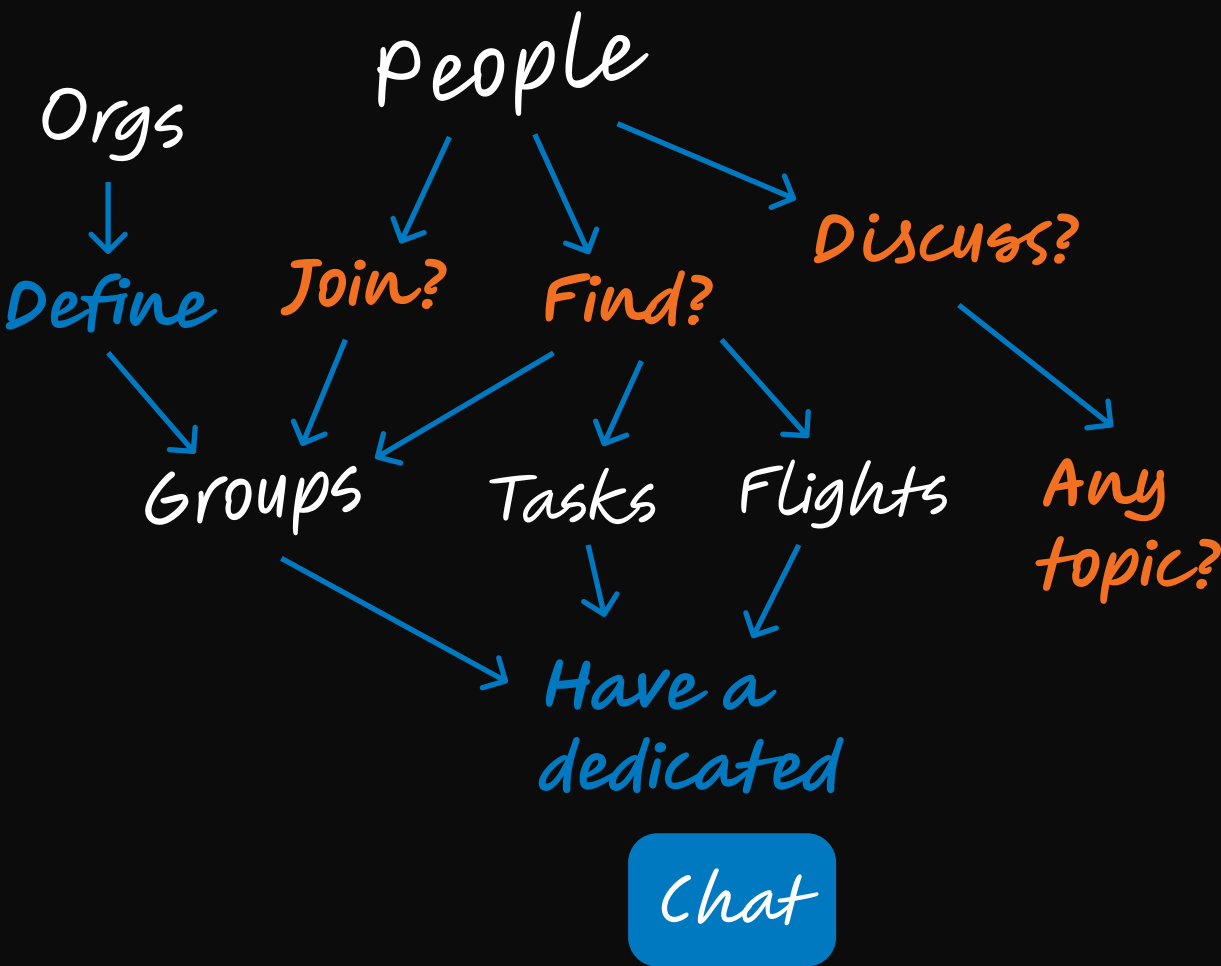
The flight board. With filters by airline, time interval, etc. Tapping a flight leads a card with all details, plus a link to join the chat about that particular flight.

Mapping out concepts

How to select the concepts, the strong centers that make a product meaningful to its users?



Starting with vague keywords and relations. People in organizations. Recurrent discussion topics (or objects of interest): flights, tasks to be done, the orgs themselves, etc.

