Doctolib Internship report

CUStomer Care cOntact Team

Johan Chrillesen ~ 04/04/2023 - 04/18/2023

Thanks

Before starting this report, I want to thank all the people working at Doctolib to their kindness, their attention and the time they've given to me. This internship allowed me to discover how you work in a big tech company that creates life changing products used by millions of people. I want to thank specially Florian - my internship tutor - for the time he accorded to help me and guide me each time I had a problem or a question. I also want to thank all the CUSCO team that has been the best team that I could have worked with. I learned a lot about new ways of working, ways of communicating and more really useful things that will help during my whole professional career.

Summary

The company

Doctolib in a few lines

Doctolib is a prominent healthcare technology company created by Stanislas Niox-Chateau in 2013. Since the creation of Doctolib in 2013, a main purpose drives all the work that is done : strive for a healthier world by improving the daily lives of care teams.

Today, Doctolib has more than 2800 employees between France, Germany and Italy. The tech department alone accounts for more than 800 people, split in different care teams, each responsible for a part of the product.

Doctolib's history

In April 2013, Stanislas Niox-Château and Steve Abou Rjeily decided to launch a new e-health project : Doctolib.

In October, they wrote the first line of code of the project.

In November, they launched the appointment service that allows people to book slots made available by the doctors. Doctolib is born.

In June 2014, Doctolib is deployed in a clinic in Boulogne-Billancourt.

In June 2016, Doctolib launched their mobile application that we all know today as patients.

In November, Doctolib is launched in Germany. One year later, in November 2017, Doctolib is deployed in the first « CHU » in Nancy.

In January 2021, Doctolib becomes the official partner of France for the COVID-19 vaccination. In December, 50 millions people got a dose of vaccination thanks to Doctolib appointments.

Today, Doctolib counts more than 2800 employees, 70 millions patients and 320.000 medical practitioners.

Our products

We have 6 main products that have add-ons that can be buy on top to improve their capabilities :

Medical Software

Doctolib medical software (EHR) is our product used by medical doctors to handle medical documents, invoices, communication with patients.

Discover more here

Telehealth

Doctolib Tele-health allows practitioners and patients to meet online, it reduce the numbers of missed appointments and have a secured billing system integrated to the software.

Discover more here

Patient

Doctolib Patient is our older and our most commercialized product, it helps practitioners to gain time and avoid missed appointments. It allows patients to book their appointments online so the practitioners don't have to handle it themselves. It also improve their visibility online.

Discover more here

Hospital

Doctolib Hospital is our product that facilitates the reception of the patients at the hospital, and includes Doctolib Patient, Doctolib Team and Doctolib Tele-Health

Discover more here

Team

Doctolib Team is our solution to allow multiple practitioners using Doctolib to communicate about patients, share documents and even transfer cases directly to another practitioner in the software

Discover more here

Kiné

Doctolib Kiné is currently only available in France and is the equivalent of Doctolib Medical Software but modified to comply the specific needs of the physiotherapists

Discover more here

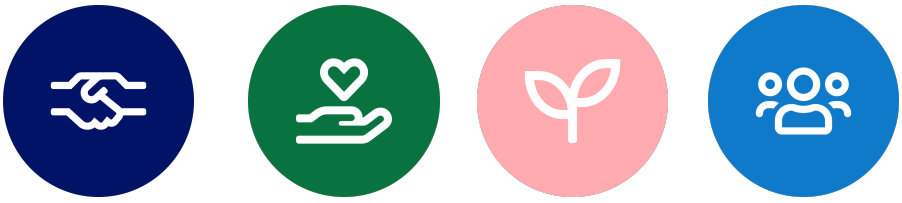
The work environment

How it begins

When you start your journey at Doctolib, you will attend to your Doctolib Academy for 1 week, it regroups every new joiners of the month from France, Germany and Italy. This can be attended in France HeadQuarters in Levallois-Perret (that's what I did), but it can also be followed online, as every meeting is recorded and shared between the 3 countries attending. During this week, you will learn everything you need to know about Doctolib, from how to enter a building with your badge to know the next big features that Doctolib wants to ship to their users.

