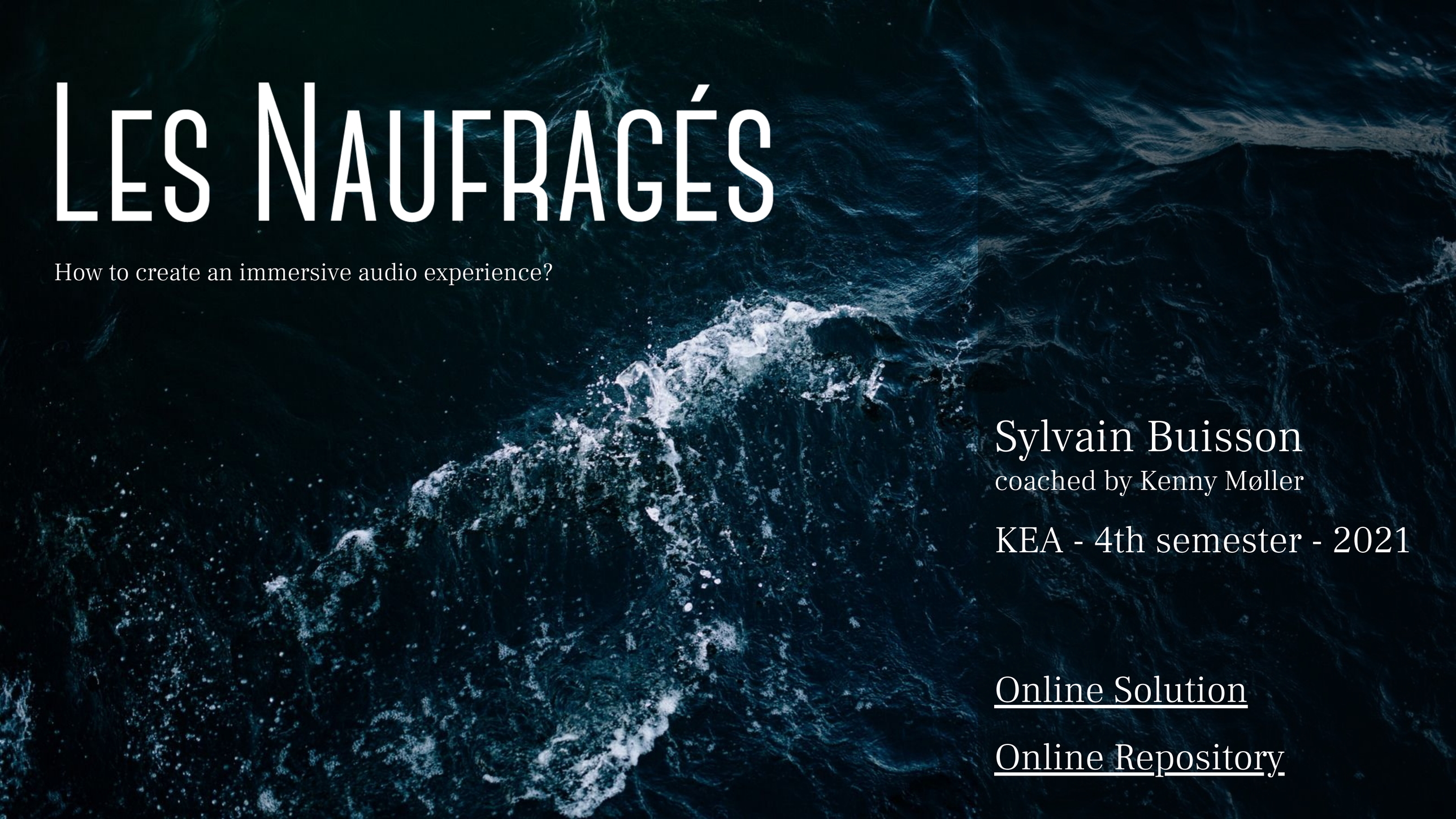
Les Naufragés

How to create an immersive audio experience?



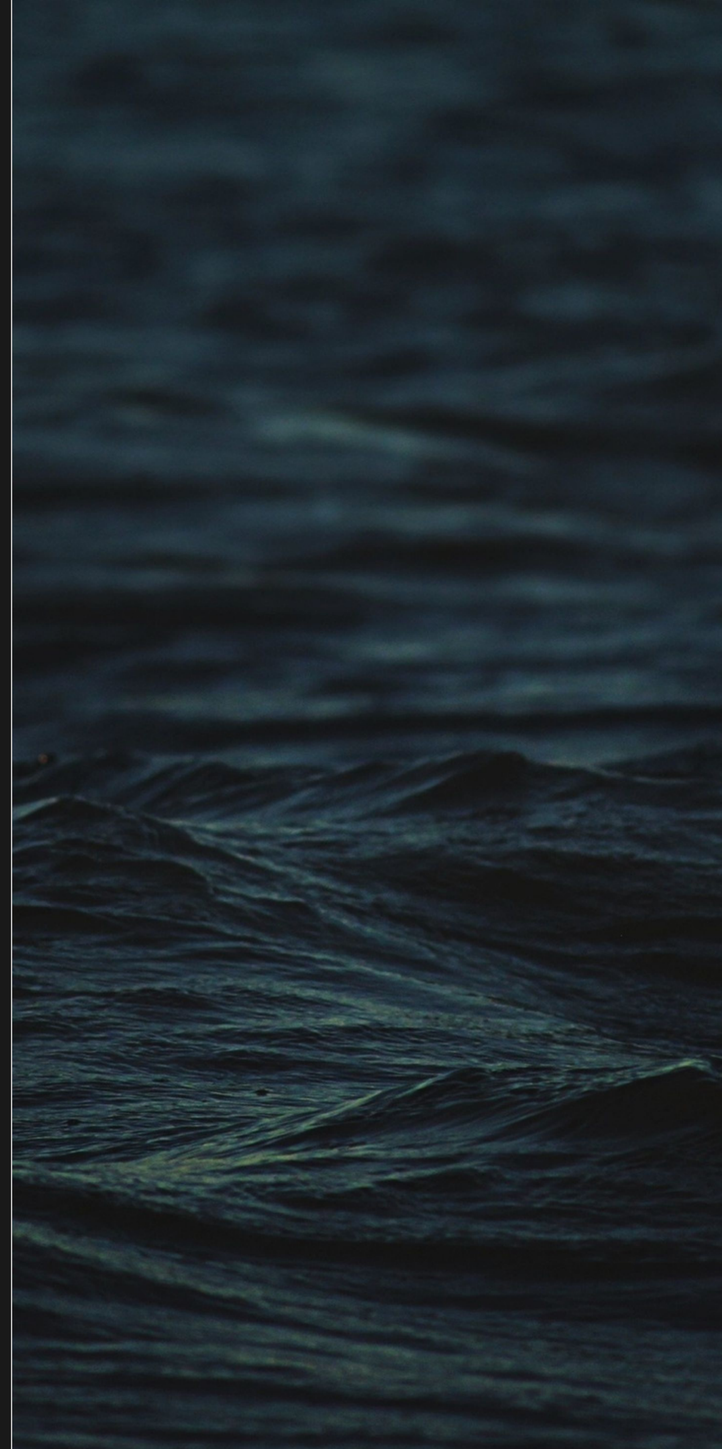
Sylvain Buisson
coached by Kenny Møller

KEA - 4th semester - 2021

Online Solution

Online Repository

Table of Content



Introduction to the case

For the 4th semester final exam at KEA, we are asked to create a multimedia solution to help a company solve a specific issue. This multimedia solution has to include a digital interactive website, and should cover every area of our degree. Namely: User interface development, User Experiences, Content production, Business and Technology.

The project is made “with a company in mind” but not for a company. The priority is to answer the pedagogical objectives of the exam, rather than satisfy the company. That being said, I have worked with the company where I did my internship - and kept working there as an intern during the project. The fact that I just did my internship with them did influence some of my choices.

Because of the specific technology stack and context (working in French for example), it was easier for me to do this project alone.

The solution is available at https://vue-orpheus.web.app/ with the login admin@tamanoir.co / Admin123!

Introducing the client

Tamanoir Immersive studio was created in 2017 by Rémi Large and Samuel Lepoil. Their idea was to be the leader in producing “immersive experiences”. It is a creative studio, doing both production and development of AR and VR projects. For example, it recently produced (meaning finance and help with art direction) an immersive theater experience based on the life of Calamity Jane.

I started working with them to redesign their website Tamanoir.co.

It had performance and UI issues at the time. The previous Web developer they had hired to create it had left during the project. It impacted our next project together in different ways.

Mostly, they were “suspicious” of web developers. Because of their previous experience and not wanting to have someone full time, they were looking to have full-control over the new solution.

Since the previous web-developer happened to be a Vue.js dev, I offered to develop their solution in Vue as well. It would make sure that the different web solutions they have (including their website) would be coded in the same framework. If they had to hire a dev in the future, a Vue.js dev could have a clear overview of all things' Tamanoir.

02

Introducing the client

Note about Tamanoir's business model

Tamanoir is a small French company. It currently only has two employees (its founders), and uses interns otherwise. One of the specialty from Rémi Large is fund-raising, grants and sponsorships. What Tamanoir does very well is pitching projects to obtain grants. Most of the projects they will present are "proof of concepts". Most of Tamanoir's revenue are grant-related. It meant that Orpheus had a limited budget of "don't cost money".

It is a very big part of Tamanoir's culture. When "Les Naufragés" were started, it was only an idea of synced events. Tamanoir would then adapt the content, perspective and branding to the grants they are applying to. If there is a grant on "Modernizing Greek Heritage", Les Naufragés would then only have characters with a mythological names, etc...

Part of the decisions made around Orpheus were let open to adaptation because of the system. Orpheus might need a re-branding at anytime to fit a specific request for grant. It is the same for Les Naufragés and most of Tamanoir's projects.

Problem Formulation

The COVID pandemic impacted Tamanoir’s activity. Many of their projects are based on “interactive theater” (as shown in Calamity Jane). They had to be cancelled or were impossible to do anymore while maintaining social distancing.

They decided to make 2021 their year on “Audio projects”. Inspired by the multiplication of “Zoom” based experiences, they wanted to develop their own set of immersive experiences based on audio. We discussed the creation of a platform to support a majority of those projects - both in terms of creating experiences, and sharing them.

Together, we found out they needed a platform that will be used to create multiple audio experiences sharing these specifications:

- Collective (2 to 100 people)
- Time-specific (ticketed like a show)
- Space specific (e.g.: Luxembourg garden)
- Choreographic (users move during them)

Problem Statement

How can I create an online platform for Tamanoir enabling a frictionless multi-device immersive experience for both their customers and admins?

- "Frictionless" is one of the selling points of the project for Tamanoir: how can we limit the number of steps for "customers/users" to access the immersive experience?
- "Multi-device", which includes both "mobile/desktop" experiences, but also have all of the websites synchronized and live at the same time. It is the question of "how to stream audio live on multiple devices?"

One of the defining factors for the project was also “Scalability”. Tamanoir is still a very small company: with only its two founders and then interns. The objective was to then have a solution that is able to scale naturally with its public and revenue.

There was no possibility of having someone full-time on it. A lot of the choices made during the development of Orpheus were led by those constraints.

Terminology

To help clarify what I am referring to:

- "Les Naufragés" (The Stranded) is a specific audio immersive experience, produced and created by Tamanoir. When I joined the company, they had the idea for it - while not being sure how to technically do it.
- "Le Continent Intérieur" (The Continent Within) was an existing prototype of two 10-minutes audio experience. It is very similar to what Les Naufragés should be.
- Orpheus is my fourth semester project. It is the platform created according to what "Les Naufragés" needed.
- A Participant is a paying customer, using Orpheus to access an immersive experience. They do not specifically participate in Les Naufragés. They are not part of Tamanoir.
- An Admin is a Tamanoir employee using Orpheus' administration pages to create stories, events, add participants, etc... They are not part of the experience. An Admin is a user with specific write and read rights on Orpheus.

Introducing Orpheus

What is Orpheus?
What is my project about?

07

What is Orpheus?

Orpheus supports live audio immersive auto-theater experiences. For example, Les Naufragés is a story to be created, involving 10 to 20 participants. They would meet at a given time, in a given place, to act together a story.

Each participant would come to the event with their own mobile phone and a headphone, go to their specific URL which would play an audio (matching a “role” in the story). The audio is specific to each role.

One of the example given in the creative brief is as such:

9 people could be asked to cover their right eye, while a tenth would be asked to raise their left hand. It would work as a “vote” in the story, and the participant who raised a hand would become the "leader" of the group.

The complete creative brief, in French, is available here.

What does Orpheus do?