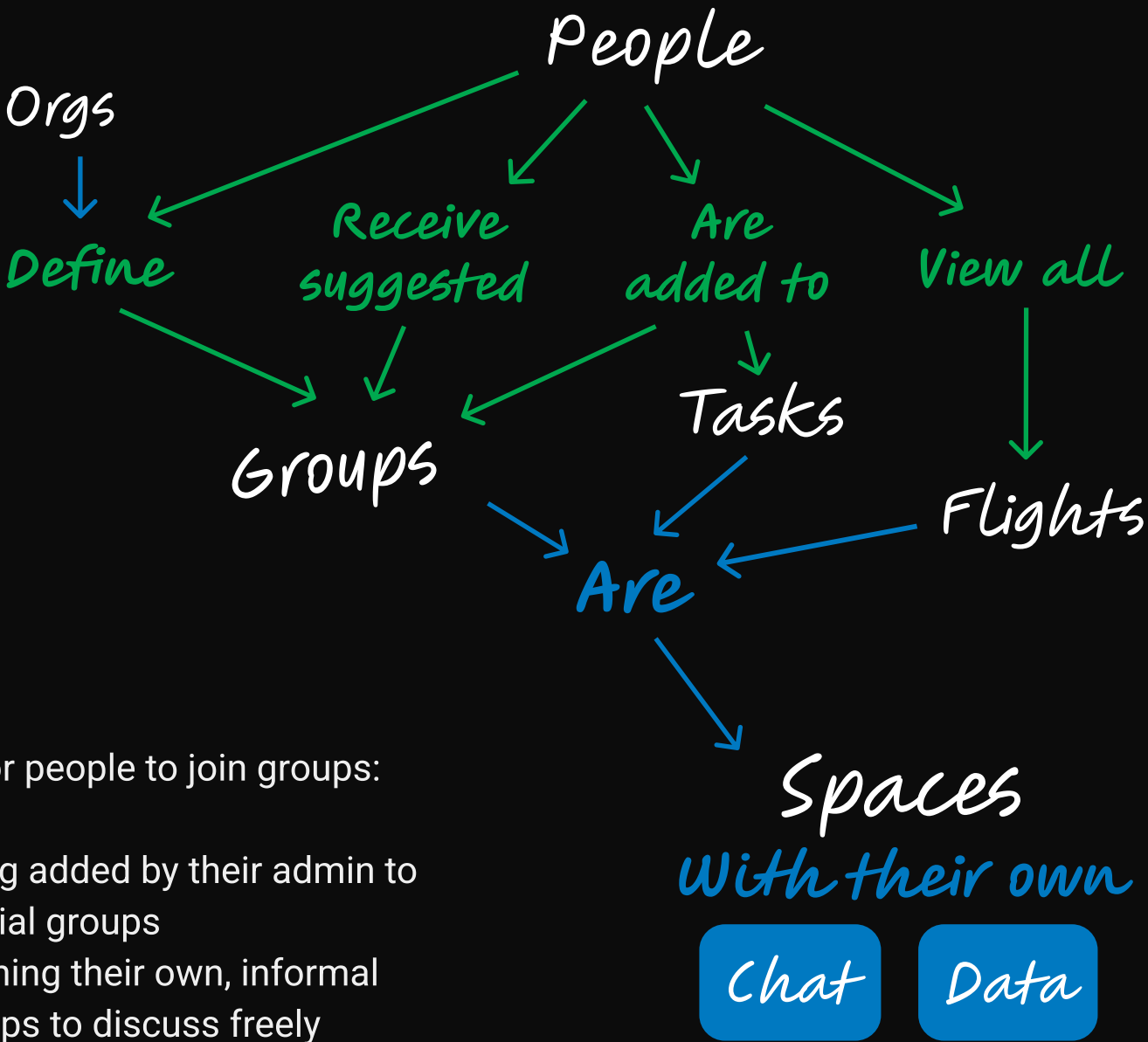


Refining... In fact people join groups, which are defined and exposed to them by their admin.

And just like people comment on a social media post, the best way to discuss a topic (such as task “Repair lights in T2” or flight AA102) may be to use a chat area *directly attached to it*.

Questions:

- How to join a group?
- How to find the available topics?
- How to discuss anything else?



Ways for people to join groups:

- Being added by their admin to official groups
- Defining their own, informal groups to discuss freely
- Via group suggestions

Users find a task when added to it by the task creator (until then it’s irrelevant to them), and can find flights via in-app flight boards.

I realized topics had become a broader thing: discoverable spaces that people jump in, to chat and view data in context – like a task’s status or a flight gate.

Next, a UI to give body to all that.

Handy topics that bundle their own chat and data... *The key was to make them really easy to find.*

Architecture

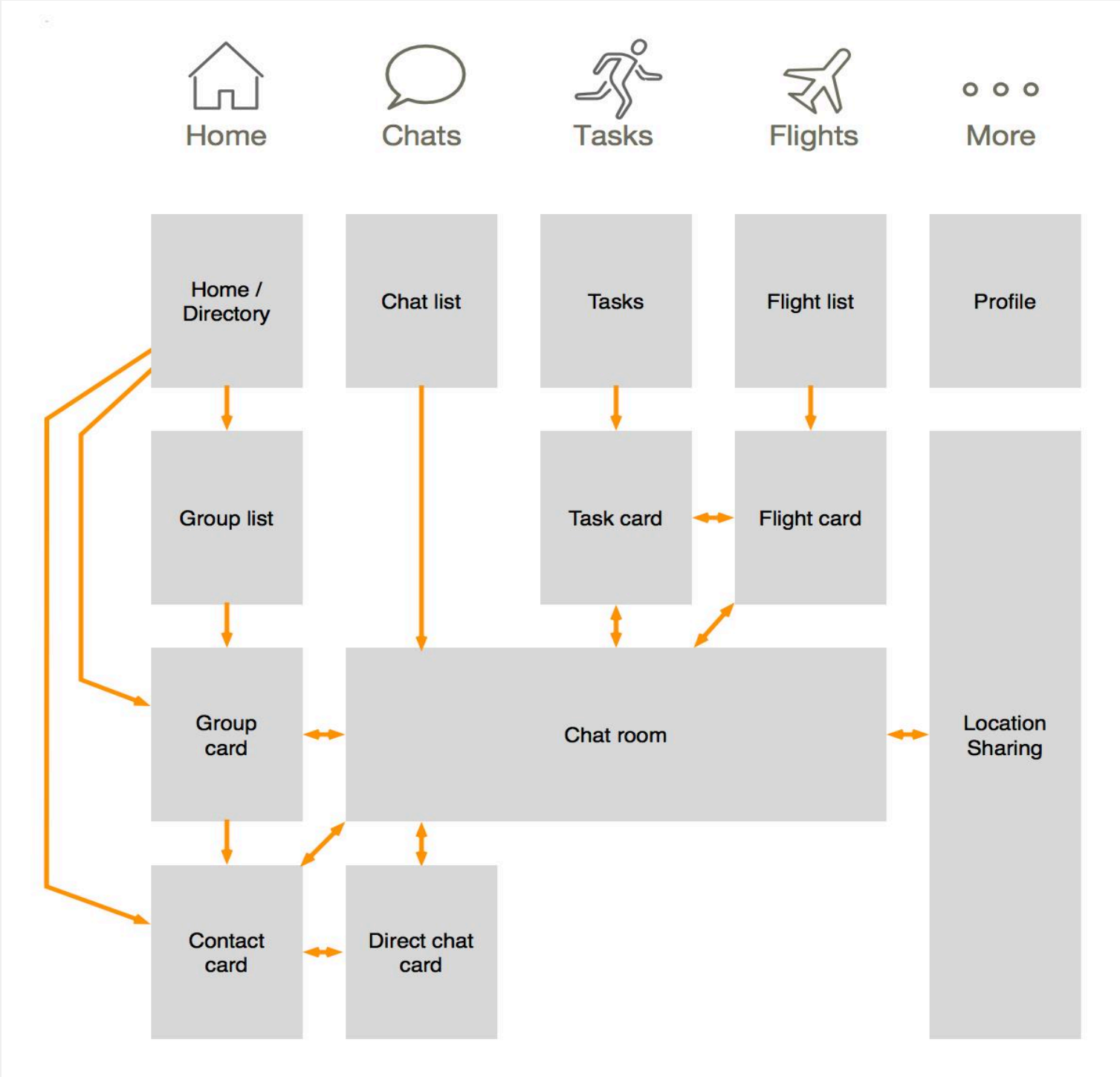
The architecture diagram shows how the identified concepts materialize into different views/areas of the app, and how they are connected.

