Becoming a full-stack developer coming from the university

Anthony Granger Villegas

Haaga-Helia University of Applied Sciences

Bachelor's Thesis

2021

Bachelor of Business Administration

Abstract

Author(s)
Anthony Granger Villegas

Degree
Bachelor of Business Administration

Report/thesis title
Becoming a full-stack developer coming from the university

Number of pages and appendix pages
33 + 7

This thesis is going to be a 10-week report of how the work experience can allow the skills of a junior developer in a Finnish start-up company progress.

For every week there will be goals and these are going to be assessed on whether they were completed or not, with the help of sources and comparisons between different approaches.

As becoming a full-stack developer is the main goal, summarizing the results will be important to assess how far a junior developer can go. Depending on the tasks, the conclusion can be succesful or unsuccesful, due to the fact that ten weeks may not be enough to consider the skills to be sufficient. It can help track the progress and direction of the desired goal.

When analyzing the results, there were many new things learned and acquired as new skills in different areas and technologies. However, as not all the parts that are necessary to become a full-stack developer are covered with these tasks, then it is important to emphasize that time is the key to reach those results. Also good work culture, as well as flexibility with the tasks time scope is important in order to fully understand the concepts.

In this report it will be proved that learning Vue.js and Django, coming from React.js and Node.js background can be achievable within ten weeks. Though they will not be fully mastered, they will lead into the right track to accomplish several tasks as a junior developer in the field.

Keywords
Start-up company, Vue.js, Django, CSS, first job

Table of contents

Table of figures

Abbreviations

AWS	Amazon Web Services
CRM	Customer Relationship Management
CSS	Cascading Style Sheets
CTO	Chief Technical Officer
CV	Curriculum Vitae
Dev Team	Development Team
Params	Parameters
SQL	Structured Query Language
UID	Unique Identifier
URL	Uniform Resource Locator

1 Introduction

During this thesis I will analyse how my skills develop during my first full-time job at a start-up company in Finland, and only coming from Haaga-Helia, without any previous work experience.

Daily I will perform tasks writing new code, testing, refactoring, fixing bugs and learning new things in general with Vue.js, Django and Docker containers.

1.1 My background

I have always been interested in the Software engineering path during my studies, for which I took courses on Java, JavaScript, React, React Native, Node.js, Relational Databases, SQL and a school project with my class where we made use of Scrum to develop a webpage in collaboration with a Finnish company. I also expanded my knowledge with several Coursera courses. I dedicated my time for job hunting and expanding my knowledge with React, JavaScript frameworks, such as Angular, and cloud services (IBM Cloud). I made my own full-stack application to promote when sending my cv and I was able to showcase my skills successfully to eventually land on a job.

1.2 Necessary skills to suit the working position

When getting recruited, I had to accomplish some small tasks before joining the company and I managed to complete most of them. I was shown a small part of the main code. This was built upon Vue.js in the frontend, and Django queries in the backend for database models, filtering, sorting and migrations. As Vue.js is a JavaScript framework and I had taken some python courses needed for understanding Django basics, I felt I could accomplish these tasks within a week and I was recruited. So the next challenge will be to learn these frameworks properly once joining the company.

1.3 Company structure

The development team consists of five people, plus me, being the new recruited. The company is growing fast. Four of my team members have worked in the company for two years since its initiation, so I am expecting to meet experienced people in the field. This tells me I would have to ask them to clarify all my doubts regarding the features, as they have developed them from the scratch. These features can include database migrations, mixins - classes made to simplify the usage of repetitive code by just importing them to another file so they are highly reusable - models and so on.

The rest of the employees within the company are the people working in the sales team, the CRM team and the CTOs. And the product owner, being an influencer herself, is responsible for the product we build and handles the communication, in most cases during the meetings, representing our feature updates. We will be using SCRUM methodology with small iterations for each feature, where I would write code, push it to a git branch and someone else would have to review it and test it in case something is not correct or does not look neat enough – code that will be hard to work on later for how it is written, so preferably it should be concise and stick to the formatting. Then this feature will be pushed to production – when it goes live and can get accessed by the customers and external users - to the real platform. If anyone from the CRM team finds something strange in the website, they will report it and then it is made a task to find the solution for a bug. Likewise, they can come up with newer ideas, that will receive feedback from others, and these can also be made a task if there is enough specification, which in return, will be useful in their daily activities when dealing with the customers.

1.4 Goals

I will evaluate how much I can get to learn Vue.js, Django and any JavaScript or CSS improvement for ten weeks during this diary. I had already been working for two months before I started writing this thesis. I will evaluate weekly how well I cope with the challenges, if I do not ask for help, and compare the sources I utilize. This will lead to understand what I worked on and assess whether I accomplish my goals or not. In the end, I will conclude based on my findings, how far I am to becoming a full-stack developer, or if I am heading towards that direction and what impediments there will be.

2 Starting point

2.1 Daily duties

As a developer at the start-up company, I will be working at the office or remotely. We are given a personal computer, and in my case I got a MacBook Pro. At the office we share the same floor with the sales team and the CRM team. We have rooms for weekly meetings and small rooms to make phone calls and to avoid disturbing other colleagues. If I choose to work from home, then I have to communicate with the developing team through Discord channels through voice or text messages, share screens, to communicate about any developments throughout the day. We are using SCRUM and agile methodologies, therefore I will have daily meetings where I will have to explain what I have done, what I will do and what has been difficult. Then my working day starts and I will meet different challenges depending on the day or the sprint – that could be a time period of up to two weeks to develop a feature, split in smaller tasks with constant feedback, not just from the dev team but also from the rest of the employees of the company. This feedback usually is communicated on Slack as they will be notified to test the application and comment on what could change or improve.

As a beginner, I am given smaller tasks that are communicated through the Slack channels, Viima - that will be the platform for suggestions for new ideas - and Shortcut, where I can create Epics and stories regarding a small feature, or bug-fixing tasks. As a worker, I should be always reachable through these channels and be able to create my own stories and time estimations – usually flexible depending on the feature. If I am doing a solo feature, then this one should be in an Epic well described with its stories and tasks within those stories, so everyone else from the dev team can see what I am doing to be able to track my progress.

2.2 The start-up company

We use Microsoft Azure DevOps services to host the builds and create pipelines for testing, staging - that would be a way to try the application - and production which is the final product visible to the outside world. We also use, as I mentioned, Vue.js for frontend development and Django for backend and to handle the database queries. Altogether is run in Docker containers.

As I progress and get more familiar with the features, I will not be working too close to someone supervising me, as it is at the beginning. So I will have to have my own initiative and I should be able to ask the right questions to not take so much someone else’s time.

This company develops a platform for influencers and content-oriented marketing, so I will have to be familiar with what campaigns are. These can be influencer campaigns, content campaigns. I will work also with organizations and will learn how to tell them apart from creators, that are actually influencers.

Organizations are our direct clients and these would create, through our platform, a campaign which then will be handled by our CRM team members to decide which influencer applying to those campaigns is the most suitable, based on their applications, also handled through the platform. Once selected, they can create content for those campaigns. Mostly they will upload content, promoting our client's brand, and they will earn money in compensation as though they were doing a job of a professional model. These influencers from Instagram will have to have plenty of followers.

This whole concept is doing well as a whole in Finland, and the company already is looking to expand to Germany and other markets abroad.

After learning how things work, I should be able to understand what would be the point of any feature. For example, if a staff user, who could be anyone from the dev team, sales team or CRM team using the platform, can get access to pretty much everything, they could communicate with clients or simply handle the applications and campaigns. And of course, answer to the influencers through chats within the platform. Whereas, influencers and organization users can only get access to things they would be interested in.

To keep everyone up-to-date, we will have weekly meetings on every Tuesday to let everyone know from different teams, what is happening with the sales, marketing, and what new features are being developed or already exist in the platform and how to use them.

And there is a monthly meeting, we as a dev team can have a possibility to have our say of what we like or would like to change, as well as what we do not enjoy or not like about the way we communicate or handle things. We can also plan ahead of what newer features we will produce and how we would work through them, giving us the possibility to, for example, decide if we would like to work more independently from the rest of the team if our schedule will not be the same as theirs.

3 Diary

I will start writing a 10-week long diary related to my experience in the company. This will include a weekly analysis and comments in general of what I experience throughout my daily tasks. For every week I will write down goals and I will analyse how I deal with them in the end.