When we started this project, Tamanoir had a few ideas for Les Naufragés, but not really about anything else.

They needed participants to all come together, at a given time and a given place. They could start simultaneously an immersive experience. It was very important to Samuel Lepoil (Artistic Director on Les Naufragés) that the participants could start the experience with minimal effort.

I was given a very first technical brief, to discuss and expand from there. It is available in English here.

Orpheus creates interactive audio experiences. It creates events, with a specific time and story - to which you can add participants. It then creates URLs for the participants to access their audio.

Structure of this report

1/ Users & Features

In this part, I'll talk more about the brief, the research before starting Orpheus and what was needed. It will focus on the features that were needed for Orpheus (building up from Les Naufragés' needs).

2/ UI & Interfaces

Then I'll explain the UI choices and designs for the solution. I'll talk more about the first prototypes, the current design and why they were made as they are.

3/ Technology & Versions

Finally, I will discuss the actual code for Orpheus - its technology stack and hosting, while discussing the data-model for the whole solution as well.

11

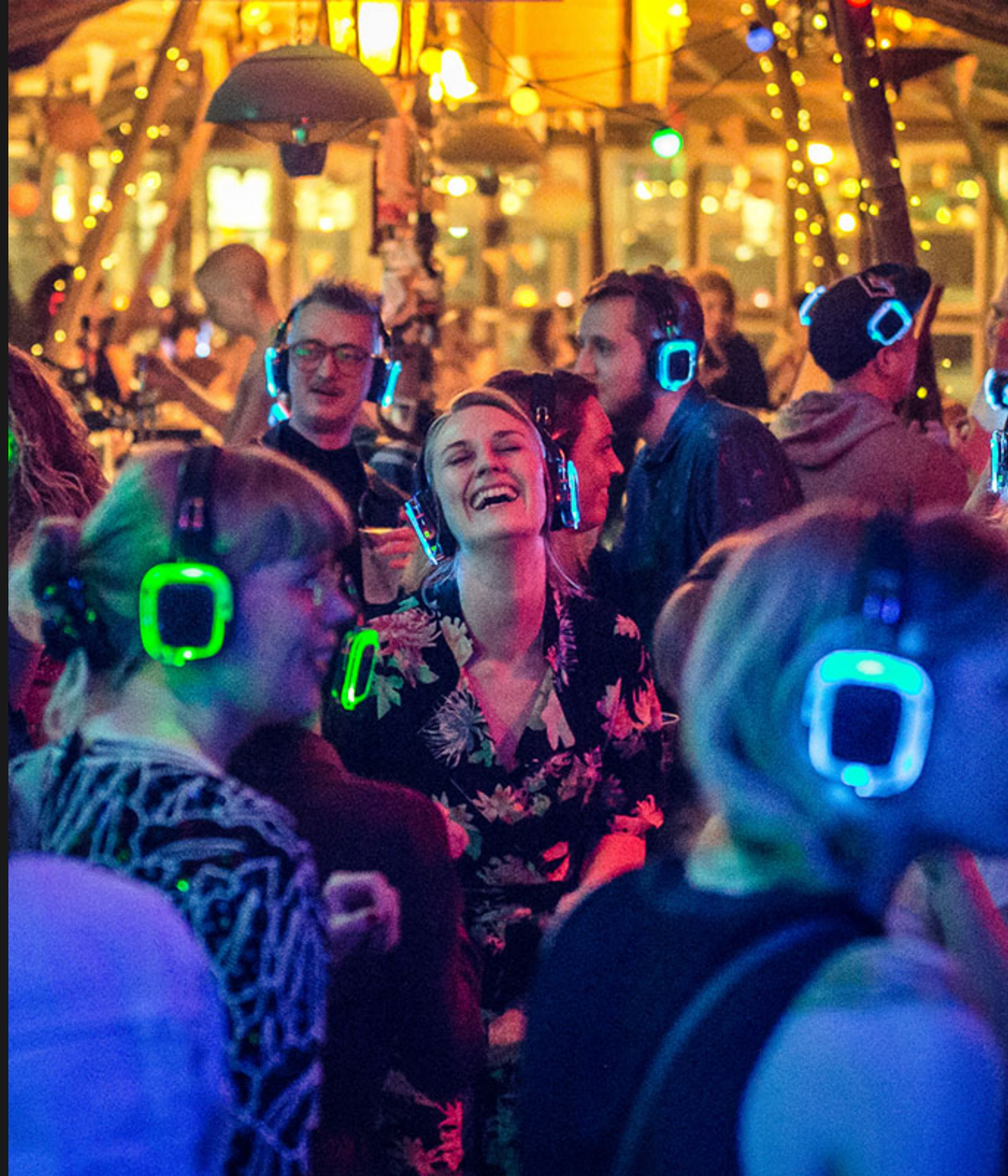
Users & Features

User research,
benchmarks,
needed functionalities

Users

What are the key users of the platform?
What can we learn from them to shape the solution?

12



Users

Outside research

Tamanoir Immersive Studio was granted a partnership with IronHack. They won a two-weeks UX research project, and decided to use it for "Le Continent Intérieur".

"Tamanoir's upcoming project 'Le Continent Intérieur' is a story of two men traveling south pole in the 1910s, who were trapped in an ice iceberg and try to escape to survive. The story is told in two different narratives to two different participants. While users listen to different narratives in different roles at the same time, they become immersed in story guided by sound and light."

Le Continent Intérieur is Les Naufragés' big brother. It was the first audio immersive project by Tamanoir, and uses the same principle of synced audio (while having individual narration). There is also an available audio prototype (in French) for the story. It was used to test out Orpheus.

Since Le Continent Intérieur has a similar principle to Les Naufragés, I used this research as a stepping stone for Orpheus. The conclusion was:

- Orpheus needs to be a **Web app**. Participants dislike having to download an application specifically for an event.
- Orpheus needs to function **with minimal personal information** from the users.

The complete report is available in English here.

Personas

Since Orpheus has two different types of user, I have done two different personas. I have used a persona to refer to:

- The administrator of the platform
- The participants of immersive experiences

The users of this solution would most likely be technology inclined: either because working for an "immersive studio" or because they are willing to go for an immersive audio experience. Since users are technology-inclined, they would recognize conventions from habits, without thinking about it .

The "participant persona" influenced the design. For example, when first designing the play button: It is possible to have a “play” and a “stop” button, or a button that will change its purpose and micro-copy.

I believe that it is a generational mental model. Older generation will associate the TV/DVD **remote** to "Play/stop", which needs two buttons - while younger generation will most likely think of a changing button. The example that I had in mind was "**Spotify**". It is the play button that I use the most.

Judith (the persona for the participant page) is closer to Spotify than TV remote.

TR

Jean-Michel

Jean-Michel is an employee of Tamanoir Immersive Studio. He was trained to use the solution, and knows what it does and why. He is working on most web tech for Tamanoir as an admin, he is young and used to technology.

Goals

- Efficient solution that helps him win time. Orpheus needs to free him time to do other tasks, rather than take up time.
- There should be one smooth process to create a complete events. J-M does not want to do back and forth between solutions

" The less time I spend on Orpheus, the better"

Characteristics

- Good with technology
- Trained
- In a hurry

Judith

Judith is interested in participating in an “interactive immersive audio”. She is very curious: that’s why she is willing to try something new. But she is also very technology oriented.

Goals

- Fully enjoy the immersive experience she has paid for.
- Technology is fun, only if it is intuitive, and thinking about it is a conscious choice

" I don't know what Orpheus is...?"

Characteristics

- Curious and enjoy design
- Coming to fully immerse herself in the adventure
- She will easily recognize bugs, missing features, failures
- Orpheus is not a brand she should recognize

User Journeys

Participants

The research from Iron Hack showed that users did not want to be redirected across multiple platforms. So we grouped Reservation/Paying on the ticket platform - which limited the interaction with Orpheus to the day of the event.

The two key steps I had to focus on for Orpheus were "sending an email with the relevant information" and "creating the URL for the event".
TR

Starting from the brief, we discussed the key touchpoints. Ideally for Tamanoir, once you have purchased a ticket, you do not have to interact with anything after that. We scoped the project to exclude ticketing on the platform itself for the first iteration.
TR

- Buy a ticket from a third-party platform
- Receive an email about the event and all of the information needed
- Connect to the platform on the day of the event

Key Issues

Participant's process

The participant's process was much more complex technically than the admin's side. The contraints for the user (limiting the number of input, making it frictionless and immersive, ...) made the process more difficult, but also clearer.

TR

- Buy a ticket from a third-party platform → What information does the admin will have?
- Receive an email about the event and all of the information needed → What information are needed by the participant? How can Tamanoir send emails?
- Connect to the platform on the day of the event → How can Orpheus deliver individual audio while limiting the inputs needed by the user?

User Journeys

Admins

Tamanoir needs Orpheus to be as profitable as possible, which means reducing costs. **To reduce costs, there is no admin physically present during the events**. Orpheus has to be self-sufficient as a solution. Any admin, from anywhere, should be able to create an event. Working backward from that:

- What is an event? What is a story?
- What do you need to create? How to limit the number of creations needed?
- How can I limit the time spent on Orpheus?

- Create a story
  - Add a name
  - Add roles
  - Add roles' audio
- Create an event
  - Add a date
  - Add a time
  - Add a story
- Add participants
  - Add an email address
  - Add a role
- Send emails

TR

Note on user testing

Because of COVID, it was impossible to test the solution “to scale”. There were three different testing sessions organized in France during the month of April that were cancelled for quarantine reasons.

Les Naufragés was supposed to be the first experience, and includes about 10 people per event. It was changed to be tested on “Le Continent Intérieur”. It is a two-person experience, which can be done at home. Orpheus and the participant page were tested with that.

As Orpheus is made to not include a physically present admin, it was possible to test around the world. It was not possible to get face-to-face immediate feedback, but rather have interviews after the experiences. It meant that testing was not as directed as I would have loved it.

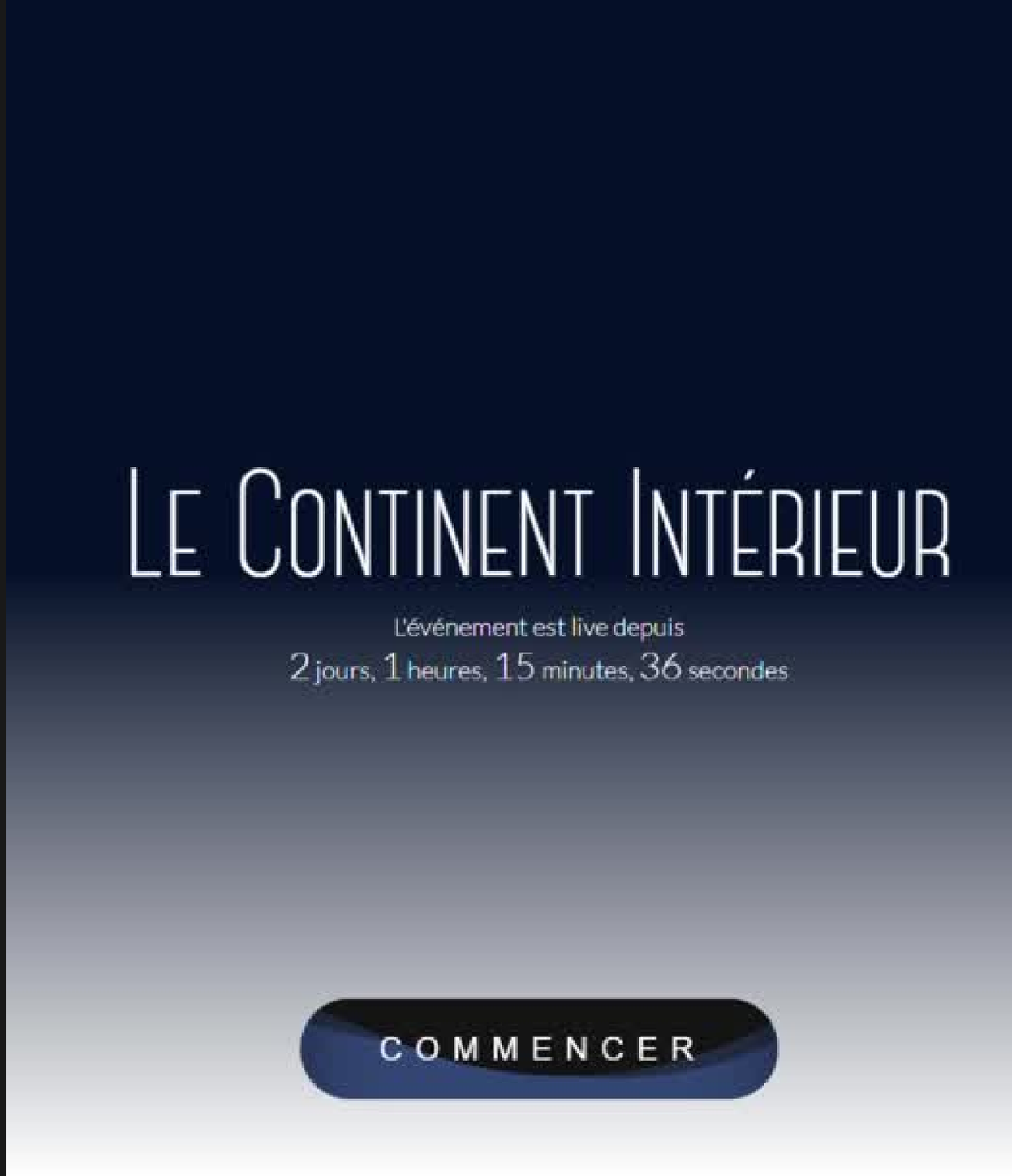
Key learnings

Because people who were testing did not buy a ticket, it wasn’t clear for them that the event was time-sensitive. There were tests that needed to be redone. It mostly impacted the “email” on Mail Chimp - and parts of the UI is made to reflect that aspect of the experience.