These informations are given with conferences made by Doctolibers that work on it. For example, at the beginning of your Doctolib Academy, Stan is presenting you the foundations of Doctolib and why it is what it is today. Next, Matthieu Birach (our Chief People Officer) presents how you can grow and learn the best you can at Doctolib. Followed by many conferences about Design, Communication, Tech and Sales teams.

Once this week is finished and before starting working, you will attend to your tech academy. This learns you everything you need to know about tech at Doctolib. You will have various sessions across your 6 first months at your job. After a total of 2 weeks, you might be able to start your job efficiently, with a working dev environment and understanding how we do things at Doctolib Tech Department. You will also know the 4 pillars of Doctolib :



Act Care Learn Serve

How you will work

Currently, we are working in a hybrid mode, this means that some days we are working from home, and other days we are working at the office. You can choose what you prefer and currently you have to be at least 2 days per week at the office (Tuesdays and Wednesdays in my case) but it will change in mid-July to be at least 3 days.

For your dev environment, you will be able to choose between :
- A Thinkpad laptop with the Linux distribution you prefer
- A Microsoft Surface on Windows
- A M1 Macbook Pro with MacOS (that's what most developers have)

What are your benefits

As an intern, you have (almost) every benefits that a normal employee have. That means that you will have a special card to buy food (Swile), you can access every offer of the CSE (after 3 months of seniority), you can attend any activity organized by Doctolib. After 3 months of internship, Doctolib offers 0.5 days of vacations per months worked and many other stuff (like working from an other European country for 10 days in a year).

The Tech department

The Tech organization

The Tech organization is pretty simple, it's divided in multiple domains, themselves divided in multiple feature teams, each responsible for a part of the product.

For example, there are these domains :
- SPICE (Security Privacy and Identity ComponEnts / Security)
- PEER (Platform Engineering Efficiency Reliability / Databse, Dev Tools, CI, Design System)
- SMP (Sales and Marketing Performance / Billing, Licences, Data, Onboarding)
- CCP (Customer Care Performance / HelpHub, FAQ, Webcallbacks)

Focus on the CCP domain

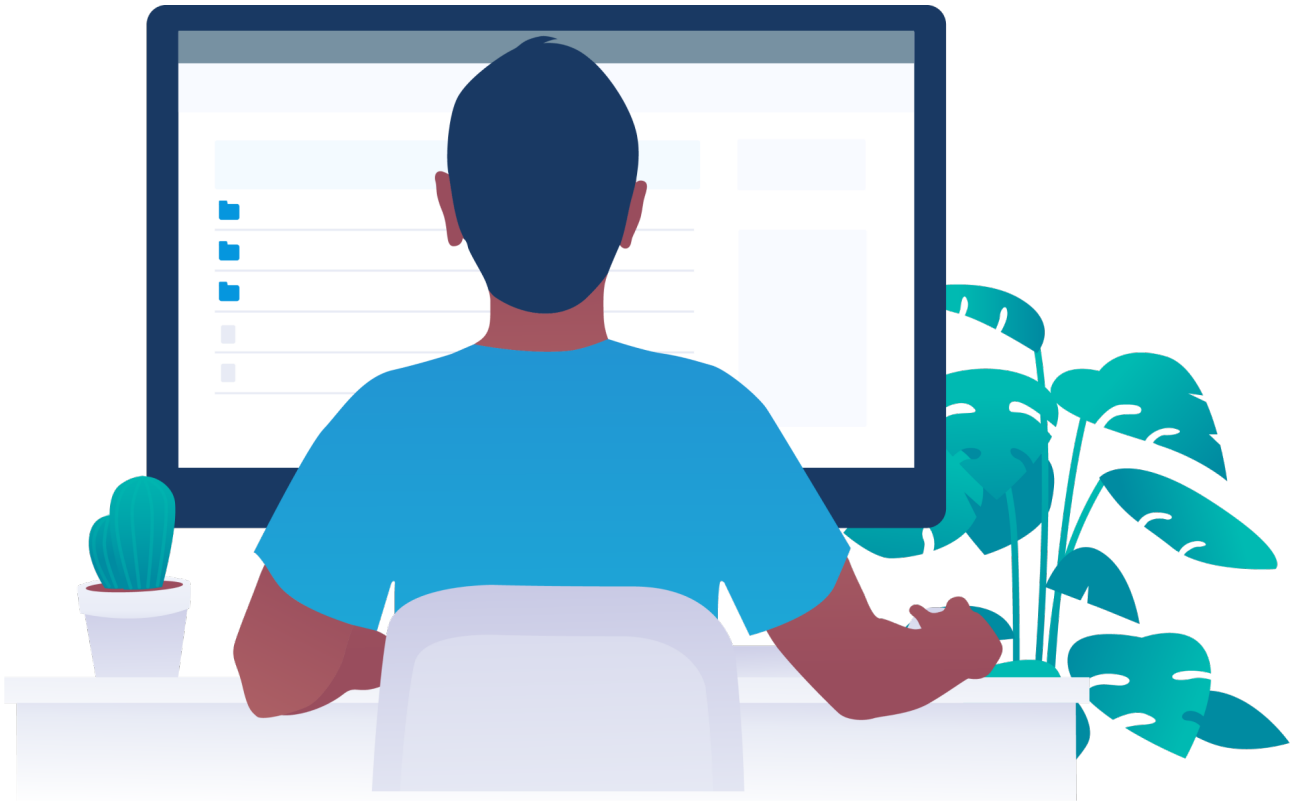
I worked in the CCP domain, its main goal is to improve the quality of the support provided to the practitioners. It's composed by 3 feature teams :