3.1 Week 1

The goals of the week are to fix the Uppy library delete items bug. Work on a whole page layout using grid layout. Make sure the mobile view for the page looks good. Create a mansonry image layout for a content library page. And fix an asynchronous call when is resolved or not resolved.

3.1.1 Monday 27.9

Today in the morning I started by asking help on how to fix a bug to delete items from a drop box using Uppy library. A drop box is where you upload items from your computer instantly by dragging and dropping them in the browser. This is the way we handle the file uploads. However, it has been buggy for a while. One of the issues was that the component was rendering multiple times for same operation because it was recreating a new id every time a delete, file-upload event was triggered. This caused to show files that should not be there. For example, I upload a video named video.mp4. Then I move forward and return to the page where the drop box is and try to delete it from there. If I change my mind, this gets erased but comes back again after a few seconds for reloading again. This was confusing and I first though it may have been an error coming from the backend. But after testing it from the browser checking the network and console, the issue was found to be the id being recreated.

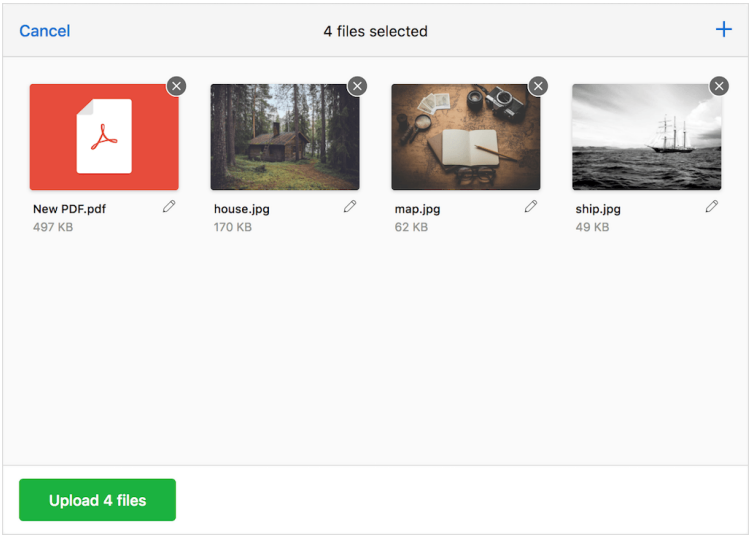

Figure 1. Uppy library drop box

Eventually, after fixing it, I pushed it to production so a new moodboard view would display pictures and video when creating a post. The moodboard items are the pictures and videos that an organization user uploads to display for a campaign page, that will be seen by the influencers later and understand the concept and visual idea of what the campaign is about. These are columns of images and/or videos on the bottom right of the page.

After the task was done, I continued to work on a Presets page that only requires frontend code as this may not change often, in other words, this page will remain mostly static. According to this criteria, then it is fine to handle everything statically and not with new models being created in the backend. So all the pictures will be uploaded to AWS S3 buckets and then get retrieved from there. Same with the instructions video on how to download and user the Presets later.

3.1.2 **Tuesday 28.9**

Today I decided to work from home so I was mostly communicating with the dev team on Discord. My main task was to change the whole layout of the Presets page. Although the designer had left some suggestions on how to handle it, that for me was lost in a Slack message. My boss asked me to fix indentations so there would be a section alongside the main content. The section should have a fixed position, so when scrolling down the page the section will always stay visible.

3.1.3 **Wednesday 29.9**

Today I continued to work on the Presets page with which I had some troubles with the grid as I had to fix the main picture of the header. The header picture should include three pictures on top and two underneath stuck together. That was only accomplished by using grid as display and not flex. With Grid layout I can then always have three pictures on top and two underneath with the same width on each row easily.

Figure 2. Presets grid layout example

Then after this one task was done, I continued with the rest of the details of the page for mobile view and desktop views. Resizing the screen has to have good indentations for any screen size, making the page look professional as it was intended.

3.1.4 **Thursday 30.9**

Today I finally got to deploy my first Presets page. As soon as I started testing it in production, I realized that a video was not working on Google Chrome on mobile screens. It took me time to realize why it was not displaying when opening the page. So I first changed the file name that had a blank space instead of underscores between words, coming from the AWS S3 bucket under that name. That could have been the cause of the problem. Eventually after testing my phone with the computer, I realized there was no error in the browser's console, so it had to be the file extension. It was mp4, but somehow the size and type was not working properly on the browser. The bug was fixed by implementing another video format from mp4 to m4v.

After that was fixed, one colleague from the CRM team told me that the scrolling part of the page was not appealing. As the section text is too long, one would have to scroll all the main content to the bottom first to then be able to scroll the section part that is located right next to it. Then I decided to ask the designer for recommendations and he answered that it was best to just get rid of the "fixed position" of the section so both parts of the page would scroll down together.

By the end of the day, when the page was finally in production, I was introduced another task. This task required adding filters to a new page being developed, and a new grid layout so pictures from different sizes would still show completely and next to each other, only keeping the same height, and not the same width necessarily. For that I was recommended to use what is known as Masonry layout with just pure CSS. And after checking some sources on how to tackle this problem, I realized this is known as horizontal masonry layout from the source "Approaches for a CSS Masonry Layout" (Chris Coyier, 2019).

3.1.5 **Friday 1.10**

Today I started the day with an introduction to a new task. My colleague explained the reason for the task on a whiteboard. The task goal was to use a component for another page and create different filters from the backend for it. The filters should now allow different organizations to view all their content in one single place and save them from having to go through their campaigns one-by-one to see the contents that creators uploaded in each one them.

So I started by creating a new page component with a new route leading there and I included the already existent component using grid that would display the contents related to the user's organization with pagination, in other words, the first call to the server will fetch the first twenty-five results, and as the end-user scrolls down the page, the rest of the results will appear, thus saving resources if all the available results are too many to load at once.

Later on, in one of the Slack channels, someone from the CRM team posted that an influencer had an issue with one server call that was not resolving and was showing the call took too long to resolve to load an Instagram account. When looking at the bug I noticed that I had left a promise without resolving properly. The issue with the promises and the Promise.race function is that once one promise completes not necessarily resolves if I do not specify explicitly in the "then" and "catch" methods when it should resolve.

```
const fetchStats = new Promise(resolve =>
  CreatorService.stats
    .update(this.$store.getters.getTokenData.uid, {
      instagram_username: this.instagramUsername,
    })
    .then(() => resolve("resolved"))
    .catch(error => {
      console.log(error);
      resolve("error");
    })
);
// The promise should return something before 4 minutes, otherwise it will be conside
const result = await timeoutPromise(fetchStats, 240000);

if (result == "resolved") {
  this.data.loading = false;
  this.$emit("onboardingComplete");
}
if (result == "error") {
  this.data.message = this.$t("Onboarding.FetchingError");
}
if (result === false) {
  this.data.message = this.$t("Onboarding.FetchingTimeOut");
}
```

Figure 3. Promise resolve example

When I managed to solve that, then the bug was considered fixed and I learned that promises do not work as I thought they did.

After that I continued with my tasks related the new content library page, and by the end of my working schedule I managed to display all the results from the endpoint I had to use. So for today I felt I fulfilled what was expected from me.

3.1.6 **Weekly Analysis**

This week felt a bit stressful to me as I had to ask for advice for every task. I managed to make my first migration with Django and pushed it to production with help. This process typically paralyzes the web page while the process is being completed, which means if someone tries to access the page, this will tell the end-user that is “under maintenance” if the migrations are not included yet. Then I worked on a bug that I created and quickly solved. I implemented successfully the mobile view and fixed the layout for the Presets page which was pushed to production.

This week helped me understand better how promises work, which I thought were handled by a “then” and “catch” function and never thought about also explicitly putting them as “resolved” when using Promise.race.

3.2 Week 2

The goals of this week are to figure how to use Vuex state management with getters, mutations and actions. Apply dynamic classes in addition to static classes for styles. Fix some minor bugs in the application that are in production and found by the CRM team. Refactor old code.