It was also important to have a "first try" mode, which serves as a mini-testing, and shows the countdown to the user.

Features

What did the different part of Orpheus need to do?
How can it all be a coherent ensemble?
How can the end experience help create the platform?



Features

As I wanted participants to have as little to worry as possible, it meant that the page itself needs to handle most of the data. Any user can go on the page, follow the one instruction on it.

The page needs to know:

- Is the event live or not?
- What is the audio (URL from the server)?
- Has the audio been started?

For the participant, it reflects in a few states:

- The Event has not started, and the Audio has not started
- The Event has not started and you are trying the Audio
- The Event has started, and the Audio has not started
- The Event has started, and the Audio has started

TR

For the admin platform, it meant that we needed to create:

- A story - which would have a name, and audios
- An event - which you tell you when it is
- URLs - which the participants could access

TR

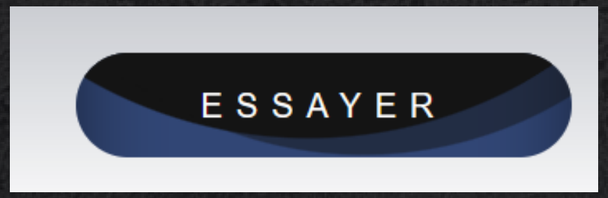
The participant page is the experience for participant. It gives you a stream to your specific audio. It currently has three elements:

- Title of the Story
- Countdown to the start of the event
- Start / Pause button

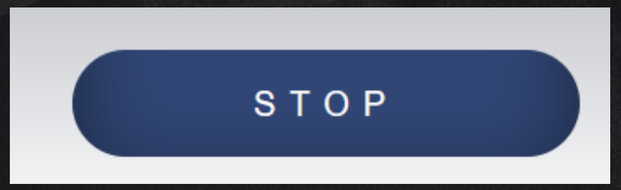
Before live event

Trying before the event

Since there is no physical staff present, it was important that the participant could try out the page at any time. Once they received the email with their URL, they can try the audio. But we don’t want it to spoil the experience, so it’s copyright free music! It is also the only time when the user can stop the audio.



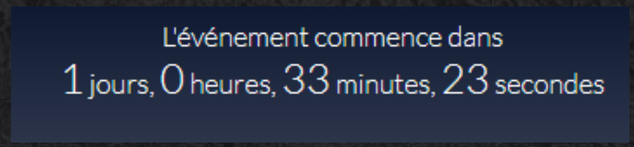
"Essayer" translates to "try"in French



Only time where you can stop the audio

Timer Countdown

The participant page includes a countdown timer. It displays when the event is supposed to start. It was a necessary feature - **emphasizing the time-sensitive nature** of the event. It also builds on the "tension" mentioned during the Iron Hack research. Once the event has started, it will tell you how long ago.



Building up expectations

Functional Diagram

Full Admin Process

Admin Logs In → Creates a story → Creates an event → Selects an event → Add participants → Send email

Full Participant Process

Participant buys a ticket → receives email with URL and instructions
- starts audio on D Day
- tries audio before

All of those steps needed to be transformed into functioning features in Orpheus. Some steps were taken out of the scope of the project (buying tickets, for example), some were kept in the project but outside of Orpheus itself (sending emails).

26

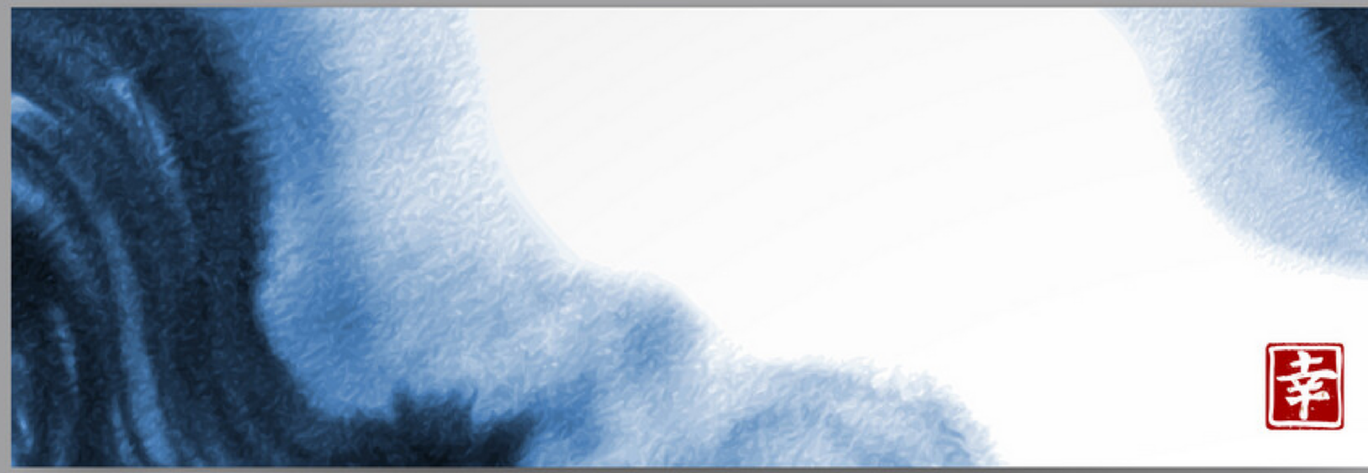
Design & UI

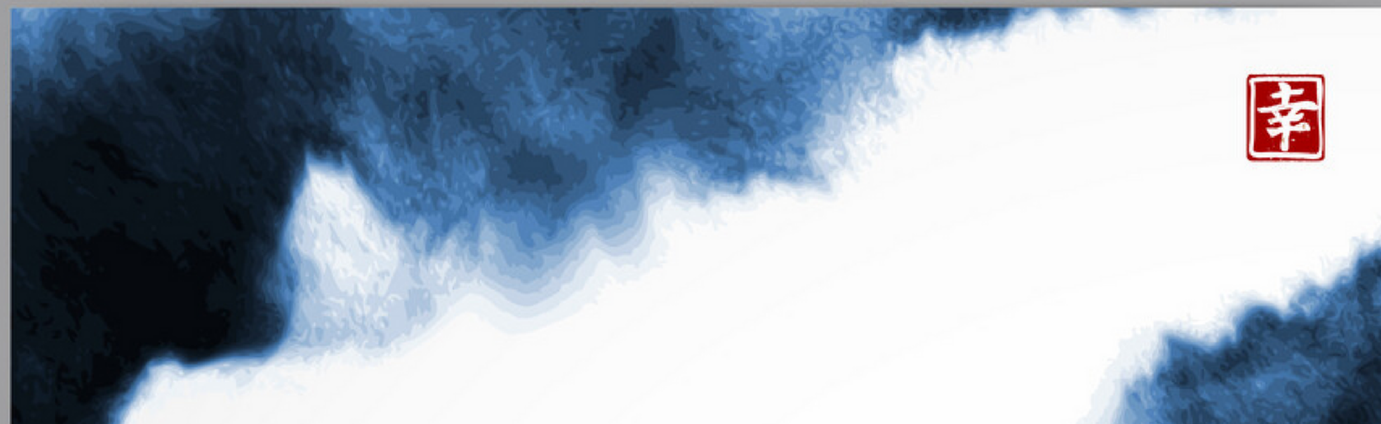
Style-tile,
Layout,
Content

Design

What are the key elements of Orpheus' design?
Why does it look and feel as it does?

27





Working on Design

Once I had established the necessary features, I could work on the UI and interface. The very first version (on XD) of Orpheus was much more audacious than the final results. It included:

- “Past Events”
- “Settings”
- a “Calendar” view.

In general, the colors and branding was kept loose, as Tamanoir works around exterior financing. Orpheus might need to change its name and colors to match a demand for funding.

The first prototype is available here.

The name “Orpheus” comes from the Greek mythology: Orpheus was a musician, poet and prophet. The solution is creating the stories, and therefore should be considered a poet itself.

The favicon is a picture of the constellation “Lyra”, shaped like a Lyre, the Greek instrument symbolizing poets.

The two main colors of the first prototype were “Purple” (#27192b) and “Orange” (#D87034). The final design kept the idea of a “background color” (Blue) and a highlight color (White) - which made the switch easy to do.

TR

First Design

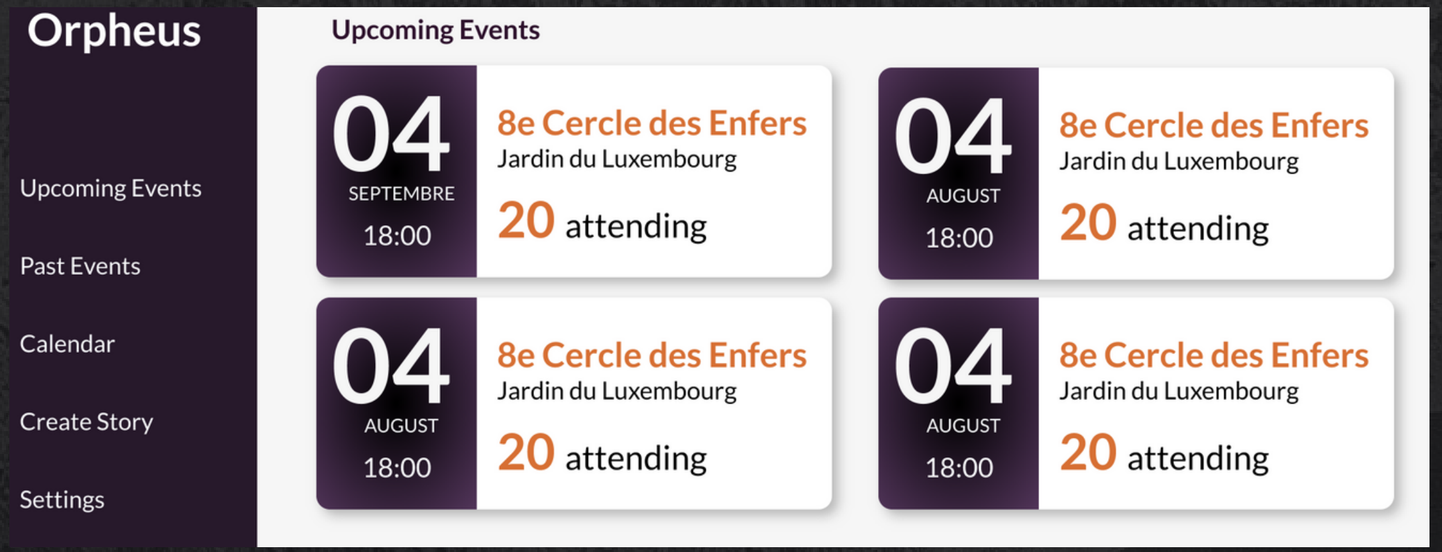
Colors

“Purple” (#27192b)

“Orange" (#D87034)

Purple is the color of Royalty, with a sense of expensive. The darker tone was meant for usability, but also to reinforce a sense of magic and mystery. There also are purple elements in Tamanoir's branding.