In short, from the top level of the app the user basically accesses:

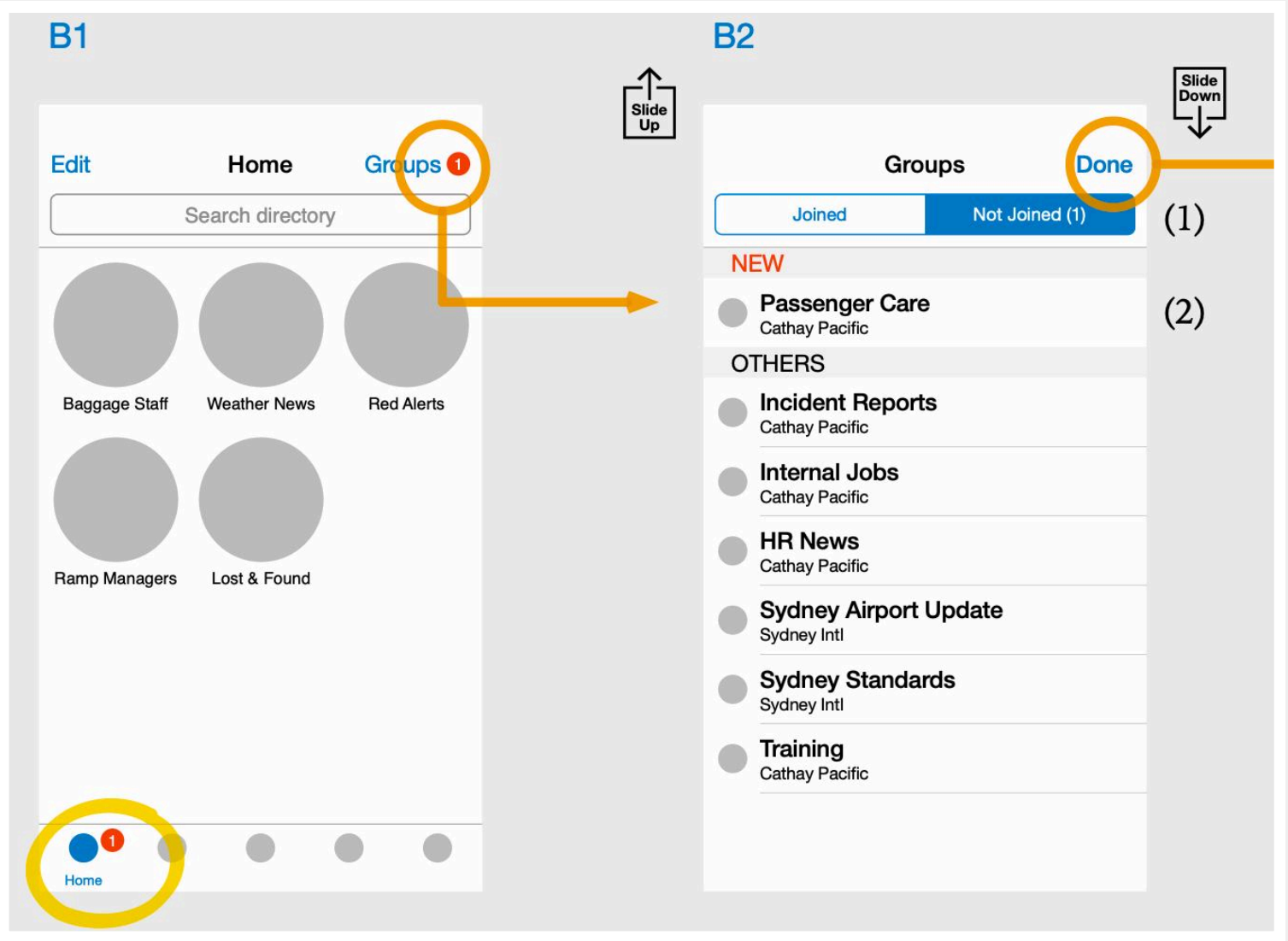
- Home, which features their groups and contact search
- Their chats
- Their tasks
- Flight boards

One level deeper is typically a card presenting details of the object selected above.