- CODEF (COntact DEFlection), their mission is to make our support scale by fostering users' autonomy thanks to an intuitive product and easy to use self-serve ressources.
- KHEOPS (ProduKtivity enHancEment for OPerations in Salesforce), they work on our Salesforce platform to help customer support agents to access the cases created by the practitioners
- CUSCO (CUstomer Support Contact), this team is in charge of all the contact channels that a user (patient or doctor) can choose to contact the customer support.

Focus on the CUSCO team 🦙

First, CUSCO stands for Customer Support Contact. I worked in this team, we mostly work on the HelpHub available in the product and rooting the users to the best contact channel to contact us (written or call). The team (as every other one) is composed by developers (5 with me at CUSCO), a Product Manager (PM) that identifies what projects we have to do and that describes the key goals of the project. The team is also composed of an Engineering Manager (EM) that oversees projects management processes and coach their team members and a Staff Engineer (SE) that is responsible for the quality of the code produced by developers, he has the main vision of a project on the tech side.

Your job as a Software Engineer



Our technologies

We are working with Ruby-on-Rails (RoR) for the backend and React for the frontend. Historically, we used Slim for the front-end, that was easier to develop and to integrate to RoR (as we wanted to grow really fast). Today, we are aware that better solutions exists to have a more scalable and maintainable product. We are using Github to version our code. You will also have the choice over which Integrated Development Environment (IDE) you want to use, even if we are mostly using RubyMine and Visual Studio Code. We are working on a monolith, this means that ALL our codebase is stored on a single repository.

Your tasks

As a developer working in an agile environment, you will of course have to write efficient code part of the current sprint. But your work won't stop there, you will also be a real member of the team and will have to provide quality reviews on pull request for other developers of your team, you will have to challenge the functional specifications. You will also have some specific tasks to do as being the Tech Holder of a sprint (its a bit like a Project Manager). With your Product Manager, you will have to technically specify the mission / steps and documentation of a sprint.

You will also be on Dev Duty, that means that during one day, you will have to deploy the code in production 3 times (at 10AM, 1PM and 5PM), keep master green (as we have automated tests ran only on master) and revert any commits that would break master or the production.

Other than that, you will also have the opportunity to attend to a learning program called 'Starsky And Hutch'. It's a 1 hour meeting, once a week that you will attend with an other Doctoliber with more experience on the stack. You will create project that could be added directly to the product, it will allow you to ramp up on the Doctolib stack, gaining skills and comprehension on the codebase.

