



BadUI

Polytech Angers
4A SAGI

Katia Jagueneau - Léo Moncoiffet

Janvier - Avril 2024

Sommaire

1	BadUI	3
1.1	Résumé	3
1.2	Summary	3
1.3	Keywords :	3
2	Introduction du Projet	4
3	Organisation	5
3.1	Logiciels utilisés	5
3.1.1	Trello	5
3.1.2	Github	6
3.1.3	OneDrive	6
4	Technologies utilisées	7
4.1	Logiciels	7
4.1.1	Replit	7
4.1.2	Visual Studio Code	7
4.1.3	Xampp	8
4.1.4	Canva	8
4.2	Langages de programmation	8
4.2.1	Client-Side	8
4.2.2	Server-Side	8
5	Exemples de BadUI existants	10
5.1	Poubelle	10
5.1.1	Présentation	10
5.1.2	Bypass	11
5.2	Date Picker	13
5.2.1	Présentation	13
5.2.2	Bypass	14
5.3	Jeu du Fakir	15
5.3.1	Présentation	15
5.3.2	Bypass	15
5.4	Calendar Hell	16
5.4.1	Présentation	16
5.4.2	Bypass	16
5.5	Subscription	18
5.5.1	Présentation	18
5.6	Manual Transmission	18
5.6.1	Présentation	18

5.6.2	Bypass	18
5.7	Conclusion	20
6	Nos BadUIs	21
6.0.1	Interface graphique	21
6.0.2	Numbertragedy	22
6.0.3	Phonecall	24
6.0.4	Labymage	25
6.0.5	BadForm	27
7	Problèmes & Résolutions	28
7.0.1	Changement de Logiciel	28
7.0.2	Bypass	28
7.0.3	Maladie	28
7.0.4	Expérience en PHP/JavaScript	29
7.0.5	Méthodes de travail	29
7.0.6	Divergence des compétences	29
8	Nos autres idées de BadUI	30
9	Conclusion du Projet	32
9.1	Fin du Projet	32
9.2	Ce que nous avons appris	32
9.3	Commentaires personnels	33
9.3.1	Léo	33
9.3.2	Katia	33
10	Remerciements & bibliographie	34
10.1	Remerciements	34
10.2	Bibliographie / Sitographie	34

BadUI

1.1 Résumé

Ce rapport porte sur le projet 4A SAGI nommé "BadUI" qui consiste à créer un site internet comportant des interfaces utilisateur intentionnellement difficiles à utiliser ou mal faites. Nous avons utilisé HTML/CSS et JavaScript pour le développement front-end et du PHP pour le développement back-end. Il était aussi important de rendre ces interfaces non-contournables ou aussi difficilement contournable que d'utiliser l'interface elle-même. Nous avons été deux à travailler sur ce projet, Katia Jagueneau et Léo Moncoiffet.

1.2 Summary

This report is about the 4th year SAGI project named "BadUI" in which we had to make a website that contained several bad user interfaces. We could either make an interface hard to use or make it look bad by example. We used HTML/CSS and JavaScript for the front-end coding and PHP for the back-end. It was also an important part of the project to not let user bypass easily our bad UIs, it had to be at least as hard to bypass the user interface as using it directly. We were two on the project, Katia Jagueneau and Léo Moncoiffet.

1.3 Keywords :

- Client-Side
- Server-Side
- User Interface
- JavaScript
- PHP
- HTML/CSS
- Bypass
- Cookie

Introduction du Projet

Projet BadUI :

Ce projet s'inscrit dans le cadre du deuxième semestre en 2 ème année du Cycle Ingénieur SAGI à Polytech Angers. Nous devons à travers ce projet créer un site et créer par nous-mêmes des interfaces utilisateur web qui sont, soit difficilement utilisables soit pas bien faites esthétiquement. Il est important pour ces interfaces qu'elles ne soient pas bypassables facilement par l'utilisateur non plus.

Le but de ce projet est d'apprendre ce qui peut faire d'une UI une "mauvaise" UI mal adaptée pour les utilisateurs. Apprendre à faire de mauvaises interfaces utilisateur nous a permis d'apprendre les erreurs qui peuvent être commises et ce qu'il ne faut surtout pas faire, c'est une approche assez unique car il s'agit de faire des erreurs intentionnellement et de créer des interfaces à partir de ce qui pourrait être considéré comme des erreurs dans un cadre normal.

A travers ce rapport, vous allez pouvoir apprendre ce qui rend une interface utilisateur mauvaise avec des exemples et des explications. Nous allons aussi observer à travers ces exemples comment il est possible de contourner certaines interfaces pour qu'elles ne soient plus si difficiles à utiliser. Puis nous verrons ensuite plus précisément le projet que nous avons mené avec nos méthodes, nos propres Bad UIs, et comment nous nous y sommes pris pour les rendre plus difficiles à contourner.