Finally, the chat level, where the selected object is being discussed with other participants.



Information architecture diagram.
Original spec excerpt. Not surprisingly, chat is very central.



The Home tab. Wireframe excerpt.
Where people go to manage their joined and suggested groups, keep shortcuts to any object, and search airport contacts.

The cards

Each card represents an object/topic and includes items that are either:

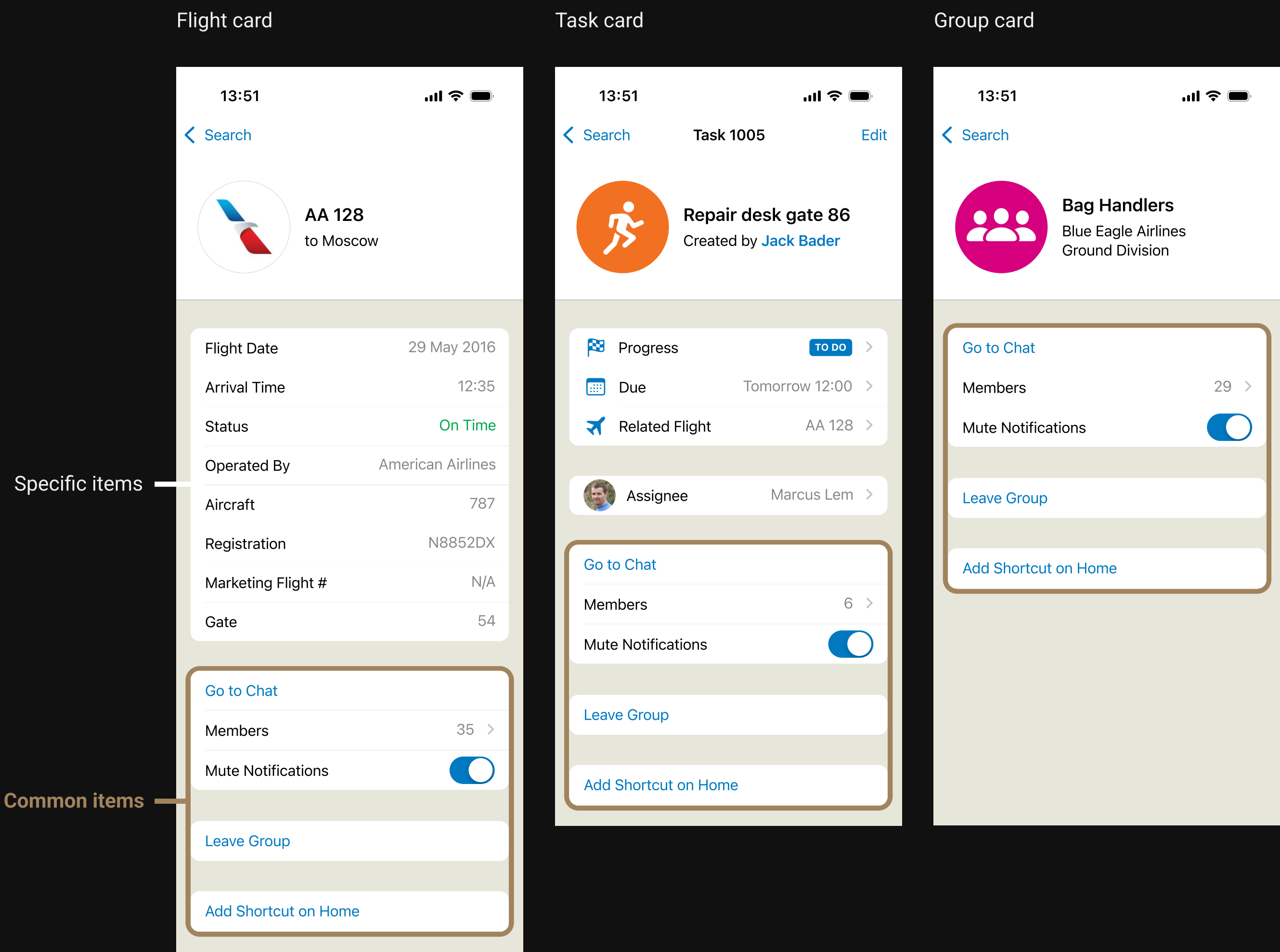
- Specific to the object type, like a flight date for flights
- Common to all objects types, e.g. actions related to their chat

I wanted the chat to always work the same way irrespective of context, in order to avoid creating dependencies (see *Orthogonality*, Interlude section).

Besides, to give users extra leverage I added universal rules such as:

- Every object can be shared with the participants of a chat
- Every object can be added as shortcut on Home

Consistent rules like these, when possible, clarify the whole, by cutting the number of special cases our mind has to deal with. Product is easier to learn, and users feel empowered.



How we worked

A geographically distributed team with PM, designer, and devs all in different timezones. SITA conducted a study with some airports to uncover needs. Design and development progressed in parallel. We presented prototypes to target customers and adjusted based on feedback.