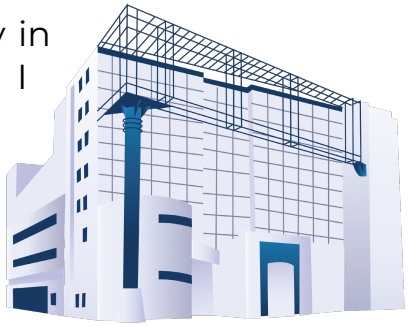
Your time matrix

As a software engineer, your time will be shared between 4 main activities:

- Code new features, it means coding new things described in tickets that are part of a sprint (around 60% of your time)
- Resolve bugs, it means investigating and fix different problems coming from the codebase (around 15% of your time)
- Various tech tasks, such as reviewing pull requests, pair code, ... (around 15% of your time)
- Tech holding, keep you at the page on the codebase, be on dev duty, read and challenge functional specifications, ... (around 10% of your time)

My work at Doctolib

My Doctolib Academy / TechCamp



As I said before I attended my Doctolib Academy in Levallois in beginning of April. It was great as I learned more about Doctolib and discovered more than just what we - as patients - know about the company and the product. It also learned me a lot about how we work at Doctolib and all the opportunities that I have by working there (formations, CSE offers, ...). The Tech Academy allowed me to take time and understand the scope of each domain / feature teams working to make a better product and a better developer experience.

Starting to work

Before starting to work and develop my first features, I obviously had to install my dev environment. All the dependencies, technologies and modules needed to run Doctolib being really huge, we have a tool name DCTL that is basically a shell script installing and configuring everything you need, using Ansible. It creates an SSH key and link it to your Github account, install dependencies using your package manager, clone the main repository of Doctolib and many more stuff. It's really game changing as you almost have nothing to do except adding DCTL to your path and executing it.

Development flow

Before starting to code, you also have to understand how you ship code and how it works to add your code to production.
First of all, you have to create a branch out of master named with the Jira ticket reference (like CUSCO-456), and then you can start coding. When you are done, you can create a pull request (PR) and ask the review of your team and the teams owner of the files you modified. While these reviews, you can comment /build to your PR, that will start all the CI checks needed to validate your modifications (end to end tests, unit tests, front-end differences, ...). Once your branch is merged, you - usually - don't have more to do, it will automatically be deployed to production. Well, not that automatically, everyday there is a Dev Duty - as I explained before - that is responsible to deploy the codebase to production. To help him do this, we have a Github Action named the « choochoo », this Github Action simply takes the latest commit that passed all the CI checks on master

branch at 10AM, 1PM and 5PM. With this, we can have the hash of the commit and then merge it to a branch called « production » with an other tool named « Dev Cockpit ». Once you've done the action of merging to production, a script is launched and is refreshing all of Kubernetes pods containing the app, and updates everything there is to update with the changes. Once this is done, you still have to monitor if nothing is broken, to do this we have two tools, Sentry and Datadog that gives us all crashes, alerts or exceptions returned in the products. If there is a spike of an alert, it means that there is an error and that you might want to come back to the latest stable build and fix the issue before pushing to production.

Code architecture

The last thing you need to know before starting to code is the Doctolib architecture. Doctolib is stored in a monolith. That means that all our code is in the same place in Github and everyone at Doctolib Tech Department has access to everything related to the code of Doctolib. In this monolith, the code is often separated into engines that are each a part of the product. For example at CUSCO, we have our proper engine called « customer_support_contact » that contains the user flow after clicking on the HelpHub button. The fact that it's a monolith can be scary at the beginning as you are in front of a huge quantity of code at the very start of your job but the more you work on tickets, the more you understand different parts of it. The engines have 5 stages of conception :

- Stage 0 : Code is almost not organized at all by functional components
- Stage 1 : Code is organized by functional components, but any component calls any other component without restriction.
- Stage 2 : The engine exposes a public interface that is semantically correct, explicitly describe is responsibilities. All inbound dependencies use the engine's public interface.
- Stage 3 : The engine only use public interfaces of other engines and is involved in no dependency cycle.