Orange is a contrasting color, helping highlight information



First version of the admin-side

TR

Current Design

Colors

#0e2d75 #1c2742 #051029

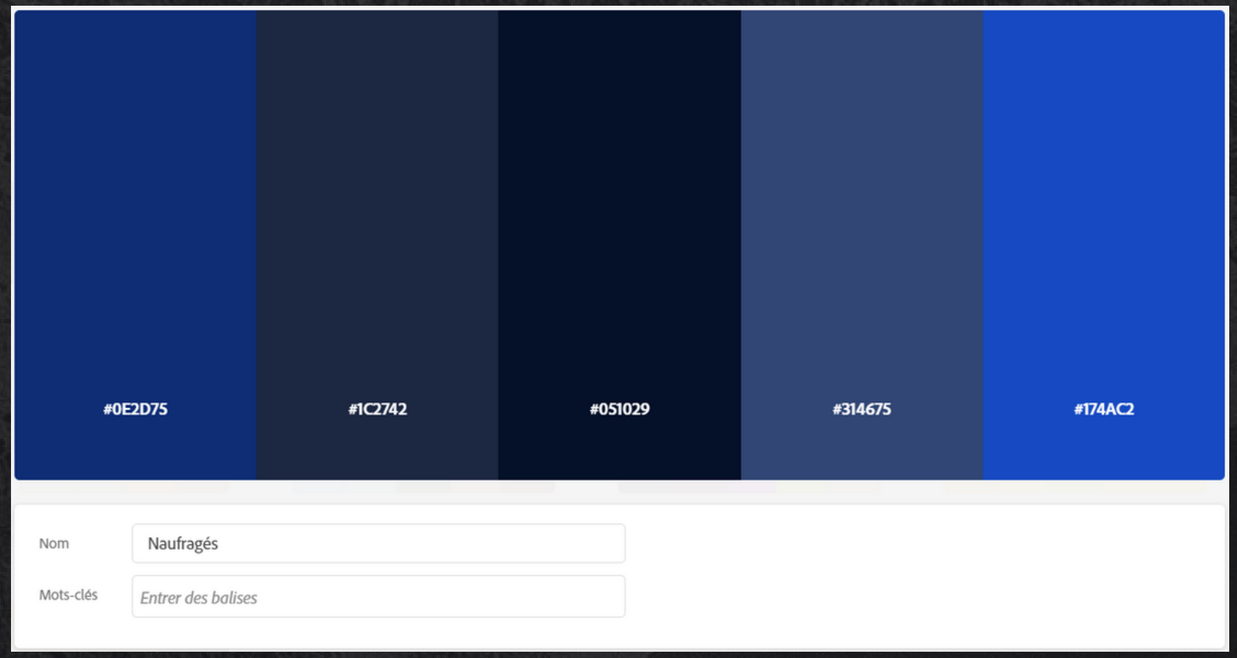
The current colors match the “Sea theme” of Les Naufragés. Rather than built on contrast, it works on gradient rather than contrast.

TR

There is a loss on the highlighting power - which reversed the organization of the colors. The first prototype had the brighter color to highlight information - while the second one works on the darker color (contrasting the white) to highlight information.

TR

Adobe Library (complete colors)



Current Design

Typeface

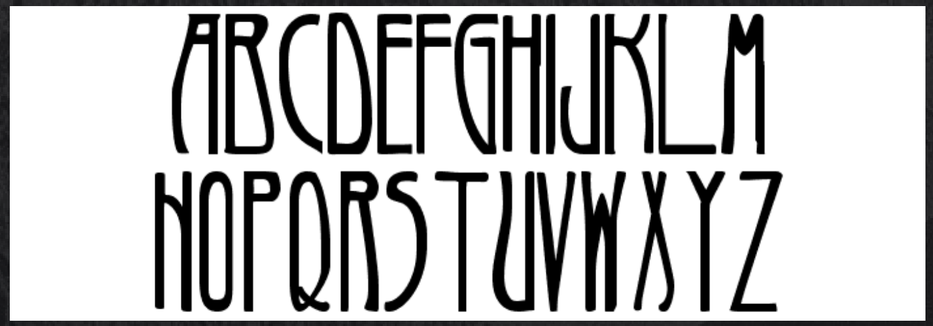
There is a Title typeface - “Canter” and a Text typeface - “Lato”.

Canter

Canter is part of the Les Naufragés esthetic. Both typefaces used are sans-serif, to inspire movement and waves, rather than having strong angles. Canter is an all caps, condensed typeface available in six different weights. It was designed as a display type for titles, headlines, and posters [source].

As such, it allows for a “timeless” feeling matching the title “Les Naufragés” and giving you a first feeling of being stranded. It happened to match a lot of the Continent Intérieur design, which is set in 1900's (which meant that possibly the stories could be packaged together).

Canter



1920's Art Nouveau Typeface

Current Design

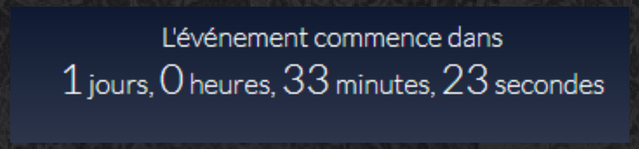
Typeface

There is a Title typeface - “Canter” and a Text typeface - “Lato”.

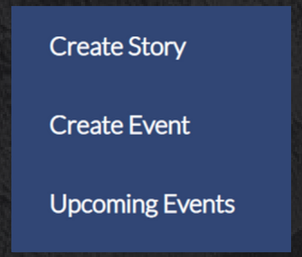
Lato

As of August 2018, Lato is used on more than 9.6 million websites, and is the third most served font on Google Fonts, with over one billion views per day [source].

I would recommend Lato for any solution where the user needs to feel comfortable, and should not think about it. For an admin platform, Lato helps the user not think about it.



Lato on participant page



Lato on Admin page

Layout

How are the different pages of Orpheus organized?
How does that organizes Information?

33

Layout Admin-side

The first version and the current version have kept the same layout for the admin-side solution

Layout explained

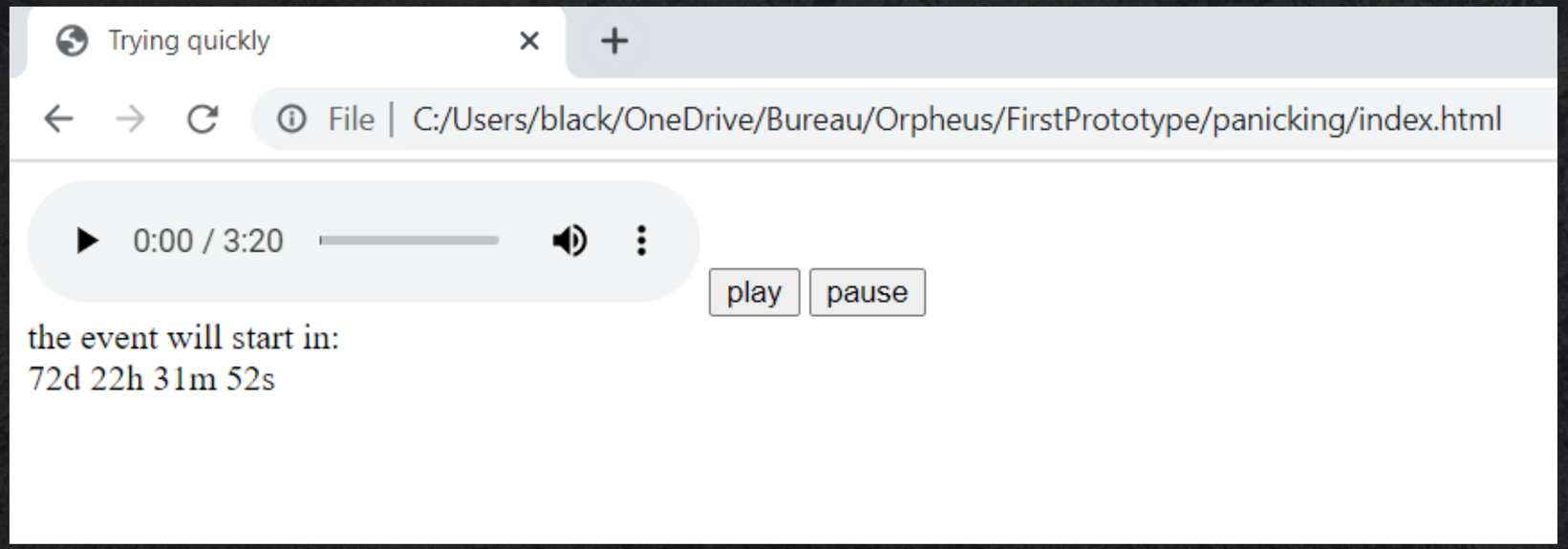
The aim behind this layout was to have a layering effect. It should reflect four different zone, with matching priority in data shown:

The first ratio divides between the menu and the content. It should keep a menu big enough to be easily usable, while not overtaking content. The layout is also coherent throughout the solution to keep "existing content" on the left, and "creating content" on the right. This type of layouts also makes it easier to have components.

A darker color is used for the menu, to contrast with the white of content. There is a stronger contrast on the card. The users gaze should be attracted by the strongest contrast. The drop-shadow on the cards for existing content also reinforce the layering.

Layout Participant-side

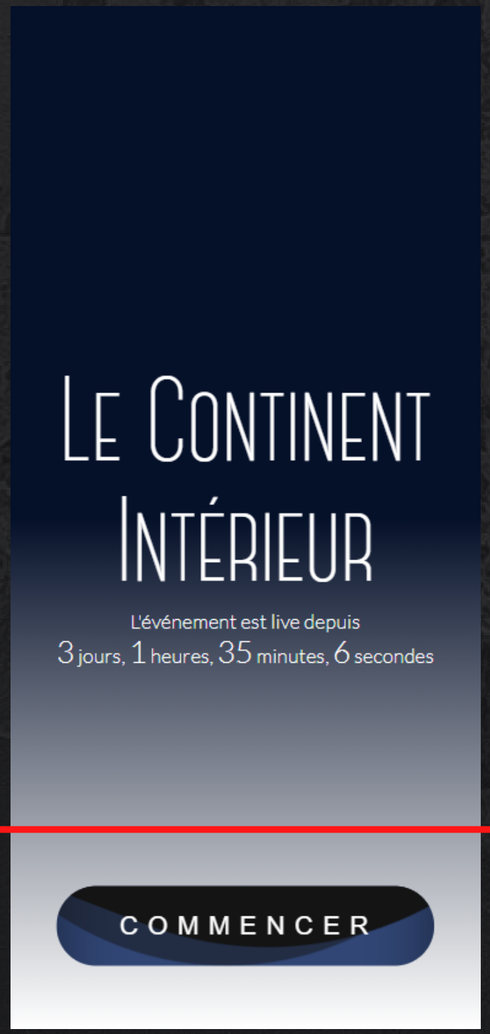
The first version of the participant page was slightly more "minimal". It was nicknamed at the time "panicking". I had to see if I could technically make Orpheus works as expected. It came at the end of the exploratory phase.



There is not much to say about this layout.

Layout Participant-side

The current participant page has a different design:



80%

20%

This layout is meant to reflect a "call" system. The button is placed where you would most likely have the "answering" / swipe to answer when you receive a call. This button would then be already associated to "this will produce audio". Having the button centered in the middle of the page felt closer to "buzzer", one-time sound.

The creative director also wanted (for Les Naufragés) to have a lot of white space - to inspiration "désolation" and "isolement". The gradient is centered to highlight the title and offers a contrast on the button.

To keep the sea theme, the button has an animation to show the sea. It is a rework from this codepen.

TR

UI

How are elements throughout Orpheus designed?
Why does everything change according to state?
How does that help the users?

38

Guiding Principles for the UI

In & Out for the Admin

For the admin-side of Orpheus, the objective was to limit the time the user would spend on the platform. The actions are relatively independent from one another - you can create a story one day, create the event the week after, add participants when necessary. Every action will save data and make it possible to come back to it after.

Information quickly available

Furthermore, the admin should not have to think about what he is doing. If you go on Orpheus with a specific intent (create something), it should be obvious at any time what the next step should be. You shouldn't have to leave an action to look for an infromation and come back to the page. All information will appear when needed.

Can't fail for Participants

With the absence of physically present admin or support, the participant should not be allowed to fail. Limited UI, guided by the page, with the page doing most of the work.