Design wise I organized the scope in feature groups, and for each I did:

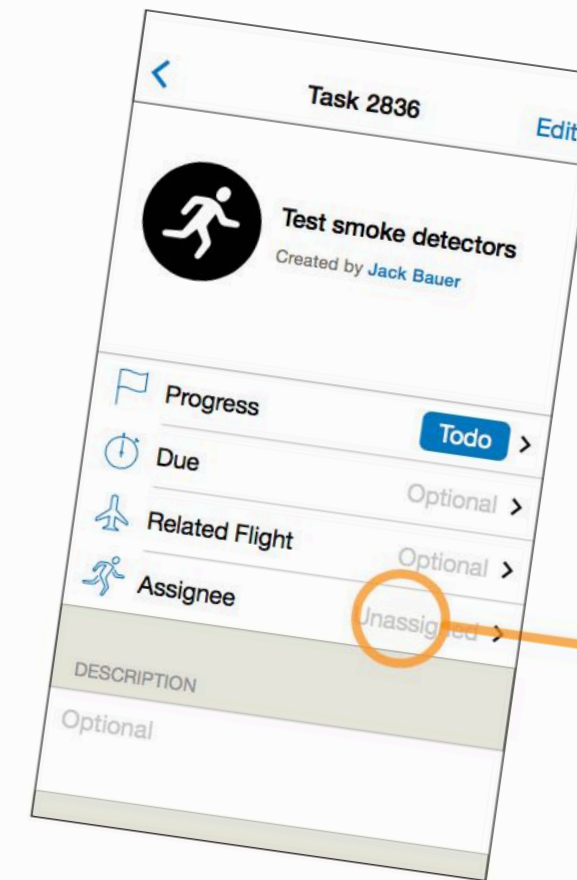
- Functional requirements and concept definitions
- Interaction design (wireframes)
- Visual design (specs & assets)

This made the design easy to grasp and update. Requirements formalize business rules that aren't directly related to the UI and are best captured separately.

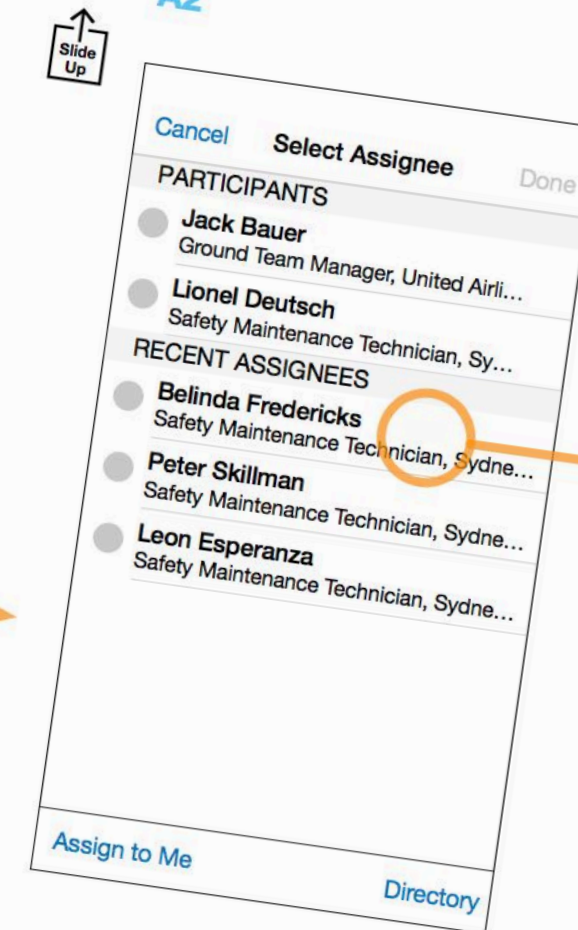
Wireframe example. Rather clear than comprehensive, it explains key flows, then for the rest I usually rely on the reader's common sense and experience, as well as on direct communication with them.

A. Basic flow

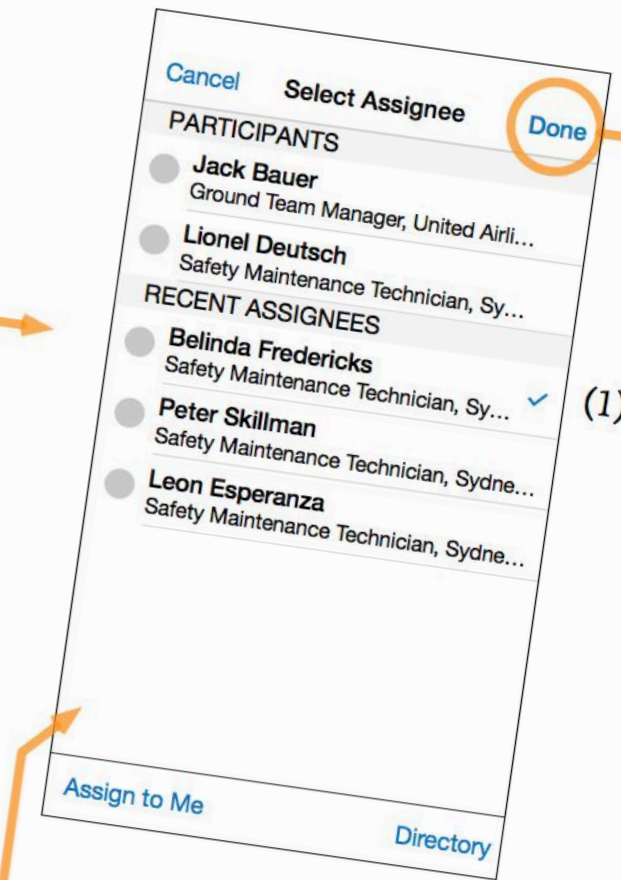
A1



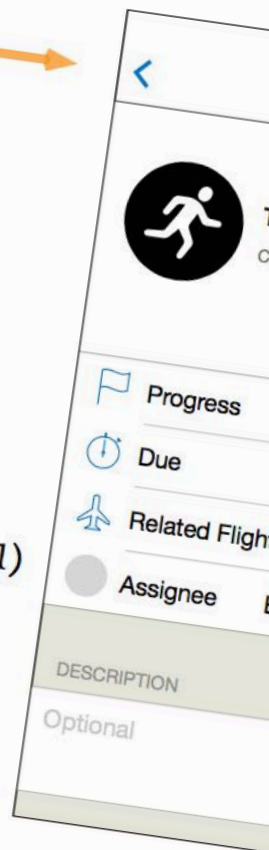
A2



A3



A4



Selector shows current participants, as well as recent assignees who aren't participants.