Organisation

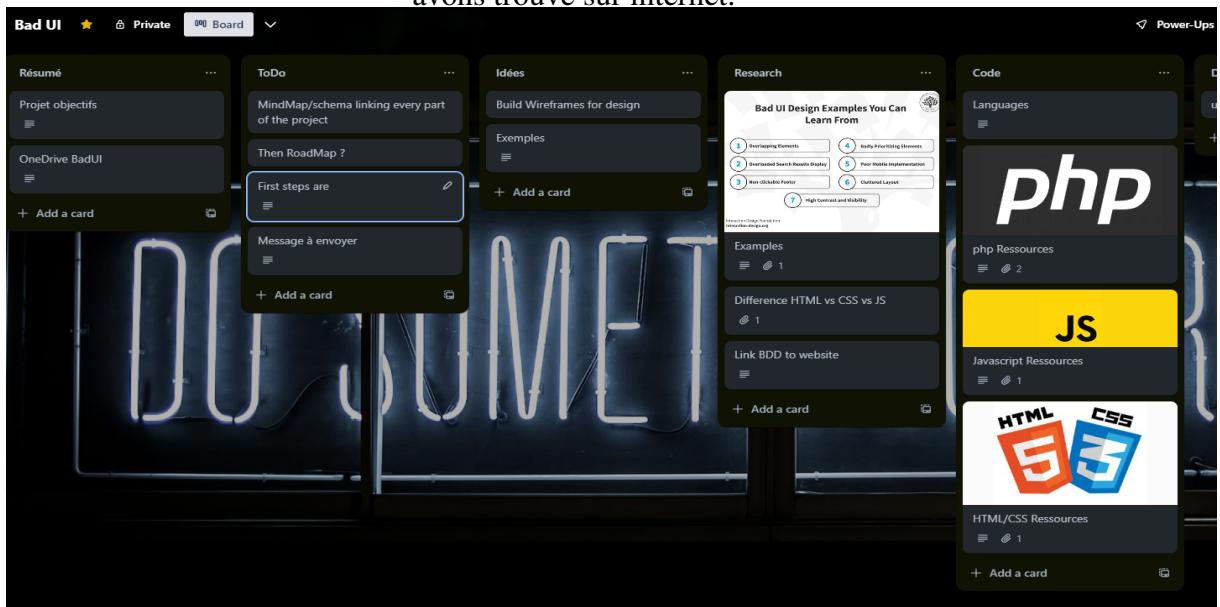
3.1 Logiciels utilisés

Nous avons utilisé plusieurs logiciels au cours de notre projet pour l'organisation et l'hébergement de fichier.

3.1.1 Trello



Nous avons premièrement utilisé Trello, qui est un site web qui peut aussi être téléchargé en tant qu'application et permet de créer des tableaux de bord où il est possible de mettre des informations, des liens, ou encore des dates limites pour certaines tâches. Nous nous en sommes surtout servi pour communiquer nos idées et y mettre les ressources que nous avons trouvé sur internet.



On voit ci-dessus notre dashboard où nous avons différentes cartes avec des informations (descriptions, liens, textes, checklists,...) dedans. Cela nous a permis de nous organiser, de partager différentes ressources dont nous avions besoin ou encore d'y écrire des idées.

3.1.2 Github



Nous avons eu recours à Github pour sa structure afin de bien organiser notre code. Il nous a fallu un moment avant de bien comprendre et de bien l'utiliser dans le projet, mais cela nous a été très utile, notamment pour coder chacun de notre côté et facilement fusionner ce que nous avions fait ensuite ou encore revenir en arrière quand nous avions besoin.

Il s'agit d'une plateforme pour développeurs qui permet de stocker des versions de code et ensuite de naviguer entre elles, et de fusionner ce qu'on appelle des "branches" de code. Cela permet de coder à plusieurs et d'ensuite fusionner les morceaux de code plus facilement.

3.1.3 OneDrive



Nous nous sommes servis du logiciel OneDrive avec un dossier partagé depuis nos comptes universitaires afin de se partager et de stocker certains fichiers et certaines images pour le projet. Il ne nous a pas été d'une énorme utilité sur notre projet, mais si nous avions l'occasion de continuer le projet, nous l'aurions sans doute utilisé plus souvent en complexifiant le projet.

Technologies utilisées

4.1 Logiciels

4.1.1 Replit



Afin de coder nos BadUIs, nous avons commencé par utiliser Replit. Ce logiciel propose un environnement de développement intégré sur son site web. C'est avec cet environnement que nous avons travaillé pendant les TDs dans le cours de programmation web que nous avons eu sur la même période. Nous avons cependant eu des problèmes lorsqu'il a fallu faire fonctionner le code PHP qui, comme je vais l'expliquer plus tard, est le language que nous avons choisi pour le développement back-end. En effet, il fallait choisir un template PHP pour pouvoir le faire fonctionner, sauf que dans ce cas là nous n'avions plus e code html/CSS de disponible directement. Nous avons finalement décidé d'abandonner l'idée de faire fonctionner notre projet depuis cet environnement et avons utilisé les deux technologies suivantes. Par ailleurs, nous n'avons finalement pas utilisé Replit pour nos projets web non plus.