Kind reminder

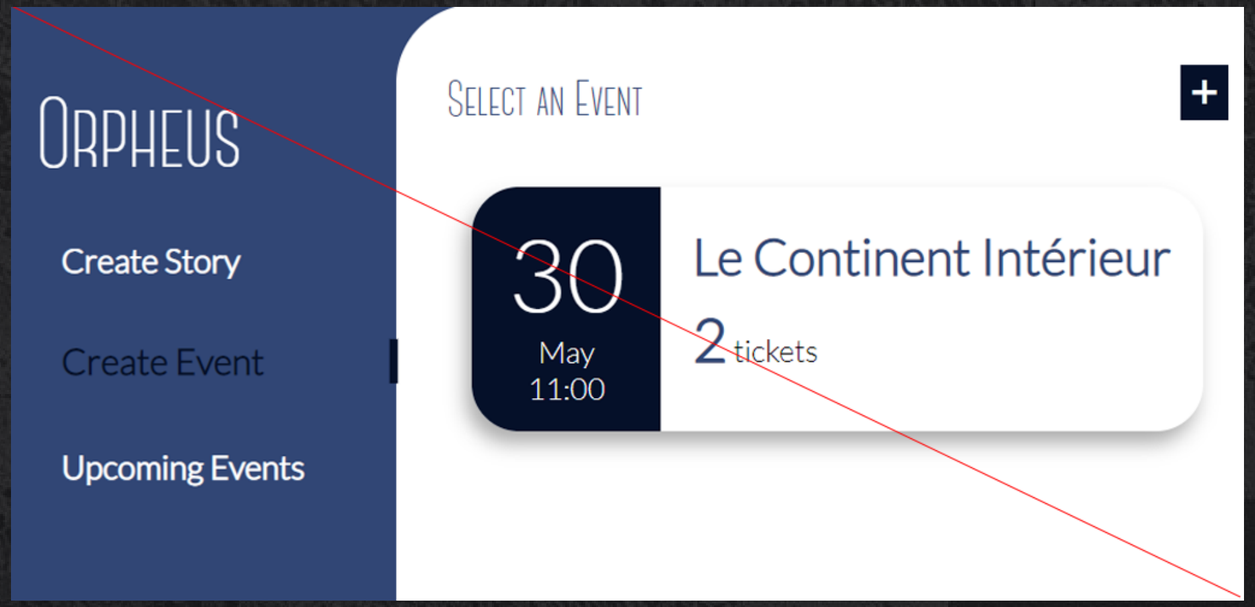
The biggest issue on the Participant page was to remind the user of the time-sensitive nature of the event. I also wanted to give a sense of wonder and excitment to the idea of participating. You are "rewarded" for trying out the page before. In the future, I would love to had more in that direction.

List Rendering

Vue.js offers the possibility to render each element of a list (or array) with a template. It makes it very easy to group elements, while keeping a matching UI.

The elements on the admin pages are either a list (existing stories, events, participants, etc..) or a form. For example, the cards representing the events or the stories are a list of events or stories. Creating templates for those elements help with consistency through the solution, keeping information (and matching data) grouped. Cards with the drop-shadow help creating a mental image of "button".

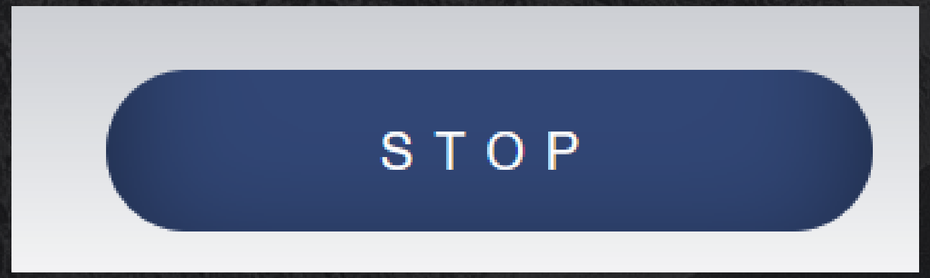
The border radius is kept from the background to the cards - giving a sense of continuity and taking the gaze of the user toward the center.



The button

For the participant page, the button will reflect the 4 possible different state of the solution.

	Audio is playing	Audio is not playing
Event is live	**Live** (live)	**Commencer**(start)
Event is not live	**Stop** (stop)	**Essayer**(try)



The button stays the same in terms of size and position. It will help the user to conceive the different state: if you press the button, it will change something.

The animation on the button only works if there is no audio playing. It should reinforce the sense of mystery and wanting to click to see what it does.

TR

Switching UI

One of the solutions that Vue.js offers is “v-if”. It is a conditional rendering - it switches between a display on / off after checking for a variable. It is the principle on which the button for the Participant Page is based: there are two variables (isLive / isReading) and it will display the matching micro-copy. I have used this a lot in Orpheus. It allows for a quick change between sets of UI, and is very modulable.

The idea was to have adaptable buttons, changing according to what is possible / needed, rather than offer options that should be disabled.

It limits the UI and amount of information displayed at any time. Rather than have a different set of inputs, the right part will change if you have selected an element on the left.

Title changes according to state

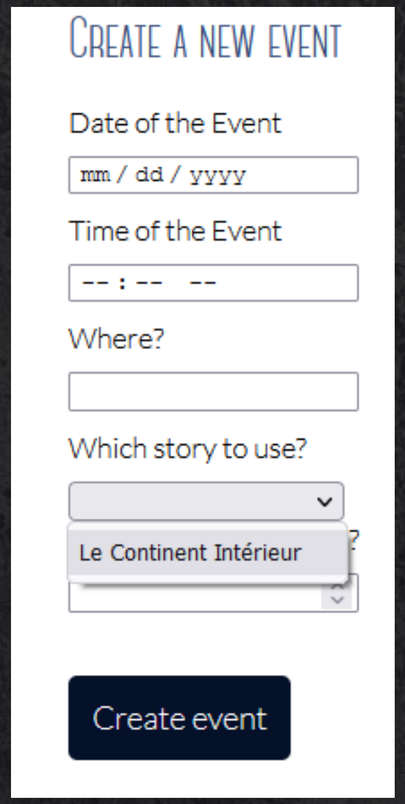
The user does not have a choice. You don't get presented the "create story" / "delete story" simultaneously. But if you have selected a story, the "create story" becomes "delete story".

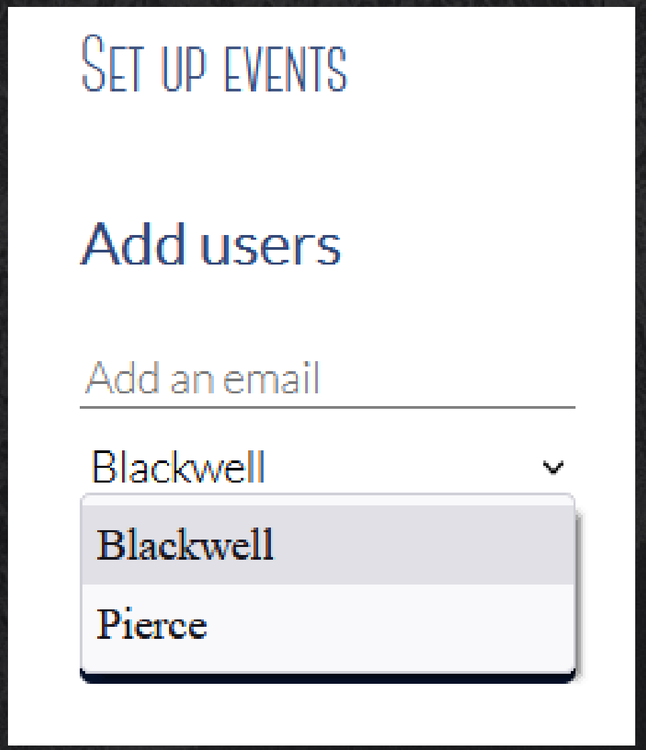
Limiting Options

To ensure that the admin has to follow the steps of the process in order, options during the Event creation, and Participant creation are limited.

When creating an Event, the admin will only have the stories created available. It is so that the admin cannot create an empty event, which would not have audio to play - which would crash on the participant page.

When adding a participant, the admin will only have the roles contained in the story from the Event. If the event was created with the story "Le Continent Intérieur", there are only two roles available when adding a participant (Pierce and Blackwell).





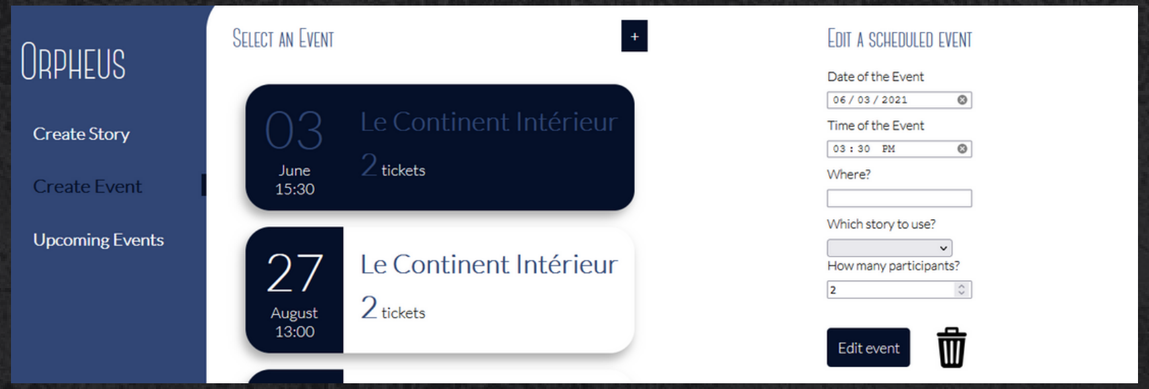
Switching States

To help make switching states smoother, there is a reset button, through the “+” button. It will reset the UI to creating and will focus on the first input on the right. The general principle is:

"Don’t let the user see something they can’t do."

So Orpheus will have different "modes" (creating or editing) reflected in a complete change of the UI. It would have been possible to layer the changes. For example, all creation would have been done with modals (reinforcing the layering). With this solution, I wanted to reinforce a tunneling effect: if you clicked on X, your next step should be Y - don't try something else.

"Editing" mode for Events



Clicking on any dark blue button (click again on the story, click on "+", click on the matching submit button) will take you out of "editing" mode.

Mail Chimp

How does Mail Chimp fit with Orpheus?
How does its content support the immersive experience?

45

Using Mail Chimp

There are two-parts that Orpheus was not meant to handle: ticketing and emailing. Ticketing is still up for discussion and not part of the scope of Orpheus. It is supposed to be the technical solution to the experiences, and Tamanoir wanted to see if it was possible.

Emailing

There was a discussion to include emailing in Orpheus: once you create the participant for an event, you would actually send the email then. But it would require Cloud functions from Firebase to do so. Firebase has pre-made extensions, making it easy to install and start sending email from the server. But it requires a (paid) Blaze Plan, which would take Orpheus outside of its “free” budget.

Mail Chimp

Mailchimp is an American marketing automation platform and email marketing service. It creates email campaigns sending them in batches with custom “keywords”. For Orpheus, it meant that I could create an “email template” - re-usable for one type of event. It would then require the admin to create a “campaign” per event.

Mail Template

LES NAUFRAGÉS

Ce que vous devez savoir

Les Naufragés est une aventure collaborative d'auto-théâtre. Vous allez être les acteurs de votre histoire. Votre participation est la création de l'histoire elle-même.

" Vous vous réveillez sur une plage, le soleil frappant votre visage. Vos lèvres sont sèches de déshydratation et votre estomac grogne de faim. Vous entendez des voix autour de vous... Est-ce des compagnons dans votre misère ou des menaces supplémentaires? Comment faire pour survivre, quand il ne reste que vous? "

Votre aventure commence le **DATE, HEURE**
LIEU

Votre aventure

Tout ce qu'il vous faut est un téléphone mobile avec une connexion internet et un casque audio. Il tient à vous de ramener les deux le jour de l'expérience. Le jour même, il vous suffit d'**ouvrir l'URL ci-dessous** et de suivre les instructions pour commencer l'aventure.

L'expérience dure 40 minutes. Nous vous recommandons d'arriver 10 minutes avant le début de l'événement.

Access Orpheus

What you need to know

The Stranded is a collaborative adventure of self-theater. You are the actors of your story. Your participation will create the story itself.

“You just woke up on a beach. The sun is hitting your face. Your lips are dry from thirst and your stomach is growling out of hunger. You are hearing voices around you… Are they the companion to your misery, or threats you will have to handle?”

Your adventure start on DATE / TIME
PLACE

Your Adventure

All you need is your mobile with access to the internet and headphones. It is up to you to bring both on the day of the experience. On the day, all you have to do is open the URL beneath, and follow the instructions to start the adventure.

The experience will last around 40 minutes and we recommend that you arrive 10 minutes before the start.

Why is it important?

The mail is the first touchpoint for the participant, and gives most of the information. There are multiple type of information given in the email:

- Practical: time / place and what is needed for the experience
- Narrative: it gives a first glimpse of the experience
- Technical: the URL for the participant page.

The URL in itself is made of two set of data:
https://vue-orpheus.web.app/events/event_ID/*|EMAIL|*

The event_ID is specific to the event. It has to be retrieved from Orpheus by the admin. In future versions, it would be nice to have a way to extract the event_ID while redirecting to Mail-Chimp, from Orpheus.

Currently, the Admin has to try out a URL from the event to retrieve the event_ID. It serves as a fail safe for the admin to have to try on the URL and make sure that everything on the participants page is working as expected.