3.2.1 **Monday 4.10**

Today I continued with the frontend tasks where I had to create a horizontal masonry style for all the image assets in the content library page. Some of these should look wider than the others. This task took me quite some time to get it to work visually as I also had to modify the styles of another child component which was not letting me accomplish what I wanted.
After that, I had to figure out how to fetch all the results from another end point which was made by my teammate and then I could display more results.
Then it took me most of the day to connect states from Vuex store and all the actions and mutations linked to the previous ones that had to be replaced into two different components. These have paginations that only would work with these states, and only this way I could display more than 25 results which are the first ones fetched. An observer here is also applied to trigger this actions.

All in all, today I again had it difficult but I succeeded with what was expected from me. Tomorrow I will continue with other end points that were created by the end of the day by my teammate.

3.2.2 **Tuesday 5.10**

Today I had a weekly meeting to talk about the latest updates regarding the chat rework feature and my presets page that is now in production.

After that, I continued to work on including other end points and styles for the filter. I encountered some problems with the Vuex state management to retrieve certain data. I was using UIDs, when it did not need them, and then I had to start creating a new way to apply filters and replace the old one. So I rebased the branches to also include changes made by my team mate working on the same tasks. And later I would move on with more individual tasks in between.

3.2.3 **Wednesday 6.10**

Today I had a task to fix the aspect ratio for the assets display with the masonry style. In order to make it more dynamic, I had to use a dynamic class written like as ":class" instead of just "class".

```
<div
  v-if="data.initialized"
  class="columns"
  style="position: relative"
  :class="{ 'loading-opacity': data.loading }"
>
  <div
    class="column column__filter"
    :class="{ 'column__filter--hidden': !data.showFilters }"
    @mouseover.stop="setBodyOverflow({ value: 'hidden' })"
    @mouseleave.stop="setBodyOverflow({ value: 'auto' })"
  >
```

Figure 4. Dynamic Class

Then to that I added a function which would take the meta data related to width and height from each asset as it loops through it. Then I got stuck with this task wondering what to return from this function. After a while learning more about dynamic classes, I found the source "How to Dynamically Add a Class Name in Vue" (Michael Thiessen 2018). As I was just mostly familiar with static ones, I finally learned that the function can return a class name and this one can be used in the styles to then define what happens for example, if the image is wide then it should look wide otherwise narrow.

Because I worked from home, today I did not have any other tasks and let's see what the feedback will be during the next day.

3.2.4 **Thursday 7.10**

Today I started by checking if the solution I applied yesterday was correct. My more experienced teammate approved it. However, I still think it should be including more aspects ratios for different images in the future.

Then I looked into a bug that was related to one of the firsts tasks assigned to me. The user profile allows users to add, delete and select primary emails. In this case, as this component was before updated by someone else and this part was made a child of the main component, the newer component lost its styles and functionality which made it difficult for users to accomplish the intended purpose. So I looked through the code and adjusted it to the children component to make it work again and this was pushed to production.

Later on, I continued working on styles for the library. One style fix was to make the preview video fit the space allocated in each item of the gallery, in case it increases the size and goes out of borders when hovering on it, and the main borders will ensure that it will remain within it and the rest will disappear. This was accomplished with the overflow property in CSS set to hidden in the parent class. In the end, I planned how to create a new way to apply filters to this page which will be left for the next working day.

3.2.5 **Friday 8.10**

Today I continued to understand more about the filters and I also worked on refactoring 2-year-old code, which was complex to understand as many things have to connect in different places and considering them all would keep me looking at five different files altogether.

Then I worked on some small feature to ask the staff to confirm actions before triggering them. This would be a modal that opens on top of the screen asking the user to confirm an action for safety in case they happened to click on something and can’t undo it later.

Later, I had to adjust translations ready to be translated by the CRM team so I just send the links to a Slack channel where they be able to add German, Swedish and Finnish language version of the content requested.

And finally, I had to rebase to master branch the feature I worked on related to the content library feature, which is always hard for me, to do one-by-one, add the files and use “git rebase –continue” until it is finally without conflicts. Then all this would be forced push with lease to the git branch.

Today in general I had learned more technical stuff that is always important to consider for the future.

3.2.6 Weekly Analysis

This week I learned to cope with old code alone. It has been challenging to refactor old code and especially when data is coming globally from the Vuex store component and not from the components where I use them. Likewise, I have learned more about Vuex store and when to use getters, actions and mutations to manage the states. I learned a new way to display a grid made with flex and using dynamic classes, which checks for a condition to use it, in other words, checks if the width is bigger than the height of the item to display the assets wider or thinner otherwise. Then I managed to fix a bug successfully that was caused by a component that was refactored previously. As I knew where the issue was, it did not take me long to fix that bug due to some data was not being passed correctly to the child component that caused the issue.

This week I have understood more about how stories are being managed, as one should not take longer than 2 weeks and the tasks are divided into smaller parts so I can accomplish at least one of them per day. In general, no one is pressing me to hurry. For my team it only matters the code quality rather than the speed it is being implemented without considering the importance of making it maintainable later

3.3 Week 3

The goals of this week are to make the filters linkable. Make sure the asynchronous calls would not collide and create duplicate values. Apply paginations for the filters. Refactor old code. Apply loading spinners when requests are not done yet.

3.3.1 Monday 11.10

Today I had more insight on what to do with the filters and I was also told to fetch another endpoint for the filters. So I spent quite some time thinking how to organize the filters, as these have to be placed in the URL, as well as the params in the request, so they will be considered “linkable” if when refreshing the page with a specific URL including will automatically use the selected ones. The filters can be chosen mostly with check boxes and there can be plenty of them. Therefore, it makes sense to place every UID of these filter values in arrays, so these can be looped through on every request.

3.3.2 **Tuesday 12.10**

Today I got a bit stuck with paginations and filters. The filters seem to work fine now but somehow they keep loading information when the observer triggers, creating duplicates, something I need to work on longer to figure out the cause of it. I also finally managed to add a “next” pagination logic to all the results included as filter parameters. In other words, a filter that lists all the products contains hundreds of different values that can be displayed, but this has a pagination, only displaying the five results for better performance. Then by clicking on a button “show more”, it is possible to display the next five results appended to the previous ones and so forth. As the end-user pleases, can continuously browse through all the results, and make use of these product names as filters to then display different items from the library.

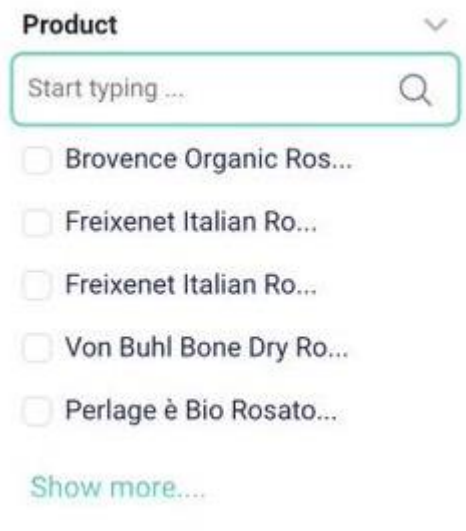

Figure 5. Paginated product filter example

Later I continued to refactor the 2-year-old code and is not an easy task by all means. So I got stuck for the whole day planning how I could make the new code more concise and maintainable.

After all, today I accomplished what I wanted and tomorrow I will continue refactoring code from the Vuex store.

3.3.3 **Wednesday 13.10**

Today I started by correcting multiple things in the looks of the page, such as displaying loading spinners when requests are still loading and fixing the multiple asynchronous calls that are, for some reason being triggered and creating duplicates.

Again I looked at all the code I had to refactor without much help, as my colleague working with me on this task was on a sick leave. In overall I managed to make the filters fulfil their purpose but there is still more to add on the next days.

3.3.4 **Thursday 14.10**

Today I fixed the issue I had with two simultaneous asynchronous server calls triggered together and causing duplicates. I also managed to understand the main reason of pagination and observers. However, after refactoring a whole component and calling it from another page, the issue with the repetitive calls to the same endpoint started occurring again. After spending thoroughly refactoring every single part of the code, I could not manage by the end of the day to find the issue with the pagination being triggered again after selecting a filter, overriding the filter.