At Doctolib, we aim to have all our code in engines at stage 2 - that's the case for ours at CUSCO.

Most of the engines are divided in two parts :

- The back-end composed by :
  - Controllers, that handles all the API calls
  - Models, that handles all the database queries and all the code logic linked to it
- The front-end that displays the app.

Time to code !

Now that you know everything you need to start coding, I will give you examples of what I did this far in my internship.

My first sprint : Clean the ticket model

My internship started at the end of a sprint that aimed to clean a model and a table in our database that we aren't using anymore. I haven't done much during this sprint as I was just finishing my Doctolib Academy, and still discovering new things in my Tech Academy. I had around 3 hours of meeting per day to explore the whole tech department, and learning how to efficiently work at Doctolib. The main learnings I had during this Tech Camp were during the sessions allowing us to take control of the various products we have at Doctolib, as the PRM and the EHR. I also had sessions of pair coding, helping me to understand how to build features according to our guidelines. Between the sessions of my Tech Academy, I've done various trainings on a website that we have called Uniq, you can learn Ruby on Rails and React. As I had no experience on Ruby on Rails so I did the most trainings I could on this, basically it's separated in two parts :

- Lessons, they are written and are routing you to parts of the official rails documentation
- Trainings, it's exercises made to help you discover how to code efficiently. It is a project aiming to code a basic todo list only using Ruby on Rails from the back-end to the front-end.

Once I was more comfortable with the technologies we are using, I started to work on the sprint by doing a parameters validation for one of our controllers. Ruby on Rails allow us to verify the parameters with 'safe_params', it allows us to verify query parameters that are required or permitted. I replaced this by a 'dry' validation, it creates a contract interface that can be validated. More than just requiring and permitting parameters, it also allow us to verify the type, the length or even the format of a query parameter.

The sprint was also focused on implementing Elastic Search to improve the quality of our support. The support agents often search for the practitioner having a request before contacting him back, the problem was that this research can take a long time. We decided to add the required data to an elastic search node to be able to use elastic search on this research. To be sure to have all data updated with our main database, I added listeners to various events that would occur when we need to modify our node. Each time

once of these listeners is triggered, we update our node so that it's almost instantly synchronized at all time.

During this part of the sprint, I also did a 2 hours pair coding with Braden, that was about improving the elastic search request to make it the quicker possible. It was mostly about reading the documentation and updating our configuration files.

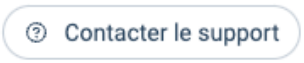
I learned a lot during this sprint, as it was obviously the first one at Doctolib and by consequence on a monolith but it was also the first time I worked with elastic search. I discovered the codebase and started to understand how the code works.