The email address is used as the unique identifier for the participants. When they would buy a ticket, they would need to give an email address per ticket. In Mail-Chimp, that address serves to send the email, and now to “complete” the URL. As such, the email address is used throughout the process and different tools.

49

Technology

Technology stack,
Components,
Store & Routers,
Code

Technology stack

What is making Orpheus do what it does?
Why were these technologies chosen?

50

Technology stack

Vue.js

Vue.js (commonly referred to as Vue; pronounced /vjuː/, like "view") is an open-source model–view–viewmodel front end JavaScript framework for building user interfaces and single-page applications. It was created by Evan You, and is maintained by him and the rest of the active core team members.

What does that mean? Vue is a JavaScript framework, similar to React or Angular - but which isn't maintained by a company (Facebook and Google). As mentioned, the first web developer who worked for Tamanoir was a Vue.js dev. It made sense to keep the same framework throughout all of Tamanoir's projects.

But Vue is also a robust model-view-viewmodel, which means it is very good at rendering (and updating) with reactive data. There are also tools to integrate Firebase easily as well.

Technology stack

Firebase

Firebase is a platform developed by Google for creating mobile and web applications. It offers a series of services for developers, such as Cloud storage or Hosting. I am using:

- Firestore: Cloud Firestore is a cloud-hosted, NoSQL database that your iOS, Android, and web apps can access directly via native SDKs. I am using it to store the “Stories” and “Events”.

- Storage: Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service built for Google scale. I am using it to store the audio per role. It allows the Participant Page to retrieve it from a URL.

- Hosting: Firebase Hosting is production-grade web content hosting for developers.

- Authentication: Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users. I use it to identify the admin on the admin page.

Technology stack

Firebase Pricing

One of the reasons to choose Firebase is the pricing model:

- If you do less than X amount of queries (requests and uploading to the servers) Firebase is free. It means that currently, Orpheus is free.

There will be fees once there are users (understand here “paying customers”). The pricing for Firebase is also scaling. While there is a threshold (while reaching the amount of Blaze plan), then it is a pricing "per" (per read, per write, per mb stored, etc...).

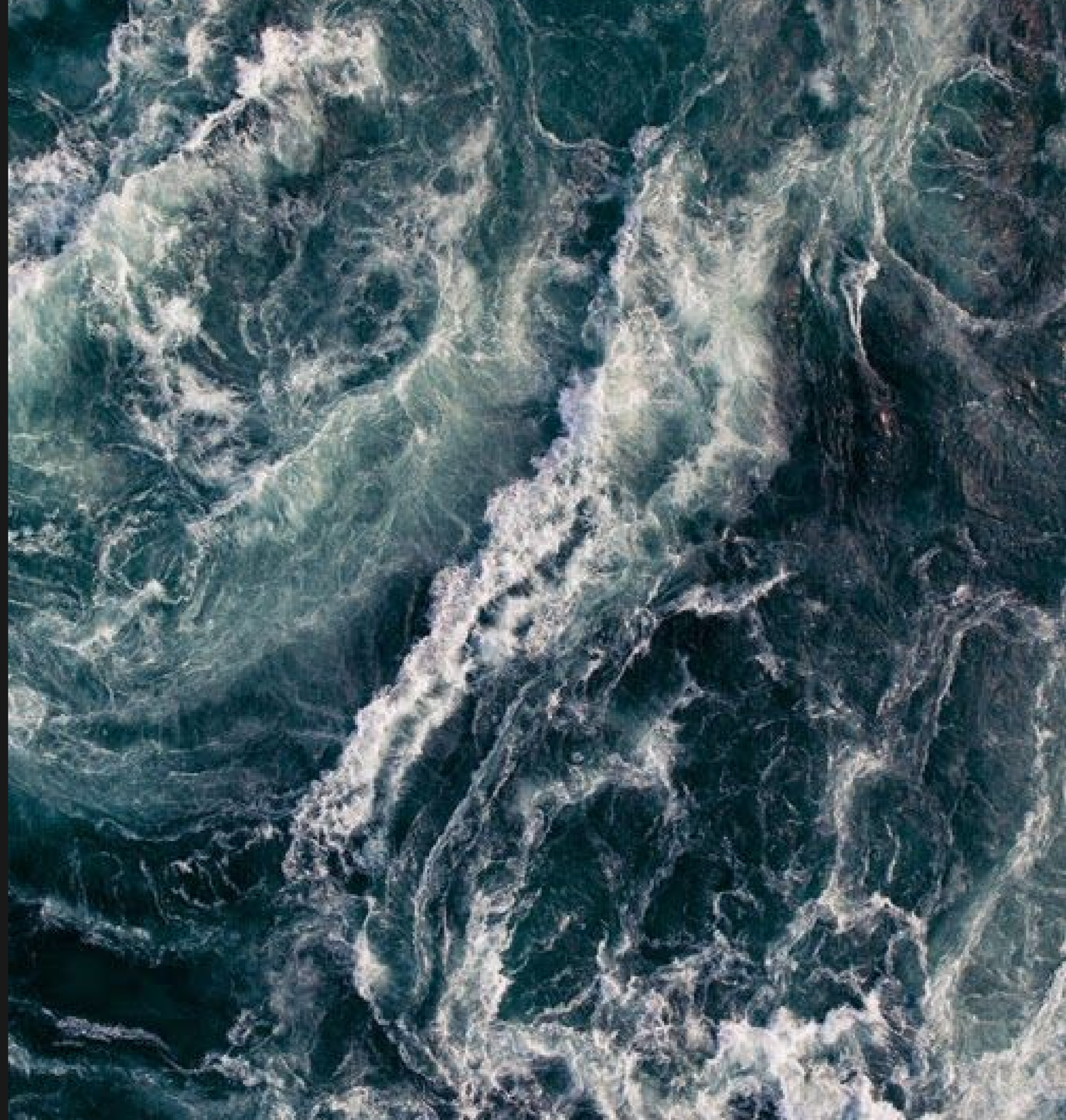
The ticket price should be calculated through the average amount of queries made. It means there is no least amount of customer for Orpheus to have positive revenue.

If Firebase was "membership"-based: we would need to have X customers per month to cover the price of the servers. As it is, if Orpheus does not sell any tickets for a month, it would not cost Tamanoir anything.

Components

How is Orpheus divided?
Why and how use components in Vue?

54

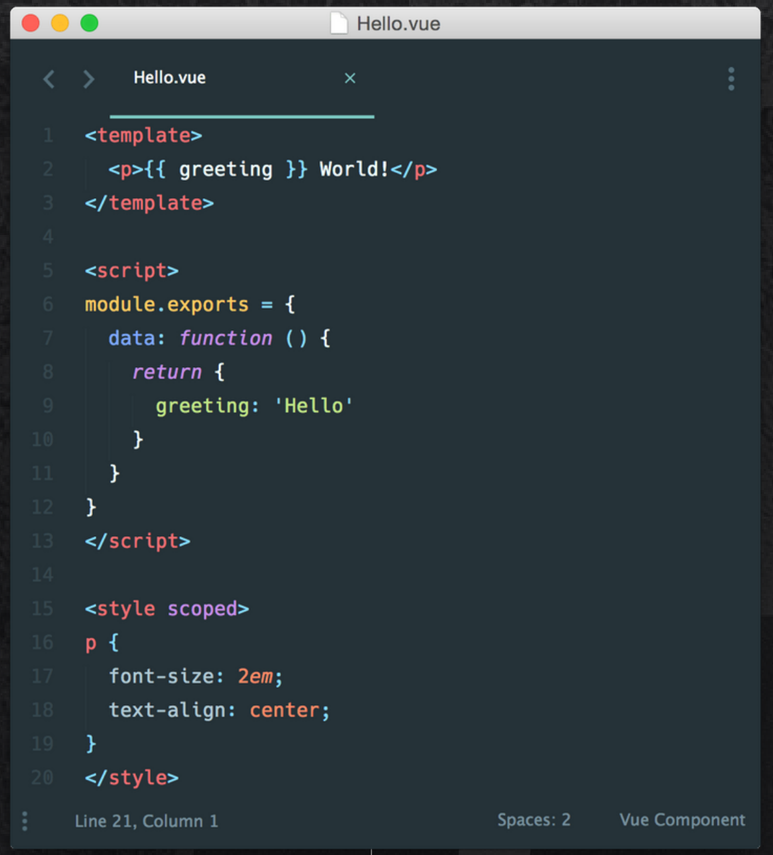


Components structure

Working with a Javascript Framework allows to divide the solutions into components (one of the specialities of Vue). A Vue instance in itself has the same structure as a component:

- **Template**: which holds the HTML structure (as a template tag)
- **Script**: which holds the Javascript code, with a specific structure for data.
- **Style**: which holds the CSS stylesheet.

Within the script, there are sub-categories. Mostly, you will find “Data” (all of the data stored locally on the component), “Methods” (all of the functions used by the component) and different lifecycle hooks (for example, “created()” will trigger all of its Javascript code once the DOM element have been rendered).



Scoped Components

Following the Single-Responsibility principle, each component should be “responsible” for only one action. Components are also “scoped”: everything that happens in a component stays in the component. For example, there are two components: “createStory” and “getStories”.

- “createStory” allows the user to create a story
- “getStories” calls the Firestore to retrieve all existing stories

If I created a “fakeVariable: true” in “createStory”, I couldn’t access it in “getStories”. Meaning, if something happens in one of the components, the others would not know. There is a work-around (the Store), but it mostly helps to create very compact code.

There are three levels to the code structure:

- Views: have a different layout all together
- Pages: dispatch data to components
- Components: one responsibility

There are remnants of the first ideas and prototypes, where the Participants would be logging in: it needed a Help page and an About page (to explain more about Tamanoir and Orpheus).

Component Actions

The minimal level of actions for a component is CRUD (create, read, update, delete) - attached a “document” in the Firestore:

- createStory: Create or Delete the stories.
- userType: Create or Delete the roles.
- createEvent: Create or Delete the events.
- getEvents: Read or Update the events.
- getStories: Read or Update the stories.
- upcomingEvents: Create or Delete the participants.

The structure of the code, and dividing it in components is important for (mostly) two reasons:

- If I have to work with other devs, they can easily read my code, understand what it does, and work on a different components while not interfering with my work
- It allows for the code to scale easily: it is easy to identify which part of the code does what, and what needs to be impacted. It is nothing but Lego blocks!

Store and Router

What is the Vue Store? What is the Vue Router?
Why and how use them in Orpheus?

58

Router

Let's get technical! Since Vue produces Single Page Application, there is no different URL. A “normal” website would have changes in the URL, fetching a different html file. For example, if you go from http://www.domain.com to http://www.domain.com/about, it will take you from one page HTML to another.

A router in a Vue application serves to change the “views” that are needed. The app is browsing through the different files (reflecting them on the URL) according to the router.

The template contains a "router-view" in which the router will render the necessary route (matching a URL/path to a file)

```
<template>
  <router-view />
</template>
```

HTML structure in the component

```
const routes = [
  {
    path: '/',
    name: 'Dashboard',
    component: () => import("../views/Dashboard.vue")
  },
```

Router structure matching a URL to a file

Redirecting & Login

The Router also allows for redirecting. If a user isn't logged in, the dashboard page will ask the router to redirect to the login page.

In more details, there are different "Lifecycle Hooks" in Vue.

While the App is creating (hear "rendering") a page, Devs can trigger actions at different steps. "beforeCreate" means that there is no HTML rendered yet. It will check in the Store if the flag "isLogged" is true or false, if it is false, it will redirect you to the Login page.

Since the Dashboard contains all of the sub-pages and components for the Admin, it is possible to have the redirect there.

```
beforeCreate() {
  if (!this.$store.state.isLogged) {
    this.$router.push({ name: "Login" });
  }
},
```

HTML structure in the component

Store

What is the store? Since the Vue App is made of small components that can not communicate together, there is a Single Source of Truth. The store handles all the shared data along the solution.

If I use the same example again, rather than have a “fakeVariable : true” in one of the component ; I would have “localVariable = fakeVariable” with the fakeVariable stored in the Store. If there is a change in the store, it would impact the whole solution.

For example, isLogged serves as a flag to see if the user is currently logged in. The Dashboard.vue file can ask the store “is it true?” if not, it will tell the router to show the Login page instead.

Note on the Store

On projects with a bigger scale, it is common to divide the store into modules. Rather than have everything, it is divided per type of data / documents. It would mean that there would be at least four different modules (“stories”, “events”, “participants”, “userType”, etc...). As of right now, the different data types have been added one after the other - which makes for a large and long file.

Store Structure

The Store has 4 types of data:

- State: the state is a group of variables at a given time. While in vanilla javascript, you would have let, var, const... Variables are stored in the “state” in the Store when using Vuex.