3.3.5 **Friday 15.10**

Today I had to again start the day by figuring out what could be triggering multiple calls when using the filters. I realized that it was best to apply conditions for when the mutations occur in the store and these can be called then in the component, so the pagination is not triggered. It took me time to fix everything till the point that in the console there is finally no more duplicated keys shown as errors. Then I managed to successfully apply a linkable filter. In other words, when using a link with parameters, this will be used recognized by the component to trigger the filters applied, also showing the check boxes as checked. The same feature was applied for the ordering of those results. After doing a JavaScript tricks with arrays and null values for the filters that do not exist, I was able to accomplish the filter feature. When refreshing the page, from the link the filters should be applied and this can be considered as done.

3.3.6 **Weekly analysis**

This week was a bit challenging for me as I only worked from home. I did ask questions when needed from our communication channel that is Discord. I read the code from different components, pagination, mixins, store and services for connecting the new endpoints created in the backend. I learned more about paginations, why it is useful to have them and when make use of them with the help of observers when scrolling down, in other words, when the browser reaches certain position, this triggers. This creates a never-ending scroll with new results which is always a nice feature to have in any serious application.

In addition, I have understood more now when to use a state management, in this case Vuex, using state, actions, getters and mutations. I took advantage of them to handle

complex paginated results and when to mix them with others in order to create filters and sorted results. I did have problems when multiple calls were being triggered instead of just one that would end up overriding previous results. But these can be handled, even if the calls are asynchronous, by not allowing the next call to trigger until the previous one had completed, which creates a nicer user experience as the end-user would not create any error by, for example, selecting multiple filters rapidly, or making a link that did not actually triggered any filter. The filters now can be used directly from the link address parameters, which is something I never really tried before, and even though it felt a bit scary not being able to accomplish it, I still managed and that is something I can consider a success.

Sometimes at work when one is going through the trial period may get scared of asking too much to not seem incompetent or too weak compared to other older and experienced colleagues. In this work place, the main dynamic is to always ask when help is needed, or just split the tasks into smaller ones and do one at a time, to not drive down morale and to just focus properly on one feature. This is a good approach in my opinion and can be one feel more sure of his own skills to face newer and more complex challenges in the future.

3.4 Week 4

The goals of this week are to fix visual bugs and continue to apply more filters to the content library page.

3.4.1 Monday 18.10

Today I helped resolve a bug and completed other additional filters for the content library page that we are currently developing. The bug consists of videos not being processed properly in the frontend. However, the issue lied in the database and migrations. We created a data field named assets that would take both, images and videos. These videos when were migrated somehow got their URL fields erased, leaving them empty where they should have been displayed from the page. As I worked on this feature before, I had to help figure out where the main issue occurred, so I asked for help and a more advanced colleague could patch them back to where they were before and also helped improve a link created after uploading them, by also adding the UID of the asset and not just the one from the campaign.

Later I completed more filters. These are used when you select an item from the gallery and can be filtered by “selected by user”, “anyone selected” or “no one selected”. Also, these can be now added as favourites by the user and be filtered from the rest the same way.

3.4.2 **Friday 22.10**

Because I had a child recently, I had to be absent at work from Tuesday to Thursday. So today I am back at work and I was assigned to change another end point from where the data is fetched and replace it with new one. This one considers all the purchased contents and people who bought it in order to be used in the filter. As videos and images can be purchased, this is an important feature to keep up with the new gallery. When impersonating a staff member, the results can be never-ending, so this new endpoint also includes a pagination. I also was assigned a task to add a link related to a specific content, redirecting the end-user to the campaign related to it, which can be beneficial when dealing with all the results. After starting work a bit late today and tired from the hospital visit, I decided to continue later and rest from the several nights spent with lack of sleep.

3.4.3 **Weekly analysis**

As this week I spent it at the hospital for the birth of my child, I only can say I worked two days and I mostly solved bugs and continued with the content library page with filters. So all in all, there is no much to say other than learning when migrations can create errors in video or image assets if something was not written right.

3.5 Week 5

The goals of this week are to learn how to use serializers in Django and when to add fields in it or not, always considering what is good for performance. Include more filters and trigger the check boxes with boolean values when they are checked or not.

3.5.1 **Monday 25.10**

Today I had to continue changing more endpoints. This one is a post request and I had to include an organization UID for the request instead of a campaign UID, as it is a content library page and not the former campaign page. This took me quite some time to apply as I tried all possible ways to reproduce any error and so I could prevent it in production. As organizations can be many to select from, I made sure the first selected organization is the first one to appear from the list, avoiding nulls and unselected values. This option is only visible from the content library and not from the campaign component page which use the same component. Then I included a direct link for assets that display in the library, connecting the user to the campaign they are related to. For now, I have applied a frontend hack to get the campaign UID from the assets to redirect the user to the campaign related to such UID, as this one has not been included in the serializer from the backend side, and this task so far belongs to another colleague. This hack was simply

made with JavaScript, cutting off an already existent URL to obtain only the part where the UID appears in it.

3.5.2 **Tuesday 26.10**

Today I could take my own initiative, due to my colleague missing the day for a holiday, to simplify things for myself. I started by changing the backend fields, related to the campaign UID mentioned yesterday. I added that field into the serializer so the hack in the frontend can be deleted and the code can be minimized. Now I can directly get the campaign UID and add it to the link redirecting to the campaign component from the content library in a proper way.

```
class CampaignBriefSerializer(serializers.ModelSerializer):
    required_hashtags = serializers.SerializerMethodField()
    optional_hashtags = serializers.SerializerMethodField()
    required_usertags = serializers.SerializerMethodField()
    requirements = serializers.SerializerMethodField()
    organization_uid = serializers.CharField(source="organization.uid")
    products = serializers.SerializerMethodField()
    image = serializers.CharField(read_only=True)
    content_properties = serializers.SerializerMethodField()
    moodboard_items = serializers.SerializerMethodField()

    class Meta:
        model = Campaign
        fields = (
            "uid",
            "description",
            "terms",
            "start",
            "end",
            "application_start",
            "application_end",
            "requirements",
            "required_hashtags",
            "required_usertags",
            "optional_hashtags",
            "products",
            "moodboard_items",
            "organization_uid",
            "image",
            "name",
            "content_properties",
        )
```

Figure 6. Django serializer class example

Then I decided to improve the filters from products and categories. This time, I also added fields in the backend related to whether they are checked or not, and migrations as the structure of the model changed (this happens in Django). As these can display many results, it can be easier to handle it this way to check whether certain item is clicked or not. For example, when I choose a filter, this one will be shown as a checked box, either when the click event is triggered, or by using a link that includes that filter. In both cases,

the checkbox will be checked if certain filters are clicked or simply included in the link that gets used when refreshing the page. This is a really cool feature I could solve today and took me time to test but I was satisfied after several tries with no console errors or wrong results compared to the API tests. The API is tested directly with Django on the browser by just calling the endpoint and add parameters to it.

3.5.3 **Wednesday 27.10**

Today I worked on another endpoint for campaign UIDs, and I spent a while modifying the components to adjust the new filter. I was also asked to change the fix from the backend and only use the boolean field is_selected in the frontend by mapping the results with it for each product, category and campaign.

By the end of the day, I encountered a bug in the gallery component, that once opened, it should display an image or video by clicking on the displayed asset, but for some reason, it does not seem to work properly when refreshing the page. So I will have to look at it tomorrow again to see what the underlying issue is.

3.5.4 **Thursday 28.10**

Today I worked on another data fetching task from another endpoint. These are five types, related to user selection. They are favourites, purchased and selected items before downloading or purchasing. This took me the whole day to test them and organize, clean and delete code to adjust it to the filters. I also created a method to be reusable when handling paginations. Today I got help from a colleague to solve an issue with the gallery opening when clicking on an item. This was solved by deleting a check when the index was null, and that was causing the problem whenever the page refreshes by not showing any assets unless you retry again, making it seem tedious to use.

3.5.5 **Friday 29.10**

Today I had a shorter day. I started by making sure all the code is clean from unused variables and content, and that is concise. Then I had to make a git pull –-rebase, to pull changes my colleague pushed into the same branch that could create conflict. Then I pushed the changes of the filters to the corresponding GitHub branch and I was given newer tasks. In the process we had to attend an online demonstration by another advance colleague on how to accomplish outlier-detection with Python and differential calculus applied in the same context. After that, I was done for the day as I had to receive visitors at home. So the next sprint will be started with smaller tasks in scope to be accomplished compared to what the content library feature took.