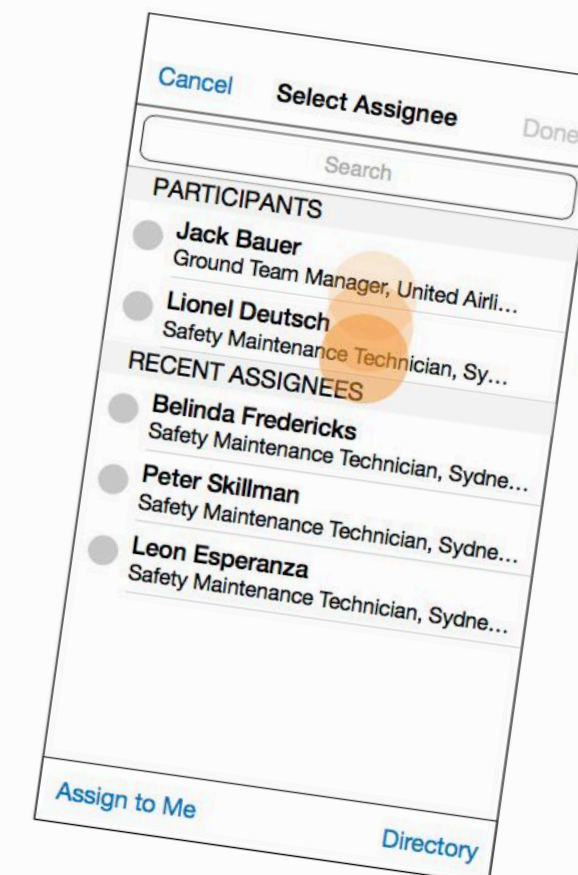
(1) Selected participant marked as the assignee.

Change is effective only after user confirms with Done.