- Getters: to access a variable, you don’t directly call the state - but a “getter” which gives a “reference” of the current state.

- Mutations: a mutation is an event modifying the state. For example, editEventID will change the current event_id in state to what is passed

- Actions: Actions are similar to mutations, the differences being that: Instead of mutating the state, actions commit mutations. Actions can contain arbitrary asynchronous operations.

Vuexfire (the library handling firestore actions in the vue store) does asynchronous operations, needing to be actions.

Data structure

How is the data structured in Orpheus?
Why and how use Firestore?

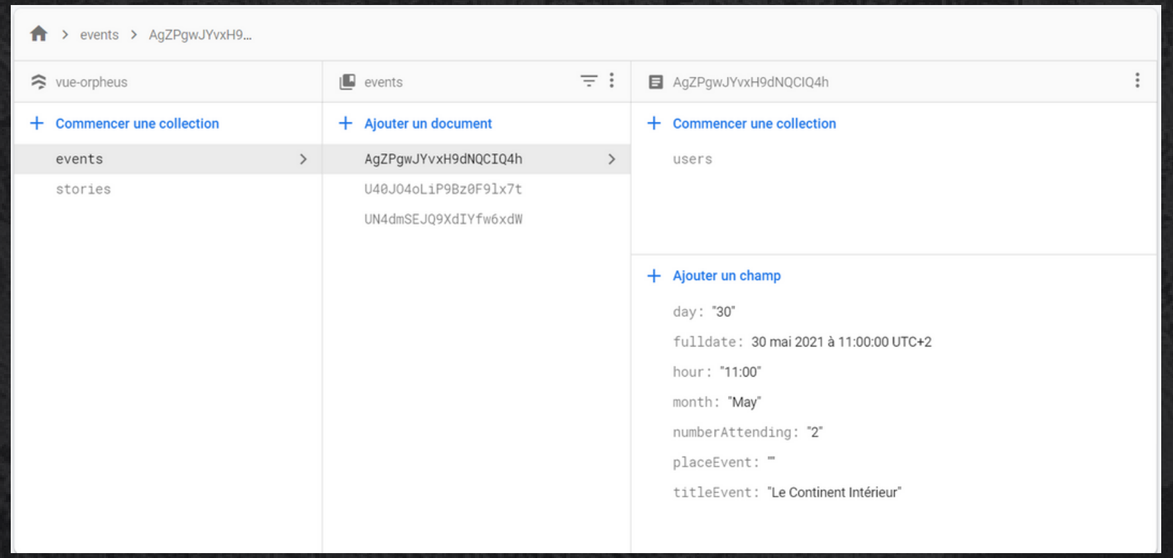
63

Firestore Structure

The Firestore functions as a library. There are documents (individual set of data, with an id), grouped together in a “collection”. For example, the collection “Events” will hold as many documents as there are events.

The problem of a “Data Model” is the “Russian doll structure” needed. For example, an “Event” currently does not contain its story. It contains a string, with the name of the story (set during the Event creation).

When adding new participants, it creates a collection “Users”. A user will contain “an email”, a “storyline” (corresponding to a role), and a “URL”. The URL is a duplicate of the URL in userType. It is made as such to limit the number of queries done by the “Participant page”.



Events stored in Firestore

Impact on Queries

When opening a “Participant Page” (sent by email), there are two sets of data stored in the URL: the event and the user. It is made possible by the router, which creates an empty set of pages.

When working with data, it is important to have a “**unique identifier**”. For example, if we try to get a “user”, it needs to be something that would not return two results. Currently, the email is used as a unique id. So if you are deleting a participant from an event, it will query Firestore by using the email (and do the same for the store).

It means that if you create two users with the same email, it will search on the event (event id is the specific identifier) - and returns the first result with the email. It needs to be change on the admin side (UI feedback) rather than the data model.

```
//query for specific user w/mail
let userDelete_query = db
  .collection("events")
  .doc(this.firestoreId)
  .collection("users")
  .where("mail", "==", clickedUser);

//deletes user
userDelete_query.get().then(function(querySnapshot) {
  querySnapshot.forEach(function(doc) {
    doc.ref.delete();
    console.log("user deleted");
  });
});

//deletes user from local array
let userLocal = this.users.find((user) => user.mail === clickedUser);
let userLocalIndex = this.users.indexOf(userLocal);
```

Flags and Rendering

A Boolean flag, truth bit or truth flag in computer science is a Boolean value represented as one or more bits, which encodes a state variable with two possible values [source]. Throughout Orpheus, there are flags to help re-renders elements. As we discussed during the UI explanation and the buttons, Vue offers the possibility of conditional rendering.

For most of the components, I start with a state of false (dataLoaded, changeDataStatus, etc...), and the function calling the Firestore will toggle the flag to true once the data is loaded onto the store. It means that there are no empty elements.

For example, when the Stories elements are rendered, if it took too long for Firestore to respond - there would be no data to render. I use the same for updating (or adding) elements. You can see it in the solution when the flag moves to “false”, all the elements are changed to display: none.

It is not necessary as Vue is reactive and should re-render elements when the data is updated. For example, if someone were to change directly on Firestore the name of a story, the binding made by Vuexfire would trigger a re-render. But it works as a quick fail-safe for rendering, while not costing much in terms of performance.

Code

What does Orpheus actually do?
What functions is it using?

67

Orpheus Admin-side

- Creating documents: mostly Orpheus on the admin-side is meant to create documents in Firestore. It creates Stories and Events, using the package Vuexfire.

For example, creating a story:

```
<button class="button-class" id="buttonSubmitStory" @click="createStory">
  Create story
</button>
```

Event listeners have short-hands on Vue, and then, even shorter-hand. There are a list of triggers available, including v-on:click=”function()” - corresponding to addEventListener(“click”, function()). Since v-on is so common, vue offers a short-hand “@”. So clicking on the Create story button calls the createStory function.

createStory() will flag the data to be changing; then call for the Store action “addStorytoFirestore” which passes the value of the input as payload. Then it will trigger a re-render by changing the data status.

```
<script>
  import "vue-js-modal/dist/styles.css";

  export default {
    name: "createStory",

    methods: {
      async createStory() {
        this.$store.commit("changeDataStatus", false);
        console.log("called");
        await this.$store.dispatch(
          "addStorytoFirestore",
          document.querySelector("#name-story").value
        );

        this.$store.commit("changeDataStatus", true);
      },
```

Vuex / Firestore

The system works mostly similarly for the rest of the data. If you create an event, a participant of a user type, it will add a document to the firestore (through the store) with the data from the forms' inputs. The left side is rendering of the data stored in the Store.

```
addStorytoFirestore: firestoreAction((context, payload) => {
    return db.collection("stories").add({
        title: payload
    }).then(() => {
        console.log("added new story")
    })
}),
```

Vuexfire adding to the firestore

Orpheus Participant-side

Data Rendering

First of all, the Participant page will get the name of the event (from the event_id in the URL) and the participant information (from the mail in the URL).

```
async created() {
  //getting info for event and user from firestore
  let route = this.$route.params.firestoreId;
  let liveEvent = (
    await db
      .collection("events")
      .doc(route)
      .get()
  ).data();
  this.liveEvent = liveEvent;

  let user = this.$route.params.usermail;
  let liveUser = "";

  await db
    .collection("events")
    .doc(route)
    .collection("users")
    .where("mail", "==", user)
    .get()
    .then((querySnaphot) => {
      querySnaphot.forEach((doc) => {
        liveUser = doc.data();
      });
    });
```

Retrieve information in the URL

Querying Database

Orpheus Participant-side

Getting URLs for Audio

Then it retrieves two URL: the waiting music / try-out music, which is Mozart. And the URL of the audio from the participant, the “role”. Then, it calls the countDown function.

```
//getting info for audio from storage?

let storageRef = storage.ref("first-story/type-a/Mozart.mp3");
let x = await storageRef.getDownloadURL().then(function(url) {
  return url;
});

console.log(this.liveUser.url);
let y = this.liveUser.url;

this.liveURL = y;
this.waitURL = x;
this.countDown();
},
```

Orpheus Participant-side

Countdown function

Countdown does, every second (to get the visual countdown on the page) :

- get the time for the event and transforms it into milliseconds
- get the current time
- subtract the two (called distance)

```
// Find the distance between now and the count down date

let fireStoreDate = this.liveEvent.fulldate.toDate();
let fireStoreDateHour = fireStoreDate.getTime();

let countDownDate = fireStoreDateHour;
let currentTime = new Date();
let distance = countDownDate - currentTime;
this.distance = distance;
```

It then has to determine if the event is live:

1. If the distance is superior to 0, it means that the event is not live yet
2. if the distance is inferior to 0, it means that the event is live.

Since the countdown is in milliseconds, it was more precise than 0 or 1. So there is a third condition of “about to go live”, between 1000 and 0. That condition also triggers (the function is called every second, all the time) if you have started the audio before the event goes live - and the event goes live during that time.

```
//event is not yet
if (distance < 1000 && distance > 0) {
  this.isLive = true;
  this.stopAudio();
  this.startfrom = 0;
  this.playAudio();
}
```

Orpheus core function

The core of Orpheus is the second condition. If the event is live, it will calculate the time passed since the event has started, and will set that time has “startfrom” for the audio tag. So if the event has started 300 seconds ago, your audio will start at that time.

It allows for real-time live synced audio across multiple pages.

The time of the event is used as an anchor for every page to be synchronized together. In the first version of the prototype, it was possible to pause the audio. It would then just have to re-calculate the time spent from the event.

```
playAudio() {
  let lecteur = document.querySelector("#lecteur");

  if (!this.isReading) {
    this.defineSource();

    if (this.isLive) {
      this.startfrom = Math.abs(this.distance / 1000);
    }

    this.isReading = true;
    document.querySelector(".liquid").classList.add("isReading");

    lecteur.currentTime = this.startfrom;
    lecteur.play();
  } else if (this.isReading && !this.isLive) {
    this.isReading = false;
    document.querySelector(".liquid").classList.remove("isReading");

    lecteur.pause();
  }
},
```

Participant page Pause

In the first prototype, the participant could pause the audio, and restart it (still synced to the others).

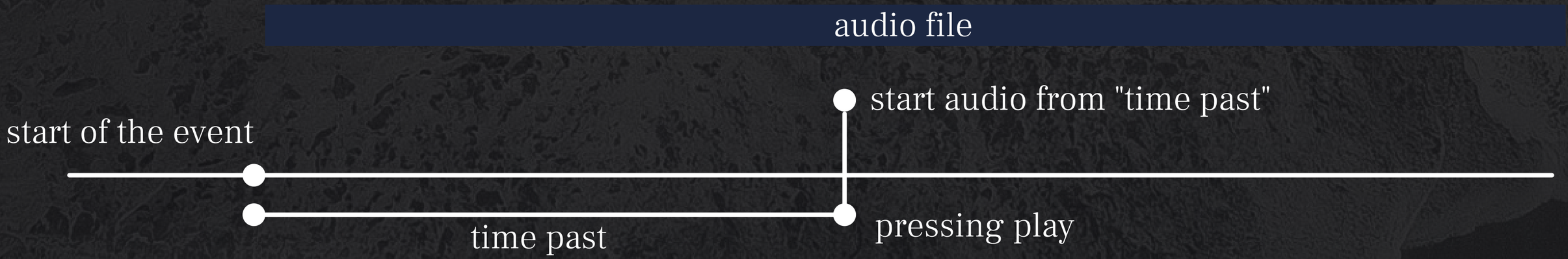
The trade-off is either to leave the user some sort of control for comfort (if panicking, if there is an issue, etc...) or to take it out.

During testing, it became clear that the participants would click on a button “to see what it does”. Rather than have the interaction, the button cannot stop the audio. But it is still the same principle of starting audio from the time past since the beginning of the event - in case the user refreshes the page or arrives late to the event.

The two states of "Audio" from the button are not symmetrical to the state of "no Audio": there is a specific CSS class to fill the button if there is audio playing.

There is no specific feedback to say that you can't pause the event once you have started. The idea is that the state of fill and the micro-copy "Live" (not being a call to action but a state) prevents the user from try to click on it.

Orpheus core function (visual)



Exploratory Phase

What were the roads not taken?
What solutions were explored but not chosen?

76

Unique Identifier for Users

Login

The main issue was to create an individual experience for the participants. It was discussed at first to have a login, and Orpheus would redirect you to your page (according to the tickets you have brought). It was better for people coming back and to keep track of the users. But Orpheus needed the least amount of clicks to access the participant page.

Geolocalisation

The second option that was explored was “the secret event”. Rather than have a specific ticket, it would be a secret event, with a time and place that was revealed at the end. It could create an additional layer of immersion. It could also have helped to give roles out according to people’s position.