3.5.6 Weekly analysis

This week I managed to finish most of the features requested in frontend for filters, such as paginated filters, adding the filter values to the link as parameters to make them linkable. I worked hard on re-writing the old code from another component where these filters were used before and will be used now. I learned that is best to apply the is_selected field only in the frontend if this is just about a boolean that will be set as false by default to check whether the filter is applied or not. Because including it in the serializer and be only modified from the frontend will have a huge impact in performance as the source “Skipping Django serialization of rarely changing objects” suggests, due to the fact that serializing unchanged data is wasteful. I did apply the field into an object but only in the frontend without serializing that specific field. This week I feel that I fulfilled everything related to this sprint and if changes need to apply, they will be discussed soon.

3.6 Week 6

The goal for this week is to fix bugs, add confirmation modals for the users before continuing with their actions. Fix a calendar deadline. Make use of mixins. Work on a new page layout following the design in Figma.

3.6.1 Monday 2.11

Today I worked on two features, one to make the staff press a confirmation box before withdrawing money from a client. The other feature I worked on was to add a different pop-up message for an organization user to notify that the application period has not started yet. Somehow still tells the creators that it is visible when is not supposed to before the date. For this, I added another check from the backend to create the option “published_not_started”, that means the campaign is accepted and published and the period has not started yet to apply.

3.6.2 Tuesday 3.11

Today I worked on three features. One was to add a fill-up message and image to a campaign content tab, that if does not have anything uploaded yet by an influencer, it would not show anything else other than an upload button. Then the another task was to add a new filter backend to allow a staff member to find a user by the Instagram username, and in addition to the already existing names or emails that can be searched from. And the other one was about the calendar not returning deadlines for a user that has not yet been accepted in a campaign. This still showed the deadlines when the user had just applied and was waiting for a response. These deadlines are schedule in the calendar

for when is the last day the user should upload content to campaigns they are taking part of.

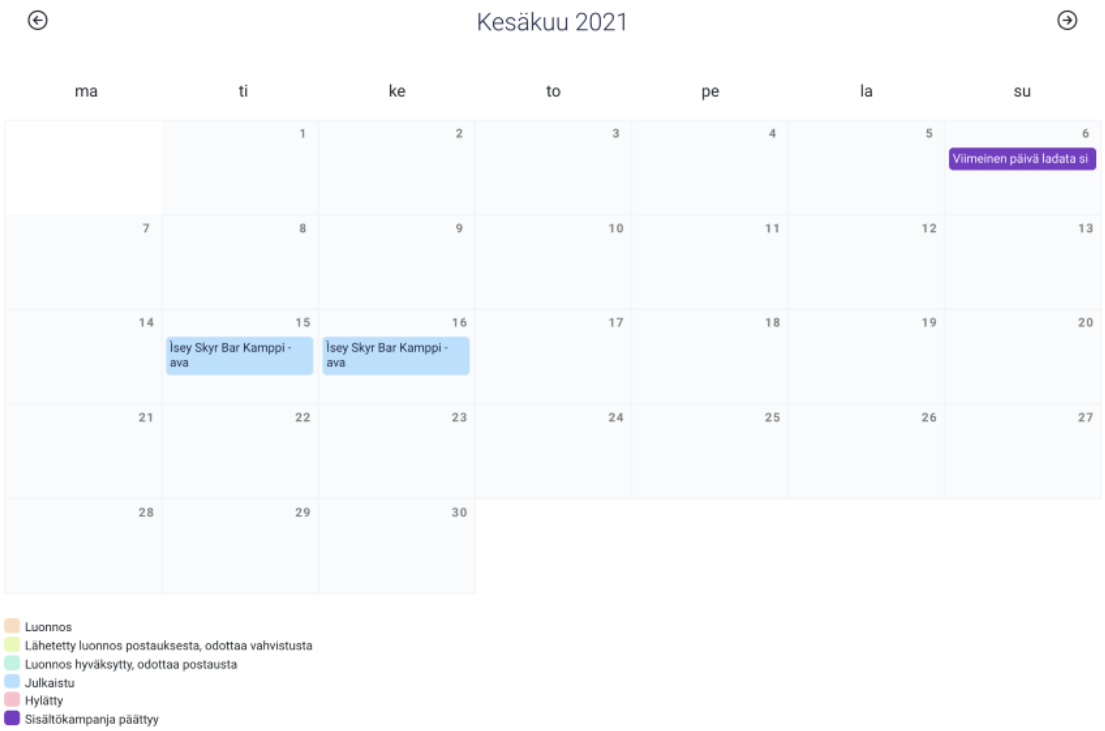

Figure 7. Calendar example

After a while, I had to update the database by fetching the latest data. This took a while to process and finish. Then after that I was able to impersonate users, which is something that we have as a feature for staff users, but when some new Django migrations are pushed to production and the database get messy locally, it is good practice to fetch the latest database version to continue to work without problems.

After I was done with that, I had noticed that the task of the calendar needs a backend fix as well. For this I asked for help from a more advanced colleague, basically my boss, about the feature I made yesterday regarding the pop-up message shown different when in a campaign the influencers should not see them before the application process has started. He said the tasks was a bit underspecified so it was left in the branch while waiting for the response from the CRM team, because the influencers actually can see the campaigns before the application time is happening, which is misleading from what they meant when this task was created in Shortcut.

3.6.3 **Wednesday 4.11**

Today I worked on two features requiring me to handle the backend with Django. For the first task I used an already existing mixin created by my colleagues some time before I joined the company. The mixins are “classes that contain objects and properties that can

be applied to other objects" (Brock Herion 2021). I added it into the filter to make sure the results will not display deadlines in the calendar if the creator/influencer has not been accepted in the campaign yet.

The second feature was about using a custom filter backend with Django. This feature was a bit more complex, I had to go through the Django documentation to be sure about what I was doing. I also asked for help. I eventually accomplished this task by adding an additional way of finding a user to impersonate, by searching them from their Instagram username.
Today I also finally could feel more sure when using the git amend command whenever I wanted to push a quick change without changing the commit's name, in case I erase an unwanted line or parenthesis.

3.6.4 **Thursday 5.11**

Today I started the day with a meeting with one colleague to introduce me another task. In this one I will be working alone and it will be mostly worked with the frontend. It is a subscriptions page and I had to estimate how long it would take. I was told to not have pressure as the design and quality is prioritized rather than speed. so my first estimation was one week. After the meeting was over, I created the story myself in Shortcut. Then I worked on the layout that I had to copy from the one already made in Figma by the UX-designer. This one has to be consistent with the text, colours and indentations in desktop and mobile screens. After writing an organized designed, I decided to continue with the rest on the next day which consists on creating side menus and modals.

3.6.5 **Friday 6.11**

Today I worked on the modals and the side menu that opens up when changing the credits plan option. This gives a slider selector of number of credits to buy, depending on each subscription with prices and discounts to encourage the user to buy more. As this requires to organize the code to make it more consistent, and it is pretty static, as not much data is involved from the backend or database other than the existing subscription from the user, it can be time-consuming to work on the design itself, so I spent most of the day making sure everything looks good.

3.6.6 **Weekly analysis**

This week I had to complete small tasks from the previous week. For many I was asked to write the code a bit different to accomplish the same and one was just not considered because it did not have enough specifications from the person who requested. I also finally managed to understand some git useful code to amend the commits in case I have

to change a small thing without writing another commit. Then by the end of the week, I was introduced another task, to build a full page by myself in a feature that only requires frontend and some backend for communicating the user with a new subscriptions Slack channel. This week felt more chilled in regards of the features and not much pressure from anyone to finish them quickly.

3.7 Week 7

The goal of the week is to follow-up the design of the suscriptions page and implement the Slack message functionality after the user confirms a subscription plan or credit purchase.

3.7.1 Monday 8.11

Today I continued to work on the layout of the page. As it is static, I tried to create it with less code and organized so it is maintainable, even though it has repetitive code that may not need to be split into smaller components if things will not change much. I have spent the whole day just trying to make it responsive for mobile screens so I can present it tomorrow to the requester and continue to add more after that. Today I basically refactored my own code to leave it ready for when the functionality of the page needs to be worked on.