4.1.2 Visual Studio Code



Afin de pallier aux problèmes rencontrés sur Replit, nous avons migré sur Visual Studio Code en couplant ce logiciel avec Github comme mentionné plus tôt, afin d'y apporter une flexibilité et une maîtrise sur notre code assez conséquente, même pour un projet de deux personnes. Nous avons ajouté les différentes extensions nous permettant de coder notre projet. Voici quelques exemples d'extensions que nous avons ajouté :

- PHP
- json
- HTML CSS Support
- GitHub Pull Requests

Nous conseillons à tout étudiant faisant son projet sur VSC de bien utiliser le "source control" afin d'utiliser Git directement depuis VSCode. Cela devient alors très facile de commit, push et pull votre code depuis votre Github, et vous pouvez même gérer les merges depuis VSCode, et ce même si nous avons utilisé le site Github directement pour gérer les conflits.

4.1.3 Xampp



Finalement, nous avons utilisé xampp, et plus précisément Apache pour héberger le site de notre projet localement. Nous avons aussi utilisé MariaDB database pour héberger temporairement une base de donnée sur notre projet, avant de décider que cela n'était finalement pas nécessaire, puisqu'il n'y avait pas de données importantes à stocker sur notre projet. Nous sommes donc revenus en arrière sur ce point au cours du projet, même si la base de données était fonctionnelle.

4.1.4 Canva



Pour la création des visuels pour l'UI labyrinthe, que nous expliquerons plus en détail plus tard dans ce rapport, nous avons choisi d'utiliser canva, une solution complète, rapide et facile d'utilisation pour la confection des murs et fond des labyrinthes que nous avons implémentés.

4.2 Langages de programmation

Nous allons maintenant expliquer rapidement nos choix de langage de programmation pour ce projet.

4.2.1 Client-Side

Pour le côté client, c'est-à-dire le côté utilisateur, ce que son navigateur web va recevoir afin de générer les pages, nous avons choisi forcément HTML/CSS, parce qu'il s'agit des langages standards et que nous avons appris en partie à utiliser ces langages lors de nos cours à Polytech. Nous avons couplé cela à du JavaScript pour le code côté client, car il s'agit d'un langage simple et répandu pour ajouter de l'interactivité au site. C'est aussi ce langage qui va permettre à l'utilisateur depuis les pages de communiquer avec le serveur sous forme de requêtes.

4.2.2 Server-Side

Ensuite, pour le côté serveur, nous avons choisi d'utiliser PHP car nous étions tous les deux en semestre à l'étranger quand le cours de PHP a été donné à Polytech Angers.

Nous nous sommes donc dit qu'il serait intéressant de profiter de ce projet pour regarder ce langage d'un peu plus près et d'en apprendre au moins les rouages, ce qui n'a pas été simple au départ surtout avec la syntaxe. Cependant, nous nous y sommes adaptés au fur et à mesure. Cette programmation côté serveur nous a principalement permis de vérifier les informations que l'utilisateur souhaitait envoyer. Il est important, comme il sera expliqué plus tard, de ne pas tout laisser côté client, sinon l'utilisateur peut changer le code à sa guise, et donc le fonctionnement du site. Cette vérification côté serveur permet de ne pas le laisser l'utilisateur si facilement "tricher" ou contourner ce que nous avons construit.

Exemples de BadUI existants

Pour démarrer notre projet, nous avons choisi de chercher de l'inspiration avec de nombreuses sources disponibles en ligne, sur différents réseaux sociaux. En effet, nous sommes loin d'être les premiers à travailler sur cette idée. Il a donc été facile de trouver des idées de BadUI, que nous allons vous présenter ci-dessous.