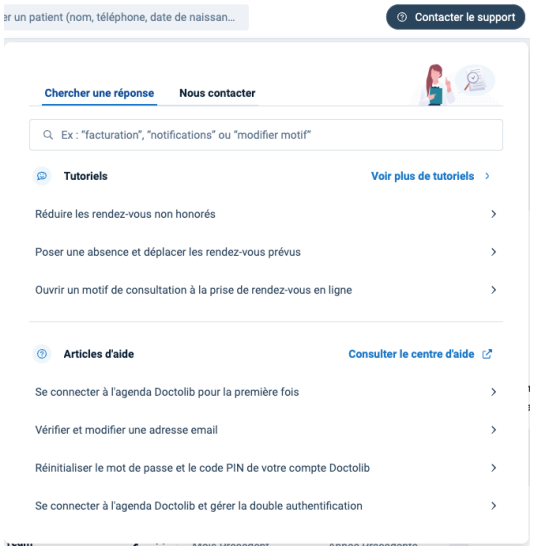
Next sprint : Merge the forms

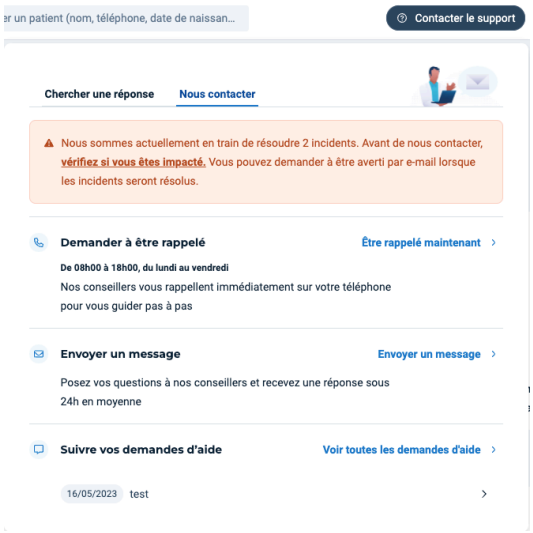
This sprint started by a special week called ✨ **The test week** ✨. This week aimed to reduce our CI financial cost. In fact, each time we run actions and tests, it launches multiple pods on Kubernetes. The more time the tests are lasting, the more pods are launched, the more money it costs to Doctolib. Indeed, we have a small issue about our CI cost at Doctolib, as we are working on a monolith, the tests of every single team at Doctolib are ran every time we need it, and more importantly they are multiple runs at the same time. All of this makes that when we run tests, it often takes more than 25 minutes to execute them. Meanwhile, the developers can't merge their pull request and can't fix potential issues that could occur. Some improvements have already been made with several projects, as running only necessary tests by checking which files were modified and decide to run only some tests that could detect potential errors. Despite all of this we are still looking for improvements, that's why this special week. We worked with CODEF and targeted our tests taking more than 20 seconds.

To achieve this, we contemplated the idea of transitioning from Capybara and Minitest to React Testing Library for our end-to-end tests. This potential shift aimed to optimize our testing process by leveraging the benefits offered by React Testing Library, provided it was a viable option. The intention behind this consideration was to explore a more streamlined and specialized testing framework that aligns closely with React components, ensuring better test coverage, improved maintainability, and increased overall testing productivity. Once this week was done, we had a better shaped CI and we now all know how to do RTL tests and the best practices to create tests that will not need to be refactored in the future.

The sprint continued by the merge of forms. In the help hub of the software, users can access all of the help they need from a button.







When they click on the button, users can either access the tutorials and the FAQ, or the contact support. We worked on the part where the users can directly contact the support. When they want to contact us, the users have a clear choice about the channel that they want to use to ask their question (phone call or written contact). We already know that a voice contact is almost always more efficient than the written contact. That's the reason of this sprint, we wanted to to route the user to the most efficient way to contact us. Before, they had two entries to fill in a form that is giving us informations on their issue. Now, we have a single entry point, for a single form that is then routing to possibly an other form if we detected that the most efficient channel for them was the written one (detection based on the product and the main reason of their issue).

To start this sprint, I changed the links that were previously used by the new ones, and I also redirected the old link to the new one (in case a practitioner had saved the link in his favorite links or if someone tried to access the old link). This task has been more difficult than it seems because it has been slowed down by an architecture change coming from an other team. To understand this, I need to explain you how the code is owned by different teams at Doctolib. We have a Codeowner file in our repository than list every files or folders and assign a team that owns it, this means that the team is responsible for specific files. If you need to modify a file that is not owned by you, you need to have the approval of the teams that owns it. The problem comes here, the file containing the React Router of the app was owned by a team called SAPE. This team is composed by Senior Staff Engineers and Senior Architects. While I was asking for their review, they removed themselves from the owning of various files, including the one that I modified, because they aren't meant to own files. The major