(1) Assignee app instead of generic

B. Using list search

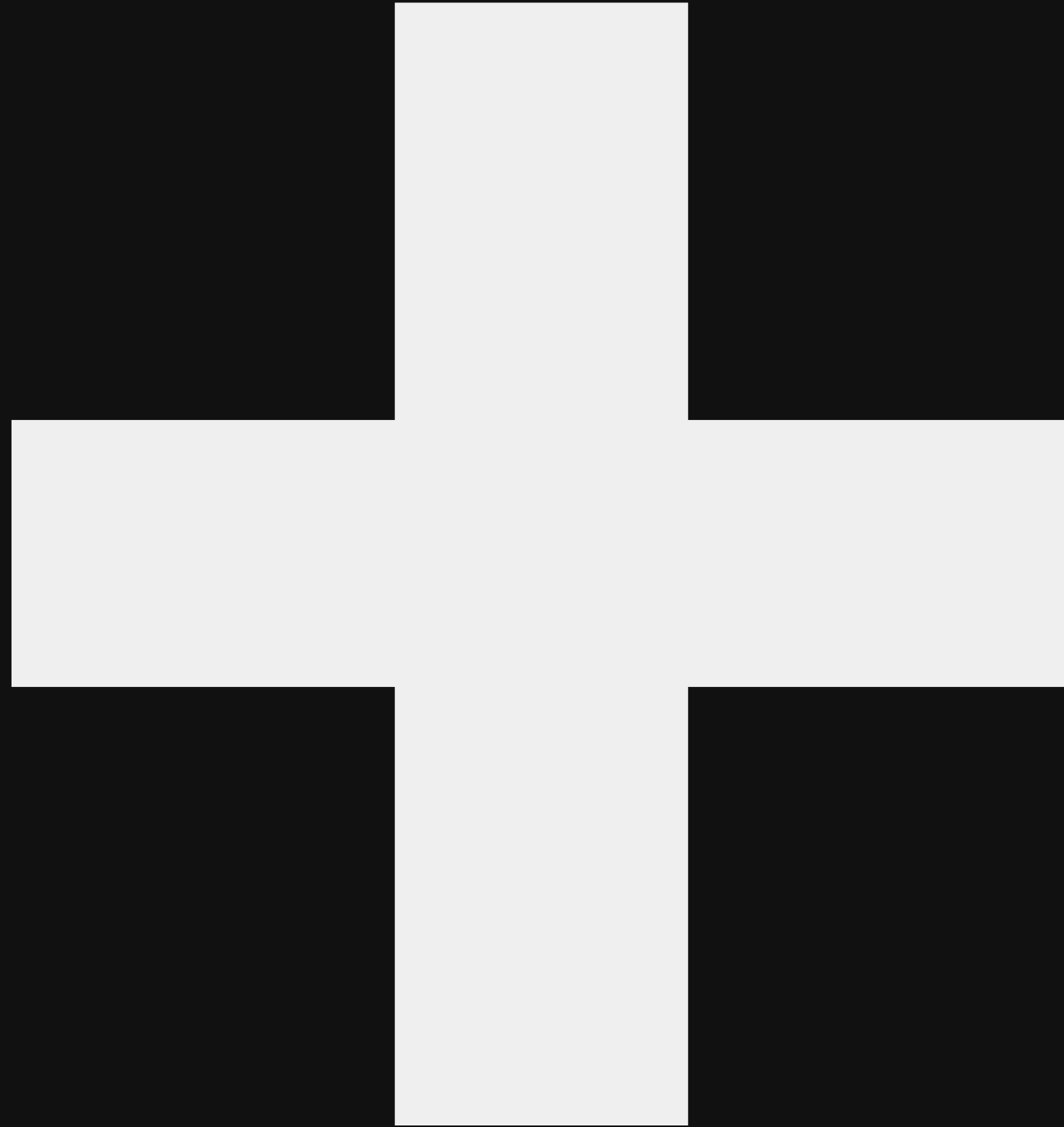
B1



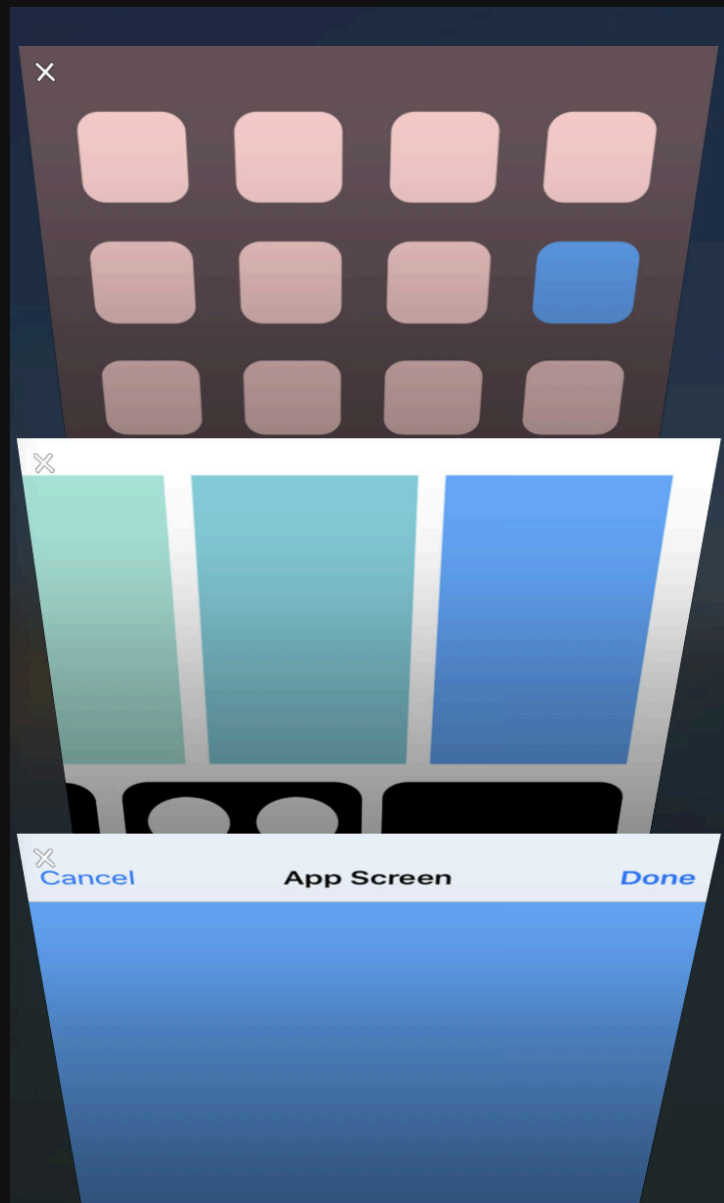
B2



Search can be pulled down and works as a list filter here.



More projects that
helped me grow as a
designer, analyst and
manager



Homeless iPhone (2017)

A few weeks before Apple announces the disruptive iPhone X, in this post I experiment with removing the historic Home button, and speculate rather successfully on how swipe gestures could replace it.

[Article on Medium](#). Went a bit viral after being featured on Jim Dalrymple's [The Loop](#).



The Sentinel is an extraordinary game (2017)

My essay on a video game classic created by Geoff Crammond. I discuss how embracing design constraints can lead to great system coherency and timeless esthetics.

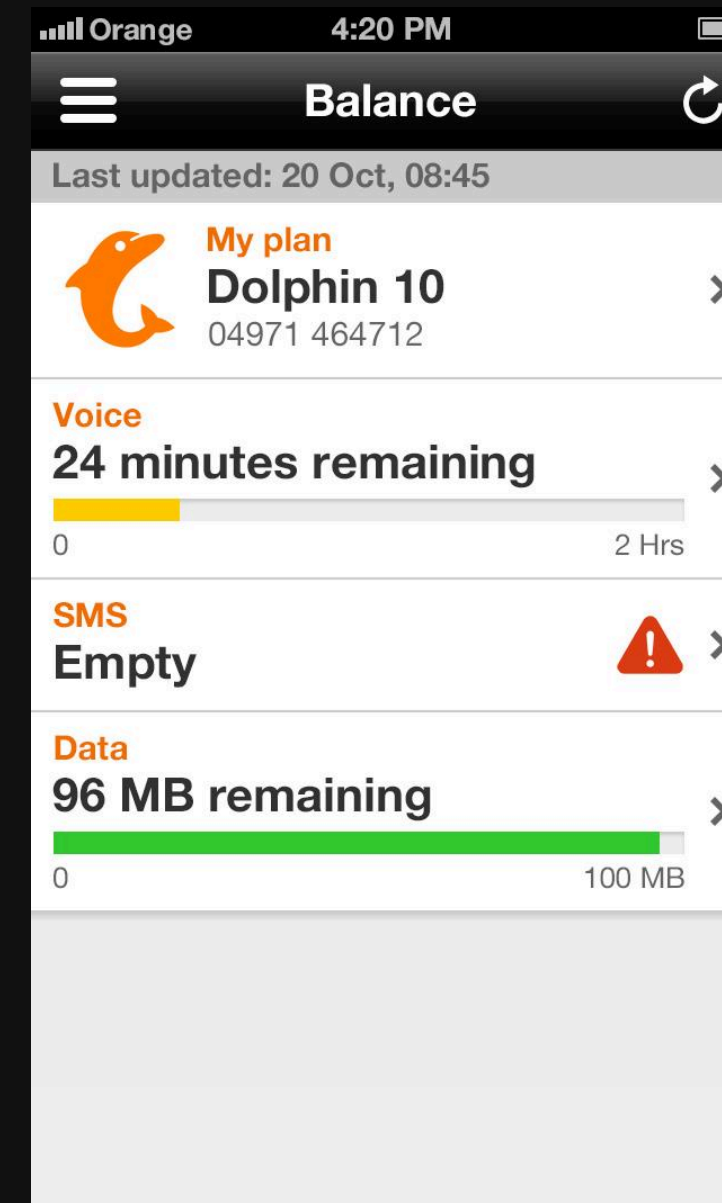
[Article on Medium](#)