5.1 Poubelle



5.1.1 Présentation

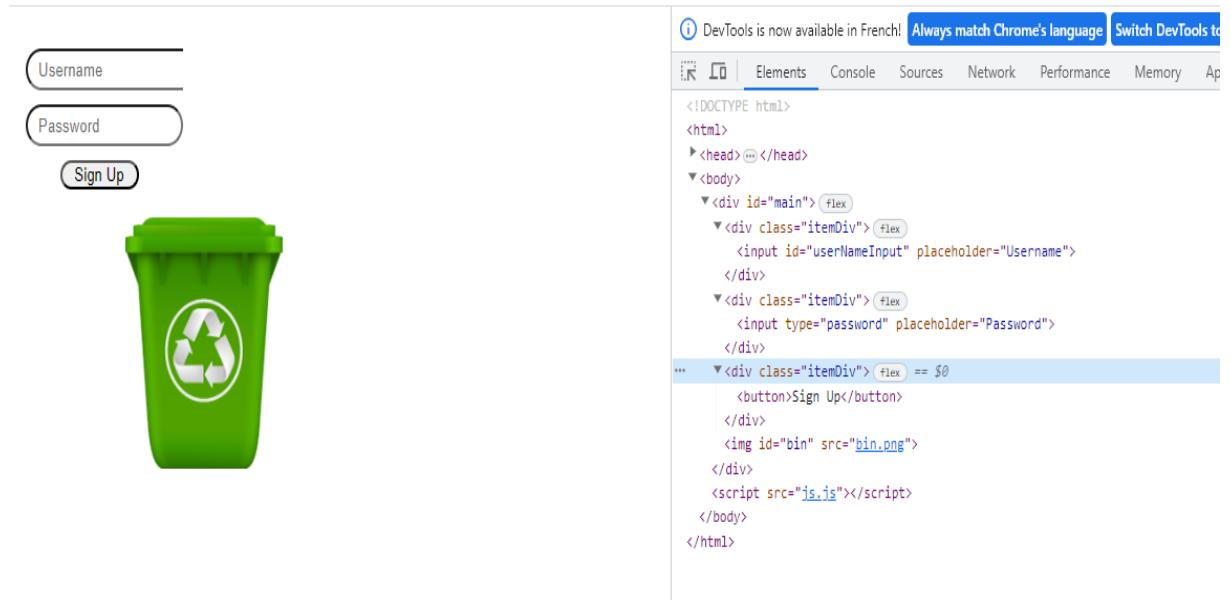
Cette première Bad UI consiste à limiter le nombre de caractères que l'on peut entrer dans la section qui permet d'insérer du texte. En effet, comme on peut le voir, les lettres « tombent » dans la poubelle. Cependant, cette première Bad UI, contrairement aux attendus de notre projet, est assez facilement bypassable.

On peut trouver cette Bad UI en suivant le lien ci-après :
<https://i01000101.github.io/RedditBadUIBattles/LimitUsernameSize/badDesign.html>

5.1.2 Bypass

En effet, l'entièreté du code est coté client, ce qui permet de pouvoir contourner la difficulté aisément. Comme on peut le voir sur cette image, il suffit de changer un peu le javascript en passant par le devtools de votre navigateur.

Observons dans un premier temps le code html de la page.



```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <div id="main">
      <div class="itemDiv">
        <input id="userNameInput" placeholder="Username">
      </div>
      <div class="itemDiv">
        <input type="password" placeholder="Password">
      </div>
      <div class="itemDiv">
        <button>Sign Up</button>
        
      </div>
      <script src="js.js"></script>
    </div>
  </body>
</html>
```

Il n'y a rien d'intéressant pour nous, à part peut être les classes des divs et l'id 'userNameInput'. Ces informations peuvent nous aider à ensuite comprendre comment le JavaScript influe sur la page en fonction de nos actions. En parlant de JavaScript, allons regarder ce qu'il en est.

```

var input = document.getElementById('userNameInput')
var y = input.getBoundingClientRect().top + 10
var x = input.getBoundingClientRect().left + 150

var mainDiv = document.getElementById('main')

var fallingDivs = []

input.addEventListener('input', () => {
    while(isOverflown(input)){
        let fallingChar = input.value.substr(-1)
        input.value = input.value.slice(0,-1)

        let fallingDiv = document.createElement('div')
        fallingDiv.innerText = fallingChar

        fallingDiv.classList.add('fallingDiv')
        fallingDiv.style.top = y +'px'
        fallingDiv.style.left = x + 'px'

        mainDiv.append(fallingDiv)
        fallingDivs.push(fallingDiv)
    }
})

function isOverflown(element) {
    return element.scrollWidth > element.clientWidth;
}

setInterval(() => {
    fallingDivs.forEach(fallingDiv => {
        if(Number(fallingDiv.style.top.slice(0,-2)) < document.getElementById('bin').getBoundingClientRect().top + 10){
            fallingDiv.style.top = Number(fallingDiv.style.top.slice(0,-2)) + 5 + 'px'
            fallingDiv.style.transform = 'rotate(' + (Number(fallingDiv.style.transform.slice(7,fallingDiv.style.transform.indexOf('d')))+5)+deg)'
        }
    })
},40)

```

Il y a pas mal d'informations ici. Déjà, nous pouvons voir que le JavaScript récupère l'élément comprenant l'id 'userNameInput' qui est l'élément principal de ce BadUI. Il y a ensuite un 'event listener' de type 'input' qui permet à JavaScript de faire une action à chaque fois que la valeur change. A chaque fois que l'utilisateur écrit ou enlève une lettre dans la barre de texte, la fonction qui suit est lancée.

Il y a donc un test pour savoir si la lettre dépasse le cadre imposé, et dès qu'on arrive ici on voit qu'il y a quelque chose d'important. Que se passerait-il si cela n'était jamais le cas ? Nous pouvons alors directement modifier la fonction 'isoverflown' en mettant return false.