3.7.2 Tuesday 9.11

Today I completely re-organized the main component into smaller ones for modal and I created a new component for the repetitive Price Card, so it is more readable and each component using it will have less code written by importing it. This process has the this.$emit(“functionName”) feature to access methods of the children component from the parent component, as well as the props, to pass objects, strings, booleans or arrays from the parent component into the child component.

```
cancel() {
  if (!this.subscription.is_canceled) {
    this.cancelSubscription({
      organizationUid: this.subscription.organization.uid,
      subscriptionUid: this.subscription.uid,
    });
    this.$emit("cancelSubscription");
  }
},
```

Figure 8. Emit custom event example

This way, now the modals can be used anywhere by just importing them and using their functions and props. I also improved the mobile view which is also hard to accomplish as components completely re-organize and even change from columns to rows or vice versa.

Figure 9. Event used in imported component

3.7.3 **Wednesday 10.11**

Today I had to discuss how to handle the main feature of the page with my colleagues. This feature is about sending messages from the application to a Slack channel. This occurs when the user buys new credits or subscriptions plans, so the staff can handle them later. As this feature required some thought on how to deal with the backend, I went through the existing code and took examples so I did manage it by the end of the day. I required to test with python “logging” library and to pass parameters to the message. Then the frontend should also communicate with that endpoint and be able to make the post request that will trigger the message on Slack.

I also decided to improve the slidebar to select the credits as this one was not well-implemented and that is how I spent most of the day, passing values and making small mathematical calculations for the price combinations.

3.7.4 **Thursday 11.11**

Today I worked on adding a pop-up message displayed to the user when a new invoice is created with both, credits and subscription plan changes. I improved the message for credits and subscriptions, having different messages for each one. So the Slack channel has the right information without null values if some do not return the same type of data. And I tried to figure out how to create a table displaying all the plans and credits that each user would have as a history. This feature I decided that I will continue tomorrow as well as fixing a bit the code, so it can be tested on Monday.

3.7.5 **Friday 12.11**

Today I worked on a few mobile changes regarding the layout and translations that have “bold” and “link” tags. The latter that replaces the standard “a” tag in v-html. Then I added a view details feature for a table, that takes the user to another component displaying more details of a certain purchase. This also includes a back button to return to the main subscriptions page.

3.7.6 **Weekly analysis**

This week was about learning how to follow the layout from a design made by a designer and stick to the details in every possible way. I also learned how to send Slack messages from the web-page to a Slack channel when the user confirms purchase options. This feature is only tested in development, so the messages go to a bottest channel and not to the real one. As I mostly used CSS and styles in general, the challenge was how to make everything look exactly the same as the design. Even when the screen changes to mobile, for which I had to come up with something myself, as this was not included in the design.

3.8 Week 8

The goal of the week is to change the page layout to follow every detail of the design, including modals and side bars. Also apply a new numeric selector for credits and implement backend code for Slack messages to make it work.

3.8.1 **Monday 15.11**

Today I got a review from the CRM team regarding the page I created, and also from the UX designer. I got a few changes requested, mostly the layout and functionality on how to purchase credits and subscriptions, so these changes that were on display only on “staging” before production, will have to keep being worked on. I had to continue with the layout again and change the slide bar idea for a number input to make it easier for the end-user to select new credits. This is the first general review I get for my code from most of the company where many people were involved, so I can look at it with positive eyes as no one really has gotten me to feel pressured or scolded for something that did not look good and, quite the contrary, I was motivated to take my time to do things the right way. In this case I agree with the article “Proof That Positive Work Cultures Are More Productive” (Emma Seppälä and Kim Cameron, 2015) which suggests that stress or cut-throat environment is bad for performance in the long-term, whereas the positive organizational psychology can make one feel more valued and engaged.

3.8.2 **Tuesday 16.11**

Today I worked on changing discrepancies between the design and my implementation and on the new numeric input feature to replace the slide bar selector for credits. As there is a lot of work to do to change class names, structure, colours, font-sizes, this took me the whole day to test on different screens, including zoomed-in screens to make sure that nothing looks bad in big screens. For simplifying the paddings and margins I used the “Axiomatic CSS and Lobotomized Owls” source (Heydon Pickering, 2014).

3.8.3 **Wednesday 17.11**

I worked today quite long again fixing the slider selector that changed to a numeric selector, the layout, and making conditions for when plans are active or not. I also added a cancel feature for the active subscription and I added a box dialog to prompt the user to accept or cancel if unsure about continuing with the operation.

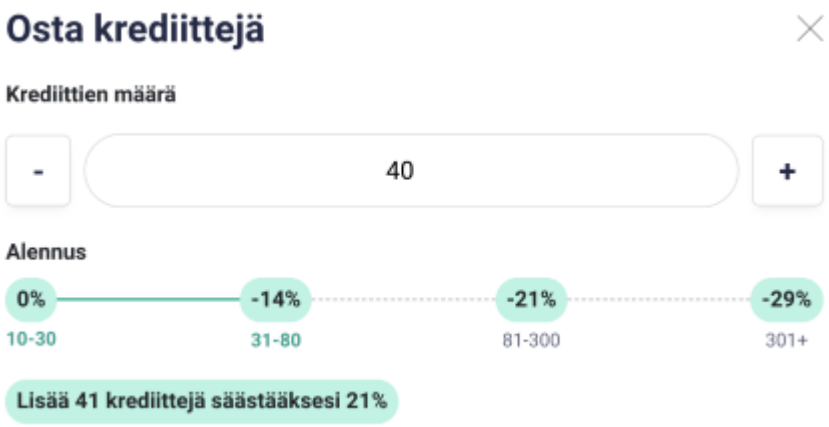

Figure 10. Numeric credit selector example

This took me again quite long but after a while testing, I feel like I am almost done with this feature and page in general but there is a lot of retouching to do to the code.

3.8.4 **Thursday 18.11**

Today I fixed some strange string values that the numeric input gives when pressing the inner up and down arrows to increase or decrease the number of credits. This was handled with the @input event and this value is obtained to be always converted into a number. Then I continued for the rest of the day handling the backend part of the feature, to send the message to Slack when the user requests to buy credits or a new subscription. For this, I had to handle Django queries to test with the shell and then log them with python. Only in production we will use the real channel that will be used by the CRM team to handle the purchase requests of the clients.

3.8.5 **Friday 19.11**

Today I had some meeting with my dev team. We mostly spoke about how the future of the company will be in regards of employees, and the reason of some leaving the company for relocation or position changes.

Then, after that I checked that the code was ok and left it in the branch. As I did not get anymore feedback from my team, I completed the day as I will be expecting further feedback and additions on Monday.

3.8.6 **Weekly analysis**

This week I started with a lot of feedback on the page and with requested changes. In general, I made a lot of mistakes with the fonts and some things going away from the screen when the resizing it. I also had an issue with the modal I created that when the screen is bigger, it also increases the width and leaves two cards very far from each other when they are supposed to be very close together. I fixed this with a fixed width and not the responsive one based on the screen width.

I also fixed completely the re-designed credit selector that was originally thought to be a slider, but this changed to be a numeric selector with a minus and plus button on each side. With this the user would quickly add or take away five credits. I also made sure the selector would never change the values to strings, so I had to think about the events @input and @keyup handling the user interaction with it.

And finally I created the code for the Slack channel from the backend, this worked fine and I also learned to debug it. Even though it feels like I am done with this feature, I still may get more feedback related on how it should work perfectly because the credits and subscriptions are not yet clear from the backend to distinguish one another.

3.9 Week 9

The goal of the week was to change and improve the layout of the subscriptions page again and bug fixes. The bug fixes were related to text being too long to escape the screen or layout borders, and adding loader indicators when requested are not finished loading or restricting pressing the button “next” too fast when images have not finished rendering.

3.9.1 **Monday 22.11**

Today again I got feedback for the subscription page and I was asked to change the layouts and how the total credit price is calculated. After the feedback, I worked on multiple changes as small tasks and this is how I spent most of the day till I pushed the changes back to staging for further review. I also applied translations and unscoped styles for those that include objects within bold or link tags.