Nokia X (2013-2014)

A fun, Android-based and affordable Nokia phone, just before Microsoft acquired Nokia. Memories of great user research and ethnography. Also my main team lead experience, at a wonderful company.

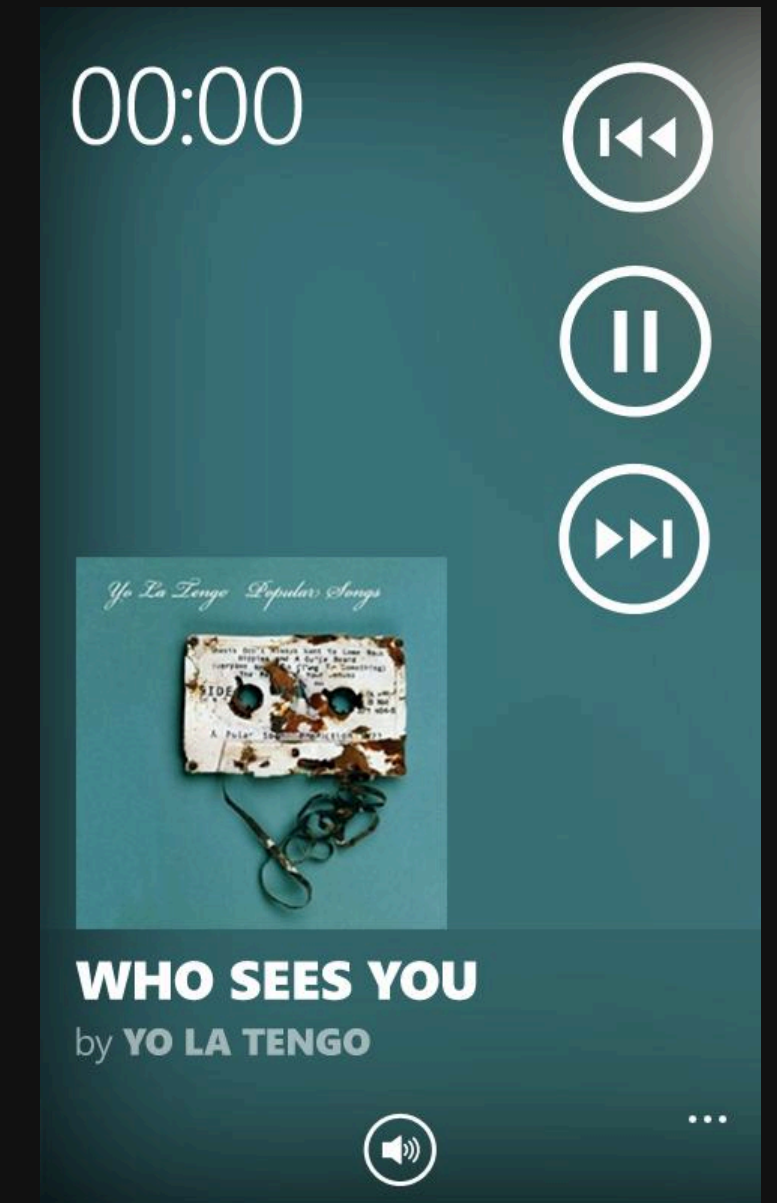
Team lead / Design lead / Design reviews and delivery / Direct contributor for some features



My Orange (2012-2013)

Evolution of the app used by Orange mobile subscribers to manage their account and options. A simplified and modular design to adapt to the needs of different markets. My main project in London, where I learned a lot.

Design lead / Information architecture / Interaction design / Design patterns



Casino (2012)

Minimalist app to stream music files from Dropbox. An experimental side project for Windows Phone, and wonderful experience of product maturation and radical simplicity. A lot of fun too.

Original idea / Interaction and visual design (Coding by Andréas Saudemont)

Contact

Email fabrice@uxriver.com

Site uxriver.com

LinkedIn linkedin.com/in/fabdubois

I'm grateful to many people who've worked by my side over the years. I want to at least credit the following design colleagues for their great support on the most recent works:

Lexy Chen
Laura Cunningham
Wenlan Dai
Jaeyong Lee
Lieven Verdin
Hanjie Yao

Updated: August 2026

uxriver