Cela nous permet donc simplement de ne jamais rentrer dans la boucle while et de débloquer la case. Il est aussi possible de changer directement dans le code html la valeur du div, mais cela ne règle pas le problème si l'on souhaite rajouter d'autres lettres ensuite.

```

function isOverflowed(element) {
    return element.scrollWidth > element.clientWidth;
}

function isOverflowed(element) {
    return false;
}

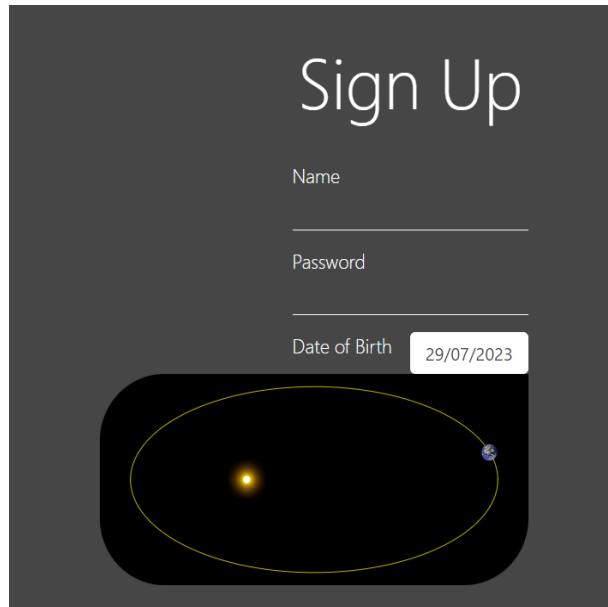
input.addEventListener('input', () => {
    while(isOverflowed(input)){
        let fallingChar = input.value.substr(-1)
        input.value = input.value.slice(0,-1)

        let fallingDiv = document.createElement('div')
        fallingDiv.innerText = fallingChar
        fallingDiv.classList.add('fallingDiv')
        fallingDiv.style.top = y +'px'
        fallingDiv.style.left = x + 'px'

        mainDiv.append(fallingDiv)
        fallingDivs.push(fallingDiv)
    }
})