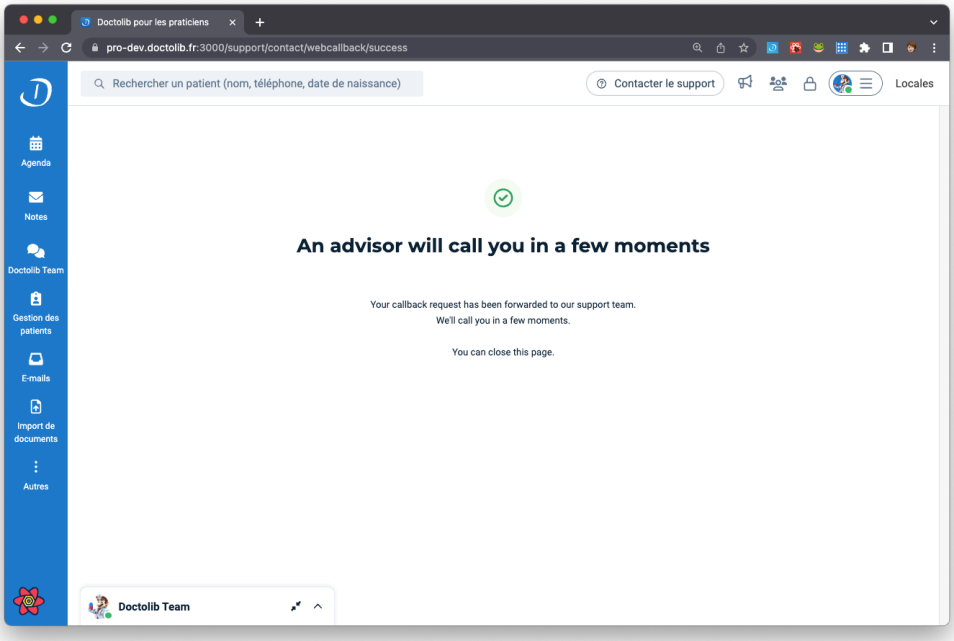
problem by doing this was that they didn't mentioned a team that owns the file int heir place, so I had to because you can't modify a file if there is no code owner on it. The fact that this file is the access point to the whole product made that not a single team has the scope to handle this. A team finally took the ownership of the file while a more stable solution is found.

Once this has been done, I have implemented the success view of the web callback form (voice channel).

Before explaining more things I want to say that for this sprint that is mostly front-end, we've completely refactored the forms that we are using, to create a new form using micro front-end architecture. This architecture is pretty simple when you think about it, you have :

- A section, containing the router of the feature
- Modules, that are basically the difference pages
- A controller, that handles all the logic of the front-end and is called by the module.

This was my first front-end task, doing front-end is really easy at Doctolib as we have a complete design system that we can use to add buttons, drop-downs, cards, text, images, ...



Next sprint : Allow agents to better handle users asking for a recontact

I am tech holder of this sprint (currently ongoing), this means that I had to specify all tickets that will be done during the sprint, explain why we do this, how we will do it and create a documentation of the sprint. I had to do technical choices and I have to be sure that all the project requirement are done. This document is shared to the whole tech department at Doctolib and is available to feedback and changes. I also had to prepare several presentations for the CCP domains architects, staff engineers and developers.

First of all, you need to know that once a practitioner has opened a case, it is transferred to Salesforce where our customer support can answer the case. This sprint was done closely with KHEOPS team from the tech scoping until the end of the sprint.

This sprint is about handling the recontact on existing cases. Currently, when the practitioners have an answer to a case, the case is then closed and they can still open it for a short amount of time. With these modifications, the cases will be closed in Salesforce and the fact that it can be reopened will be handled by Salesforce. The problem before this was that practitioners reopened previous cases that had no link to their new issue. That led in losing time for our agents at the support and the practitioners.

My dev duty day

I already explained the role of the dev duty before but to sum it up your role is to :

- Keep master green (the CI checks have to be passing all day long)
- Deploy to production 3 times a day at :
  - 11 AM
  - 1 PM
  - 5 PM