3.9.2 **Tuesday 23.11**

Today I worked on fixing when a subscription is a credit subscription or a campaign subscription that can include credits but is not necessarily the same from the backend. Then I adjusted these conditions in the frontend and tested the cancel subscription button and realized this had some errors related to the modal not opening. So I had to work on a fix for this and I also had to change the unscoped styles that I applied yesterday. I used for this the “v-deep” for “v-hmtl” tags.

```
::v-deep .content__notification p a {
  color: #5e0f00;
  text-decoration: underline;
  &:hover a {
    color: $color-primary;
  }
}
```

Figure 11. V-deep selector usage in styles

```
<div class="content__notification">
  <!-- eslint-disable vue/no-v-html -->
  <p
    v-html="$t('SubscriptionContract.ContactUs', { link: 'mailto:sales@company.com' })"
  ></p>
</div>
```

Figure 12. V-html example for translations

The reason why I had to use this tag is because the translation file includes a bold tag and a link tag within the translation file object, and these tags are not recognized with standard CSS classes. After everything was completed and tested, the changes were pushed for further review and I was introduced other smaller tasks totally unrelated to the subscriptions page.

3.9.3 **Wednesday 24.11**

Today I had a shorter day as I had a Christmas party at the company, so I worked on smaller features, mostly bug related and when I was done I left the keyboard to attend the event.

3.9.4 **Thursday 25.11**

Today I continued with other tasks that involve bug fixes and as yesterday we had a celebration party, today again is a slower day so I will have to wait till tomorrow for feedback on these features. The bugs are mostly related to text that goes away from the screen, adding a loader when an image has not rendered yet in a gallery component and changing the text for a button, instead of having a “send”, it should say “open” to prevent misleading the user.

3.9.5 **Friday 26.11**

Today I worked on fixing another submission text escaping the screen when is too long in characters. This was a bug. I also changed a reply button text to another one to not mislead the clients with the “reply” former text. Instead it should just say “open” and I created a loading status for images that open in a content gallery component. The images take time to load if they are big in size, and pressing very quickly the button next would not give enough time to load the next image, but it would only change the text. So I restricted that event with the is loaded logic with @load event used in the image tag and I also added a spinner appear before the next image renders.

3.9.6 **Weekly analysis**

This week was a bit slow as the Christmas party took place, so two days were far from being a full-working schedule. I did accomplish the changes for the layout requested for the subscriptions page, mostly with CSS, adjusting fonts and changing things in the main page, such as texts and colours. I also managed to work on bug fixes. One bug fix was again to cut long strings, and make them continue in a new line if the word is long enough to exceed the layout border. Then I changed some translations for buttons and I used a loading spinner for images before they finish rendering in a gallery component with a load event in the image tag.

3.10 Week 10

The goal of the week was to finish up the subscriptions page with all the latest additions. Implement a full name extension that would replace the repetitive method field that joins the name and last name char fields. Add a download button to directly download an image or video asset.

3.10.1 **Monday 29.11**

Today I fixed again the layout from one of the cards with grid and flex displays, and added another breakpoint so the cards would not escape the screen when resizing it to a very precise width. I tested it in different screens and worked fine so I pushed it to the branch before pushing to production eventually as the rest of my colleagues will test it on staging after their sprint ends.

After that, I fixed some review comments on my last pull request, basically changing naming conventions and CSS class names.

Then I continued with a completely unrelated frontend and backend task. For this I had to display the number of deleted content, in addition already existing uploaded and purchased content for organization users.

Then after that, I started with a task of adding a “full name” extension for user models with Django, and I had to change all the serializers that previously were used as a method field to a char field.

3.10.2 **Tuesday 30.11**

Today I worked on two tasks and I had the monthly meeting with my colleagues where we discussed how happy we are with our sprints and what we could suggest for a change. We used to have daily meetings through a voice channel daily and we decided to change that habit to posting our daily tasks on a text channel, which makes our communication more asynchronous.

We have today also managed to push to production one of the features I worked on a month ago. The content library page which took long to be deployed for all the issues the filters presented and also due to newer additions from the CRM team as they reviewed. Today I added a download button to a chat attachment, so users can directly download the videos and images that are posted in the chat. As these assets come from a different origin, it does not work properly in development but it will in production, but I will hear for some recommendation tomorrow from another colleague.

3.10.3 **Wednesday 2.12**

Today I worked on a task to change the publisher’s name signature sent as an email in one of the Mailjet templates when they are accepted to campaigns. These requires only backend code, and basically a change of variables passed from other files and inserted into a function named “send_email”. Then when restarting the Docker component, these are only allowed to test in development if I comment out an environment variable that blocks me from sending emails. In order to play safe, it is a good practice to not send emails when testing in development mode.
After testing the it for some time I managed to change the email template and this task was later pushed to production.

Then I worked on a bug that is only reproduced on Safari browser, where one component will not render properly with the z-index property. This property makes a modal be always on top of everything in the screen, but for some unknown reason this works fine on Chrome and Firefox and not on Safari. I decided then to just change this component to a

simple confirm dialogue, as this feature is only for staff users, this change was also approved.

Then I continued for the rest of the day fixing the styles, once again, from the subscriptions page, to make it better in mobile screens when opening a modal, and to make that modal work properly on Safari, as again, this one works a bit different and makes it look somehow different with the indentations. So I fixed that with more concrete sizes in pixels and not with percentages as some were, and I made use of ems. As Kathleen McMahon suggests in her post “Pixels vs. Relative Units in CSS: why it’s still a bid deal”, the pixels would not solve the zooming of the page and font size scale content in the browsers as much as the relative unit “rem”, which will keep the structure of the content consistent with the layout when it is at 100% zoom and twice as that.

Then I also fixed other things requested to changes, such as titles, translations and how the credits are fetched, and again to asses when a subscription is really a credit or a campaign plan, we may need to discuss this more in detail before this is pushed to production as it still remains unclear for now, after several times tackling this feature.

3.10.4 **Thursday 2.12**

Today I worked on another fix for the subscriptions page by adding a total amount of credits to the credits that are visible for users. These credits are the unspent credits only and are a sum of every credit package or campaign plan subscription regardless of whether they have been cancelled or expired.

Then I continued to work with the download feature for the chat, I tried a few things on the frontend and I was told to just lead the user to the link of the asset as the Cross-Origin Resource Sharing, also known as CORS, is not allowed from the CDN the asset is coming from. So opening the asset URL in a new window to then download the content is the best option for now to place safe.

Later I also pushed other features that I worked on early this week to production. And I spent most of the day on the task for the full name extension. This results to be quite challenging as I have to change the full name methods, previously used to combine the user’s first name with the last name, in every possible file were was used and in every serializer. And in the view sets insert the full name extension with a mix into the QuerySets. Today I struggled with this task so I will continue tomorrow and see if I can get further help with it.

3.10.5 **Friday 3.12**

Today I continued working with the full name extension and also adding the last modifications to the code to get the subscriptions page ready.

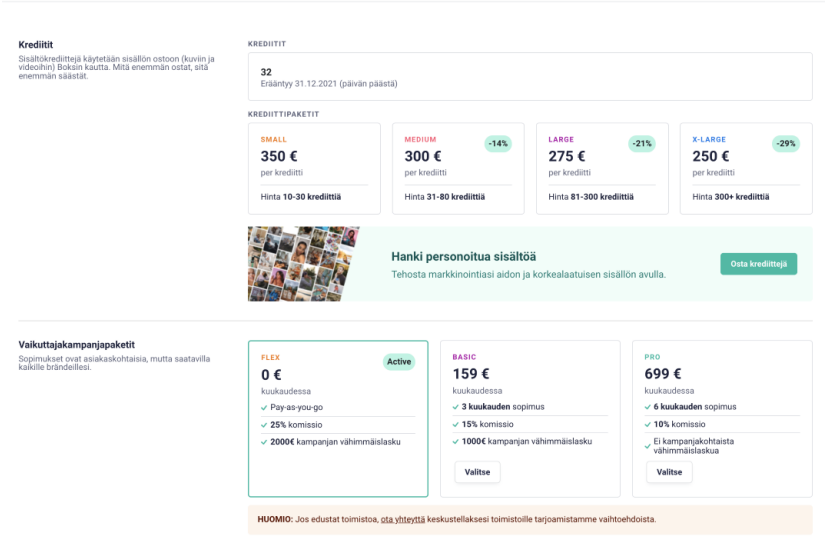

Figure 13. Main subscriptions page example

As the full name extension contains a so-called “ExtendingManager” class, this one I will have to leave it for further inspection.