```

5.2 Date Picker



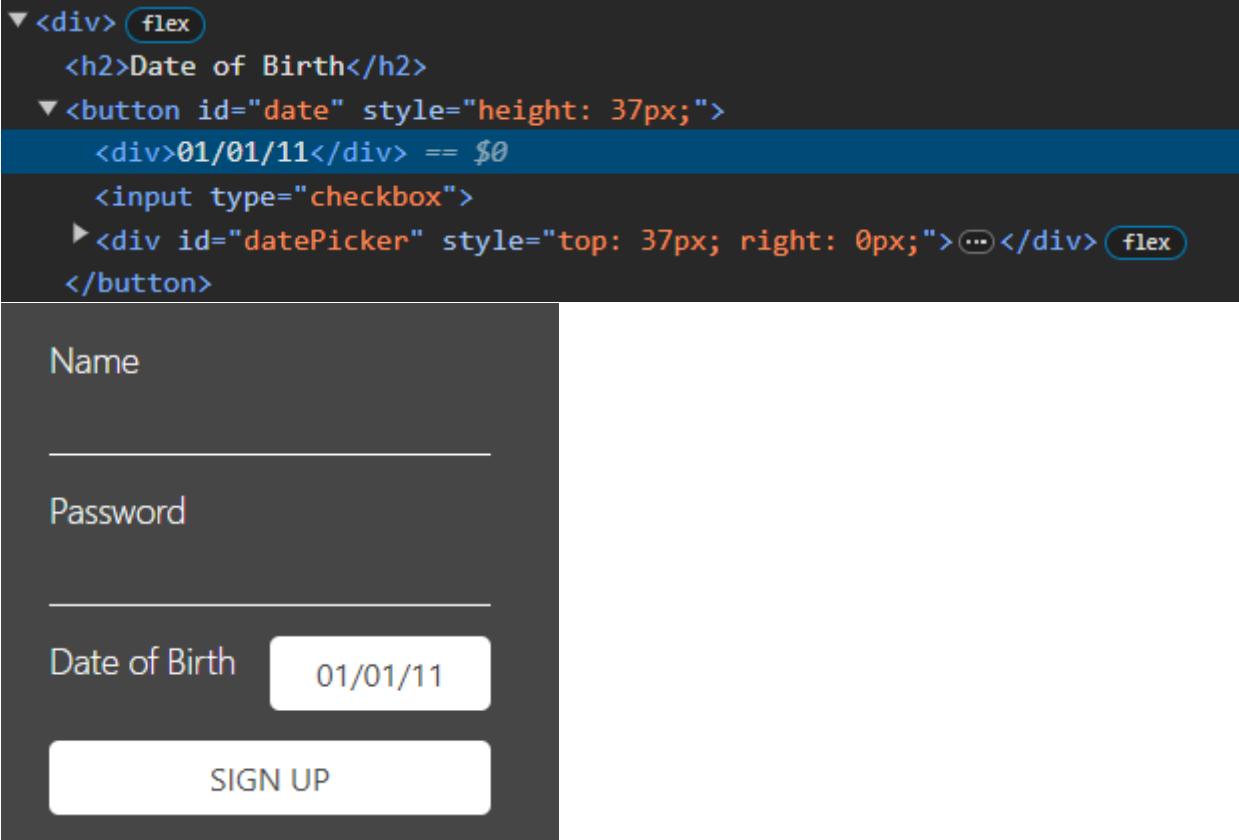
5.2.1 Présentation

La difficulté de cette UI ne réside non pas dans le texte mais cette fois-ci dans le choix d'une date. En effet, pour sélectionner la date, il faut non pas la sélectionner sur un calendrier mais faire tourner la Terre autour du soleil pour retrouver sa configuration au jour que nous souhaitons choisir.

On peut trouver cette Bad UI en suivant le lien ci-après :
<https://i01000101.github.io/RedditBadUIBattles/ScientificDatePicker/index.html>

5.2.2 Bypass

De la même manière que l'UI précédente, tout le code se trouve côté front. Il est donc aisément de le bypasser : Il suffit juste de changer la valeur html, car c'est la seule chose qui est changée depuis le côté JavaScript. Il n'y a pas de vérification, donc il suffit de changer cette valeur sans avoir besoin de passer par l'interface.



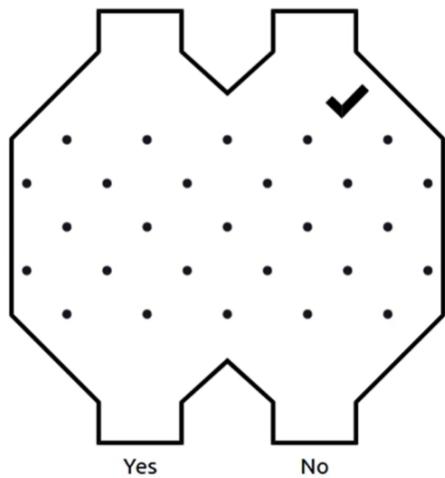
The screenshot shows a mobile application's sign-up interface. At the top, there is a code snippet in a dark-themed code editor:

```
▼ <div> flex
  <h2>Date of Birth</h2>
  ▼ <button id="date" style="height: 37px;">
    <div>01/01/11</div> == $0
    <input type="checkbox">
  ▶ <div id="datePicker" style="top: 37px; right: 0px;">...</div> flex
</button>
```

Below the code is the application's UI. It consists of three input fields: "Name", "Password", and "Date of Birth". The "Date of Birth" field contains the value "01/01/11". To the right of the date input is a small white button with the text "01/01/11" on it. At the bottom is a large white button labeled "SIGN UP".

5.3 Jeu du Fakir

Would you like to delete your account?



5.3.1 Présentation

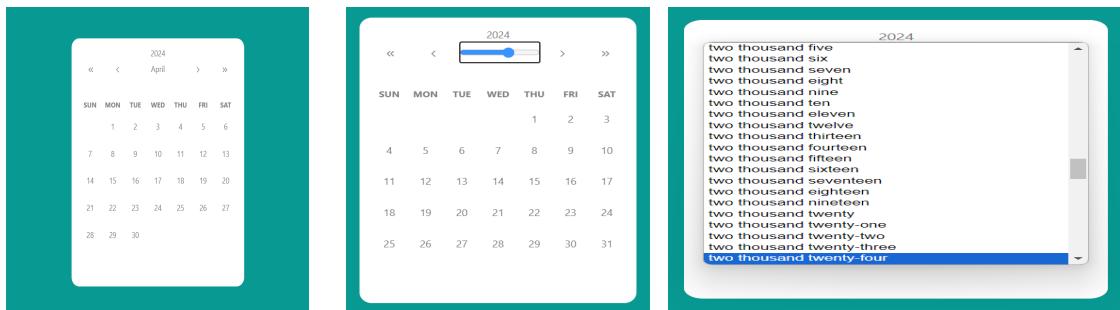
Dans cette UI, il s'agit d'un choix à faire entre 2 cases, et lorsque le choix est validé, un jeu du fakir se lance pour sélectionner la réponse finale, qui peut être différente de la réponse donnée précédemment.

On peut trouver cette Bad UI en suivant le lien ci-après :
<https://im.ocean.lol/choose/set>

5.3.2 Bypass

En observant le BadUI après plusieurs itérations, nous pouvons remarquer qu'il y a toujours le même résultat : Si on choisit "yes" au début, alors on tombe sur "No" et inversement. Il suffit juste de mettre la réponse inverse pour obtenir le résultat que l'on souhaite. Il n'y a pas vraiment de contournement à faire pour ce BadUI car il n'y a pas vraiment de finalité dans le code JavaScript.