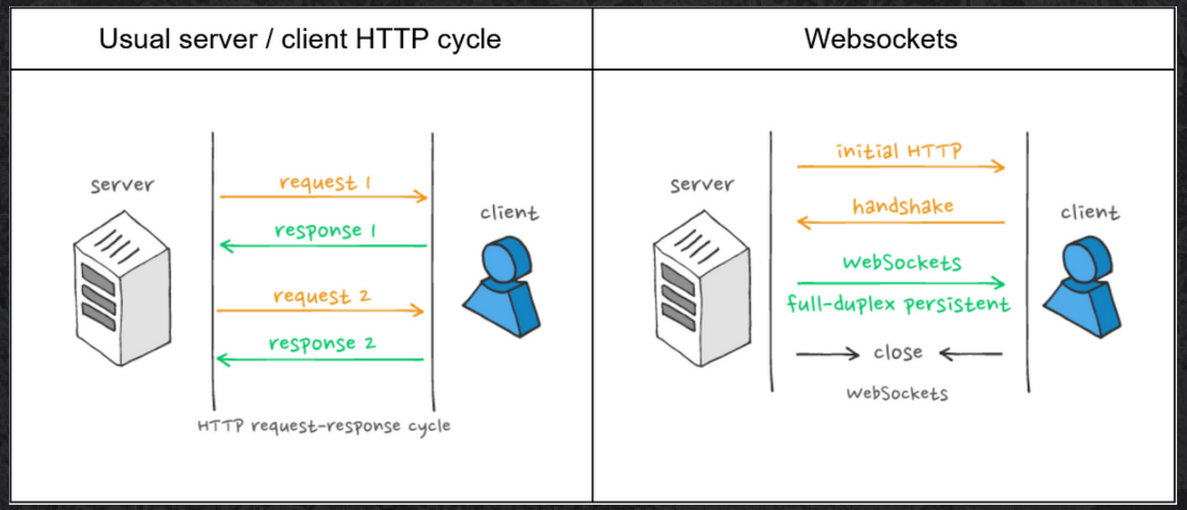
Geolocalisation is native to most browsers, which makes it very easy to use. But the geolocalisation API needs an Https connection - which Tamanoir does not currently have and would add to costs of the solution.

Anonymous token

It was also discussed to have a “queue” system for the solution. Rather than have the admin set the roles for the participants beforehand, it would be a list of roles, and the first person to connect would get the first role, second one, second role, etc… It was to heavy for both Orpheus and the users.

Streaming Audio

The first idea that comes to mind when talking about “simultaneously listening to audio” is radio. The participants would have their own set of streaming audio. First of all, streaming does not mean “live”, but the way data is downloaded. “Streaming media is multimedia that is delivered and consumed in a continuous manner from a source, with little or no intermediate storage in network elements. Streaming refers to the delivery method of content, rather than the content itself.” [source].



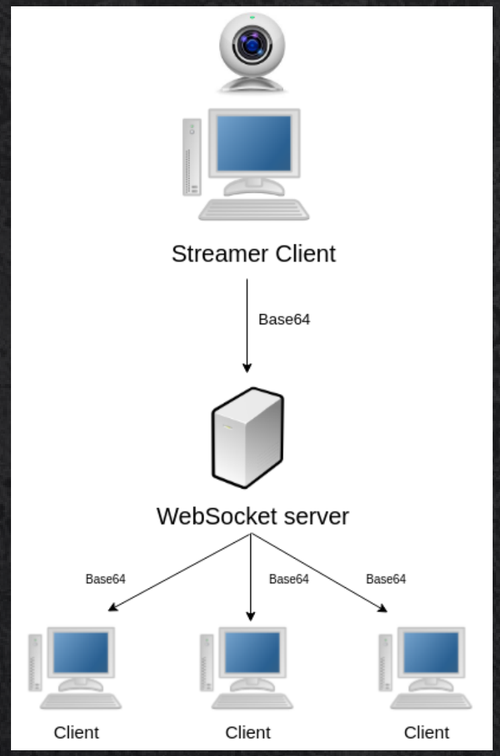
Comparing web sockets to http cycle

Web Socket

One of the foundations of streaming media is Websockets. It means that rather than have a back and forth between your client and the server, you open a persistent channel of communication.

Streaming (continued)

Live streaming is very rare in that sense. The examples are radio, or video or call chatting. For example, Twitch is a streaming service to show live video-game footage and webcam footage. Your webcam records you, the Twitch app records your screen, which are sent directly (with a buffer) to the Twitch servers. On the spectator's side, you'll have access to that specific part of the server, to see (in real-time) what is uploaded by the streamer's client.

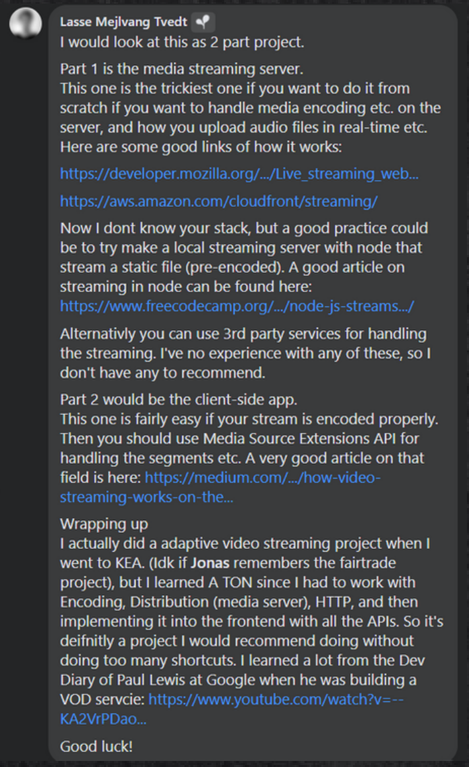


I had to ignore that solution because Tamanoir did not want to have a live Administrator. It would have been interesting to have a live administrator to help the participants, and to start the event - once everyone is present. But it limited the number of events that could be running at the same time, it limited the possibility for people to do it at home, and finally, it would have cost money.

Streaming (continued)

The second solution would be to have a Client-server streaming. It means that there isn't a Client streaming on one-end. Think Netflix: it streams video online, accessible through a login. There is not an operator starting your video when you want to look at the latest episode of Lucifer. I had to ignore that solution for two reasons:

1. It would require "Cloud functions" on the server for Firebase, to start the process of streaming, according to requests, which would also need to be in real-time. Cloud Functions are part of the Premium services of Firebase, which would add to the cost of Orpheus.
2. I could not do that much back-end coding to save my life. The whole process of understanding what is streaming, what is real-time / live, how would a server start streaming media on a timer, etc... took me a long time to understand what was happening. It did not even include starting to code it.



Lasse Mejlvang Tvedt

I would look at this as 2 part project.

Part 1 is the media streaming server.
This one is the trickiest one if you want to do it from scratch if you want to handle media encoding etc. on the server, and how you upload audio files in real-time etc. Here are some good links of how it works:

https://developer.mozilla.org/.../Live_streaming_web...

https://aws.amazon.com/cloudfront/streaming/

Now I dont know your stack, but a good practice could be to try make a local streaming server with node that stream a static file (pre-encoded). A good article on streaming in node can be found here: https://www.freecodecamp.org/.../node-js-streams.../

Alternativly you can use 3rd party services for handling the streaming. I've no experience with any of these, so I don't have any to recommend.

Part 2 would be the client-side app.
This one is fairly easy if your stream is encoded properly. Then you should use Media Source Extensions API for handling the segments etc. A very good article on that field is here: https://medium.com/.../how-video-streaming-works-on-the...

Wrapping up
I actually did a adaptive video streaming project when I went to KEA. (Idk if **Jonas** remembers the fairtrade project), but I learned A TON since I had to work with Encoding, Distribution (media server), HTTP, and then implementing it into the frontend with all the APIs. So it's deifnitly a project I would recommend doing without doing too many shortcuts. I learned a lot from the Dev Diary of Paul Lewis at Google when he was building a VOD servcie: https://www.youtube.com/watch?v=--KA2VrPDao...

Good luck!

While being very helpful, Lasse was only covering the "streaming audio" part of the problem, without even starting on the "synced on real-time between users".

Next time

What are the features that are missing from Orpheus?
What would make for a great version 2 of Orpheus?

Next iteration

Upload through CSV

Currently, the admin would have to manually add all the participants to an event. While it is doable (limited to 10 people) and serves as a verification process for the event going live - the next version needs to have an upload from the ticket providers (most likely CSV) with the email address, to be able to add all the participants at once. It questions the repartition of roles / URL for their audio, and among other things, is one of the reasons it was not designed this way yet.

Filters

As the solution grows bigger, there would be a lot more events, stories, etc... Which means that there is a need for filters and search: an admin could need to find a specific event or a specific user to edit their info. At the moment, there is no other way than scrolling to find what you are looking for.

The first design kept the events / stories in an overflow:scroll container. It kept the number of visible stories or events to 3 at any time. It was meant to not overwhelm the admin. But it also implied a secondary way to quickly find the information, aka filters.

Integrating Mail / MailChimp

MailChimp is part of the process of creating and setting up an event, but the user has to manually open MailChip on the side. There, it can upload a CSV to add the participants (reinforcing the need for CSV upload on Orpheus). It would be smoother to have a link to MailChimp from Orpheus - or better, having MailChimp integrated within the solution.

Next iteration

Keeping Track of users / users profile

There is no current tracking of the users. Tamanoir needed an “analytics” page, where an admin could find all of the information about a user (including all of the events he / she participated in). Since there is no on-site admin, it is also important to have tracking of how the event went. There should be a tracking page per event of “time listened”, if everyone started online, if people tried the solution beforehand, etc…

Furthermore, there is currently no end to the events. An admin would have to delete the past events if they want the URL to stop working. Part of the end process could include an “ending phase” of the event, to ask for feedback, and not just have participants left on site.

Recurring events / ticketing integration

For the scaling of Orpheus, there is a chance that we implement a schedule, which would repeat every week or every month. Rather than have to create events one by one, it would be nice to create "series of events" which could be every Thursday, once a month, etc…

It could also be then possible to create a Calendar view and a ticketing system facing the participants outside of Orpheus, to go event quicker!

84

Conclusion

A frictionless Multi-device solution

Frictionless

For the participant in an immersive experience hosted by Orpheus, a whole new world is accessible through only two clicks. All Orpheus needs from a participant is an email address. It will be used to both send you the email and create your individual URL, containing all of the information needed for your experience.

The use of unique event id in the URL guarantees a source of security: outside users can't know it.

As a participant, you don't need to do anything that isn't clearly stated in the email sent to you once you buy a ticket. No login, no passwords, no token, ... All you have to do is get carried by the button on your unique page.

Real-time Synced

The second challenge to Orpheus was to have a synced experience between users. The solution was to sync all the users to the same outside point: the start of the event. It appears to be live, to take you exactly to where you need to be, from the participant's perspective. It is Orpheus counting every second the time past since the beginning of the event, and starting your individual audio there.

No need for an on-site admin, for someone to start the event, for the user to do anything specific. Orpheus will send you on your adventure exactly where and when you need to be.

… but a small pandemic

A collective experience?

A lot of Orpheus also felt through the cracks of the pandemic. It was illegal to meet in groups in France when we were ready to test Les Naufragés and Orpheus. It was complicated to think about creating a 10 to 20 people physical experience, with touching and interacting, while the world was going into lock-down.

Orpheus shifted to a more intimate mood: I used a two-people experience (Le Continent Intérieur) that could be set-up at home, rather than have the epic 20 people Les Naufragés was supposed to be. It showed that Orpheus was a platform that could adapt, but it felt sad to post-pone most of the project.

More things to come

But Orpheus is an open-ended project. As it is hosted and created now, it offers the possibility for infinite scaling (and creating revenue with tickets rather than grant). It marked a change for Tamanoir's project: Orpheus is not content but structure. And it was interesting to help them with it. It is a "all-in-one" platform to create and share audio experiences. The possibilities are endless now!

I am looking forward to the content created for Orpheus, and how the platform will evolve. Creating a sense of mystery and anticipation on the participant page is a great project.

Conclusion

On working on Orpheus

Orpheus is first and foremost a technical "proof of concept". Its destiny is to be presented for a grant from a theater, a bar or a museum . And it that sense, Orpheus never really belonged to me. It was a challenge.

"If we wanted to make that happen, could you...?" and I am glad to know that I can. It opens the door to so many immersive experiences, to improvements and changes. I am glad to finish my Multimedia designer degree with Orpheus. It answered my own challenges: can I become a web developer?

I was presented with a puzzle, and it feels nice to have a solution to offer.