And I finished the day with no further things to add to the subscriptions page as I left the last changes in the branch ready for review and possibly push for production.

3.10.6 **Weekly analysis**

This week I managed to finish up the subscriptions page and to be ready for review. As I asked for feedback from my colleagues, they really did not have anything to add to modify in the looks and the functionality. Then I should wait only for the feature requester to tell what else should be include or if it is ok to be pushed to production. As there is nothing else to add I can consider this task as accomplished.

Then I managed to solve the download feature for attachments in the chat and the full name extensions is the feature I could not accomplish during this week fully, as this requires a great deal of modifications in the backend, and anything that is not include could break the whole application. I also came across with the “ExtendingManager” class that I could not solve this time on how I would handle the extension with.

4 Discussion and conclusions

After ten weeks of writing the report on my work and the skills I have gained, I can conclude that I have developed several new skills with CSS, Vue.js, JavaScript itself and Django.

This report is about becoming a full-stack developer with no previous work experience. My skills were stronger in React.js and Node.js previously and now at my workplace I have to get around with Vue.js and Django. Throughout this ten-week period I challenge myself to understand what it is like being a full-stack developer as I start as a junior developer. I will be assigned tasks for the frontend and backend of the main code of platform we are currently developing.

Weekly I explore how well I coped with each task and what I had learned from them.

The results showed me that as a junior developer it is not easy to understand anything without taking time to explore the code, ask questions and find the right source. Then it is important to know how to test and anticipate possible errors in the code. The code could have been written a long ago and most of it is understood by asking people who wrote it. However, some code should be re-written or upgraded to produce a better application. Therefore, by understanding what each line does, organizing everything, making new components that will simplify their re-usability later and writing human-readable code, so it becomes easier for someone else working on it later, can be a challenge for every new developer like me.

I have learned from the results that it is always good to ask the team members for clarification when working on something, even if a question sounds too redundant or trivial, because the tasks may turn out to be a wasted time if they are underspecified, and this ends up with everything being redone soon later or simply forgotten.

I learned from this report that without work experience communication can be challenging at the beginning, when one is a new in a group. So it is important to keep everyone up-to-date with what one is doing and have plenty of questions that can be asked, all at once, to avoid distracting colleagues from their own tasks much. But also many of the features can change and communication can seem affected, when both parties did not understand or not clarified details. This seemed to happen a great deal with the subscriptions page, as I got feedback I realized there were newer requests to changes in the layout or the functionality, when initially was meant to be finished rather quick.

The results showed me that I have accomplished most of the tasks with front-end but the back-end was more complicated. For the back-end I used Django and complex database queries that were written in predefined classes. Most of them written by my own colleagues, and my previous experience from the school only allowed me to use SQL queries, which works different and it is seen as less maintainable within this company. Also finding the right source for what is right with Django, and with it, have the certainty that the code would be approved by my reviewers, was more difficult than finding a source to improve the layout and general looks of the application with CSS, which is used for styling and is essentially part of the front-end.

The results showed that I could learn to handle Github properly, which was one of my weakest points before writing this report, to utilize different branches of code and merge them together, and this way, prevent having conflicting code.

I have learned from the results that working in different tasks can feel overwhelming and lead to confusion later or slow down professional development. Then it is important to focus on one task at a time, and only when is fully tested and approved, then move onto the next one. For this reason, it may appear that I worked on very simple tasks each working day, when in fact, a great part of my working schedule is spent on reproducing, finding and analyzing the code previously written by someone else. This is important to fully understand and learn from each task, otherwise one can produce poorly written code and the result will contain bugs. Therefore, timing is not set in stone and flexibility is something every work place should stick to in order to produce the best results.

During this report, after using my own sources and I have found my own way to how I code. This has been mostly reviewed and commented on by other team members. Generally, the first solution is almost never final, so I had to redo things to satisfy the company’s way of doing things. The code can be optimized and the experience from my team members provided me a different approach to things. For example, with Django queries, It was not a good idea to assign a field to a serializer that would not change and keep the default value. In my case this was a Boolean to create checks for filters. So it was better to use it from the frontend and not serialize it.

Finally, it is important to emphasize from my results that one cannot possibly be everywhere and know everything, or take care of everything yet when is new at a company. There were parts of the code I did not touch at all yet. For example, Docker containers, the formatting checkers that take care of the code to launch errors when the format of the code is not appropiate or automated tests fail, database optimization, or in general, how the information is cached or stored. The company's built-in translator code and the web sockets that allow to have a real-time chat notification and group interactions

in the application. As these were not part of my responsibilities so far, then the path to becoming a full-stack developer is still in progress, and from the results, it will take me longer but I am heading there.

For the future, it will be important for me to understand new ways of optimizing code and techniques that will allow me to write better code for its reusability and documentation. The testing skills are something to look to improve as I mostly test from the console, debugging, browser's developers tools and network tabs. All this can be improved by learning new techniques for simplifying testing and minimizing the time spent on this process.

References

Ananda Projapati 2021. How to use CSS breakpoints to create responsive designs. URL: https://getflywheel.com/layout/css-breakpoints-responsive-design-how-to/. Accessed: 29 November 2021.

Brock Herion 2021. Create Reusable Models with Django and Mixins. URL: https://python.plainenglish.io/creating-reusable-models-with-django-and-mixins-2126c5f11eac. Accessed: 4 November 2021.

Chris Coyier 2021. A Complete Guide to Flexbox. URL: https://css-tricks.com/snippets/css/a-guide-to-flexbox/. Accessed: 29 September 2021.

Chris Coyier 2020. Approaches for a CSS Masonry Layout. URL: https://css-tricks.com/piecing-together-approaches-for-a-css-masonry-layout/. Accessed: 1 October 2021.

Chris House 2021. A Complete Guide to Grid. URL: https://css-tricks.com/snippets/css/complete-guide-grid/. Accessed: 29 September 2021.

Django REST Framework 2021. Serializer fields. URL: https://www.django-rest-framework.org/api-guide/fields/. Accessed: 29 November 2021.

Django REST Framework 2021. QuerySet API reference. URL: https://docs.djangoproject.com/en/4.0/ref/models/querysets/. Accessed: 2 December 2021.

Emma Seppälä, Kim Cameron 2015. URL: https://hbr.org/2015/12/proof-that-positive-work-cultures-are-more-productive. Accessed: 15 November 2021.

Grant Pierrus 2015. What is a moodboard and how to create one?. URL: https://interiorstylehunter.com/what-is-a-moodboard-and-how-to-create-one/ . Accessed: 27 September 2021.

Heydon Pickering 2014. Axiomatic CSS and Lobotomized Owls. URL: https://alistapart.com/article/axiomatic-css-and-lobotomized-owls/. Accessed: 16 November 2021.

James Gallagher 2020. How to Use the Git amend Command. URL: https://careerkarma.com/blog/git-commit-amend/. Accessed: 4 November 2021.

Kathleen McMahon 2019. Pixels vs. Relative Units in CSS: why it’s still a big deal. URL: https://www.24a11y.com/2019/pixels-vs-relative-units-in-css-why-its-still-a-big-deal/?ref=heydesigner. Accessed: 2 December 2021.
Kubo 2021. Skipping Django Serialization of Rarely Changing Objects. URL: https://kubo.rocks/wissen2go/skipp.ing-django-serialization-of-rarely-changing-objects/. Accessed: 27 October 2021

Louis Lazaris 2017. Overflow-wrap. URL: https://css-tricks.com/almanac/properties/o/overflow-wrap/. Accessed: 26 September 2021.

Michael Thiessen 2018. How to Dynamically Add a Class Name in Vue. URL: https://michaelnthiessen.com/dynamically-add-class-name/. Accessed: 6 October 2021.

Renat Galyamov 2019. How to Check If an Image is Loaded. URL: https://renatello.com/vue-js-image-loaded/. Accessed: 26 November 2021.

Taylor Lorenz 2018. Custom Photo Filters Are the New Instagram Gold Mine. URL: https://www.theatlantic.com/technology/archive/2018/11/influencers-are-now-monetizing-custom-photo-filters/575686/. Accessed: 28 September 2021.

Vue.js 2021. Event Handling. URL: https://vuejs.org/v2/guide/events.html. Accessed: 18 November 2021.