5.4 Calendar Hell



5.4.1 Présentation

Il s'agit là encore d'une Bad UI qui exploite le plein potentiel des calendriers. A première vue, il s'agit d'un calendrier classique, mais si on regarde de plus près, les jours de la semaine ne sont pas dans l'ordre, pour choisir les mois, il faut déplacer un sélecteur sur une barre, sans voir le mois sélectionné, et pour l'année, il faut choisir avec des nombres écrits en toutes lettres.

On peut trouver cette Bad UI en suivant le lien ci-après :
<https://goulartnogueira.github.io/BadUI/calendar-hell/>

5.4.2 Bypass

Commençons notre aventure de bypassing avec l'année. Nous ne pouvons pas sélectionner une année inférieure à 1900 ou supérieure à 2099 dans la liste proposée. En allant regarder le code JavaScript, nous pouvons nous apercevoir que l'input donné par la sélection est ensuite redirigé dans une fonction "setYear(year)". Nous pouvons alors directement appeler cette fonction avec l'année que nous souhaitons choisir. Nous pouvons aussi choisir le mois par le même procédé.

Il suffit ensuite d'utiliser la commande "update()" pour mettre à jour les jours proposés sur le calendrier en fonction de l'année et du mois choisi. Nous pouvons mettre l'année directement dans la fonction, mais pour le mois il faut faire attention car le code JavaScript va directement récupérer dans le tableau des mois, donc il faut mettre le bon index, 0 à 11.

```
> setYear(1223)
< 1223
> setMonth(3)
< 3
> setMonth(5)
< 5
> update()
< undefined
> |
```

```

function setYear(year) {
    date.setFullYear(year);
    form.year.parentElement.dataset.value = year;

    return year;
}

function setMonth(month) {
    date.setMonth(month);
    form.month.parentElement.dataset.value = months[month];

    return month;
}

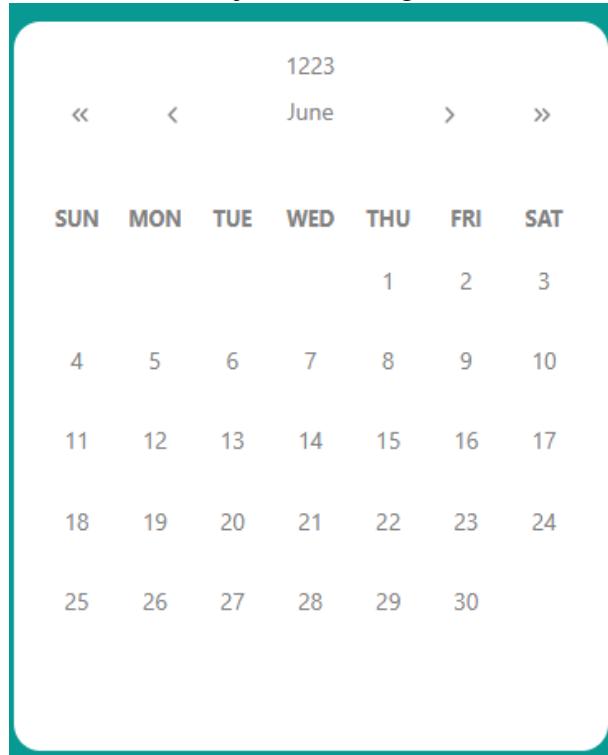
function update() {
    const startOfMonth = new Date(date.getFullYear(), date.getMonth());
    const endOfMonth = new Date(date.getFullYear(), date.getMonth() + 1);
    const paddingSpan = (startOfMonth.getDay() + offset) % 7;
    endOfMonth.setDate(0);

    const week = weekdays.map((day) => `<span>${day}</span>`).join("");
    const padding = paddingSpan > 0 ? `<span style="grid-column-end: span ${paddingSpan}"></span>` : "";
    const days = [...Array(endOfMonth.getDate())]
        .map((_, i) => {
            return `<button data-day="${i + 1}" type="button"></button>`;
        })
        .join("");

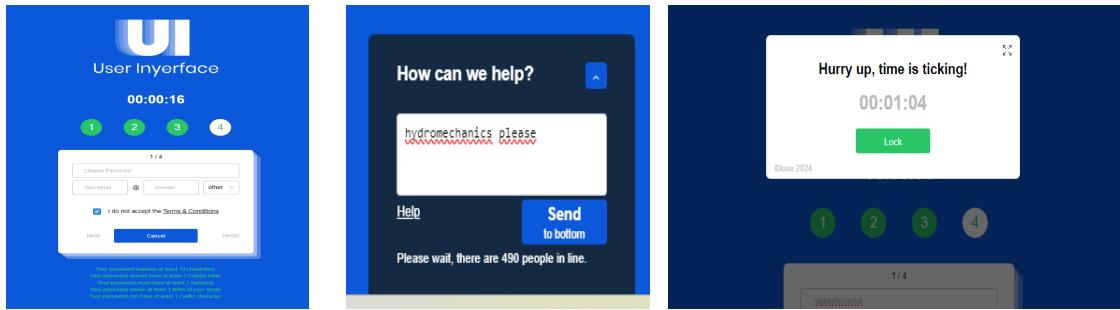
    month.innerHTML = `${week}${padding}${days}`;
}

```

Nous pouvons alors voir le résultat. Nous n'avons pas eu besoin de toucher à l'interface et en seulement trois commandes, nous pouvons changer l'année et le mois du calendrier tout en mettant à jour l'affichage de ce dernier.