Obviously, I would be completely lost if I was the only responsible to do all of this so I had a 'shadow' with me to help me that has already done the role of dev duty multiple times. The day before mine has been 'particularly bad', so the dev duty had done only one deployment during the day (the one of 11AM), so we had to do their 5PM deployment to have the modifications of the day before on production. The problem is that some new features weren't working, to handle that we can open what's called a 'DoctoCrisis', that will create a Google Meet that will be accessible by anyone in tech. A crisis is composed by the dev duty, the one that launched it (not always the dev duty), a scribe (he will take notes of what is said), an incident manager (IM) that is responsible for the decisions made during the crisis, and other people here to help with the issue. So we saw that there was a major issue so we launched a crisis, many people joined to help us fix the problem. The fact that no deployment has been done the day before has not helped as the error could be anywhere in the commits that were done from 11AM the day before. The error was weird as it was coming from inside a reject (it's used to throw exception in javascript) and identify where it comes from. After a while, we discovered that the reject was failing because an object was sent using the reject instead of an exception. Once we found out where it was coming from, we cherry-picked the commit from the production branch to remove the failing feature. Other issues were found during this crisis but this was the main one. After resolving all the problems, we continued the crisis to deploy to production everything that was supposed to be in production. We finally did the deployment of 5PM that went well. 🥳

My Starsky and Hutch

My Starsky and Hutch project is currently with Camille, a member of CODEF team. The project is to add a to do list accessible on our test website for all Doctolibers. First we did the back-end, as it was the part where I needed the most learnings. I learned how to create an engine, add basic controllers that handles all the API calls, and I learned to create / modify / update a database that will store all the todos. I also discovered some models that I linked to mine, such as accounts (to have only one account by todo). And we of course implemented security with Pundit to be sure that not anyone can modify any tasks. We did a fast front-end interface to test it using slim, slim is the historic front-end language that was used as it is really simple a really well integrated to Ruby on Rails.

Now that the back-end has fulfilled his main function, we will improve the front-end by using React and a micro front-end architecture. We did a mock-up and a quick tech scoping on this so that we know in which direction we will go.

What have I learned ?

My internship at Doctolib as a Software Engineer in back-end programming has been a transformative experience. I want to express my sincere appreciation to the talented individuals I had the privilege of working with.

Their guidance and support have shaped my understanding of back-end development and industry-standard tools. This internship has provided me with a solid foundation and hands-on experience that will benefit my future career as a Software Engineer.

I am grateful for the opportunity to work on real-world projects and collaborate with cross-functional teams. This experience has enhanced my technical skills and instilled in me a sense of adaptability and resilience.

As I embark on my future endeavors, I carry the lasting relationships and invaluable insights gained during my time at Doctolib. I am confident that these experiences will guide and inspire me in my professional journey.

I extend my sincerest gratitude to everyone at Doctolib for their support, mentorship, and the opportunities that have shaped my growth. Thank you for making this internship a remarkable and enriching chapter in my life.

Why keep me longer ?

Hi Florian and Thibaut !

I hope you're doing well. First of all, I wanted to express my thanks for the opportunity to be part of Doctolib, and more specifically part of the CUSCO team during my internship. It has been an incredible experience where I learned so much, as I've had the chance to work with an amazing team. All of this combined to make me grow as a Software Engineer.

While I worked with the CUSCO team, I've had the opportunity to learn so much about back-end programming, front-end architecture and major project management. It has been really interesting as I've been able to serve a great cause that is health. I've been through some challenges that helped me develop my coding skills and that improved my problem-solving abilities.

I've always wanted to work on projects that make a positive impact on people's lives. Being at Doctolib has granted me access to this as I was part of the healthcare domain. This experience has only made this passion grow.

I know that I still have room to grow and I am confident in my adaptability and enthusiasm to make a positive impact not only in Doctolib but also in the world. I kindly ask your support in considering me in a close future as my passion to healthcare won't have decreased. I can't wait to work again with all the team and bring my best to the table.

I'm writing to you to express my keen interest in joining Doctolib once again when my studies at Epitech will be over or even sooner, for my 5th year internship. I think that my passion for healthcare and coding combined with my willingness to grow make me able to adapt and evolve here.

Thank you both, you've made me grow professionally and personally. Thanks also for the time you've taken to read this message. I really appreciate your consideration and I'm excited to contribute again to Doctolib. I would love to discuss it further and address any questions or concerns you may have.

Best regards,
Johan Chrillesen