5.5 Subscription



5.5.1 Présentation

Pour celle-là, il s'agit de s'inscrire sur un site internet, mais le mot de passe a beaucoup de critères à respecter, ce qui rend sa mémorisation compliquée. De plus, le temps est limité, sans quoi le site se bloque. Encore une fois, tout le code se trouve côté client, mais cette fois il est plus compliqué à bypasser, puisque le créateur s'est amusé à glisser des blagues dans le code, qui fait plus 11000 lignes. De plus, chaque petite fonctionnalité contient un paramètre qui ne facilite en rien l'expérience utilisateur, comme par exemple, la case d'aide en bas à droite, qui est impossible à remplir puisque les mots se changent dès qu'on presse la barre espace, lorsqu'on clique sur le bouton pour abaisser la case, elle s'abaisse mais très lentement et réapparaît peu de temps après, et lorsqu'on clique sur « help », un message « please help, there are 410 people in line » dont le nombre s'incrémente à chaque clic apparaît. De même, avant la fin du temps imparti, un pop-up impossible à fermer s'affiche, et le temps défile en sens inverse. En outre, le site comprend beaucoup de petites fonctionnalités conçues pour rendre l'expérience utilisateur frustrante et compliquée.

On peut trouver cette Bad Ui en suivant le lien ci-après :
<https://userinyerface.com/>

5.6 Manual Transmission

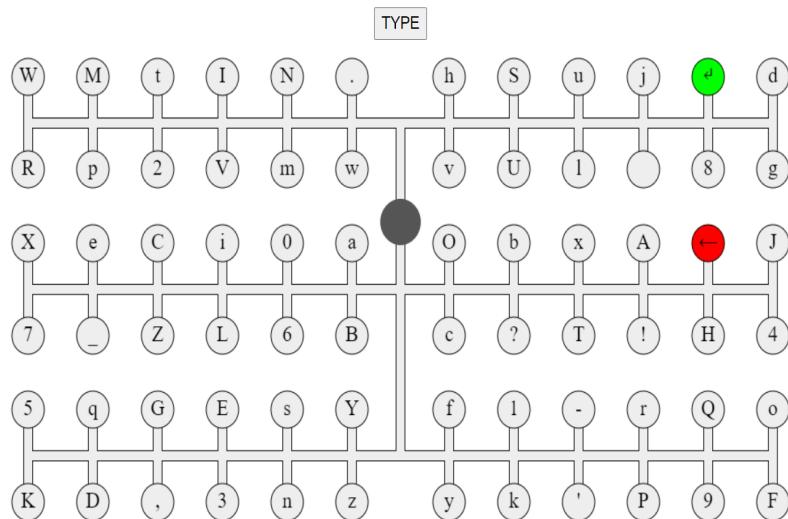
5.6.1 Présentation

Ici, il s'agit de créer un message, non pas en tapant sur le clavier, mais en faisant bouger ce qui imite un levier de vitesse, ce qui rend l'écriture du message très longue et fastidieuse.

On peut trouver cette Bad Ui en suivant le lien ci-après :
<https://user-789.github.io/GearBoard/>

5.6.2 Bypass

L'interface utilisateur est très handicapante ici, mais en regardant le code, on s'aperçoit qu'il n'est pas si compliqué dans le fond.



```

function typeSelected() {
    if (selectedChar == "") {
        return;
    } else if (selectedChar == BACKSPACE) {
        writtenChars.pop();
    } else if (selectedChar == ENTER) {
        alert(writtenChars.join(""));
        writtenChars = [];
        input.innerHTML = "Type here whatever you want!";
        return;
    } else {
        writtenChars.push(selectedChar);
    }
    input.innerHTML = writtenChars.join("");
}

```

On peut voir ici la partie la plus importante du code. Il s'agit de la fonction qui est lancée quand on appuie sur "type". On remarque rapidement que "writtenchars" est un tableau de caractères (char) et que c'est cette variable qu'il faudra que l'on change pour que le résultat soit changé dans le code JavaScript. S'il y avait une connection à un serveur quelconque, cette valeur pourrait être celle renvoyée à la fin. Il suffit donc de changer "writtenchars" puis de changer l'"innerhtml" pour obtenir le résultat que l'on souhaite obtenir.

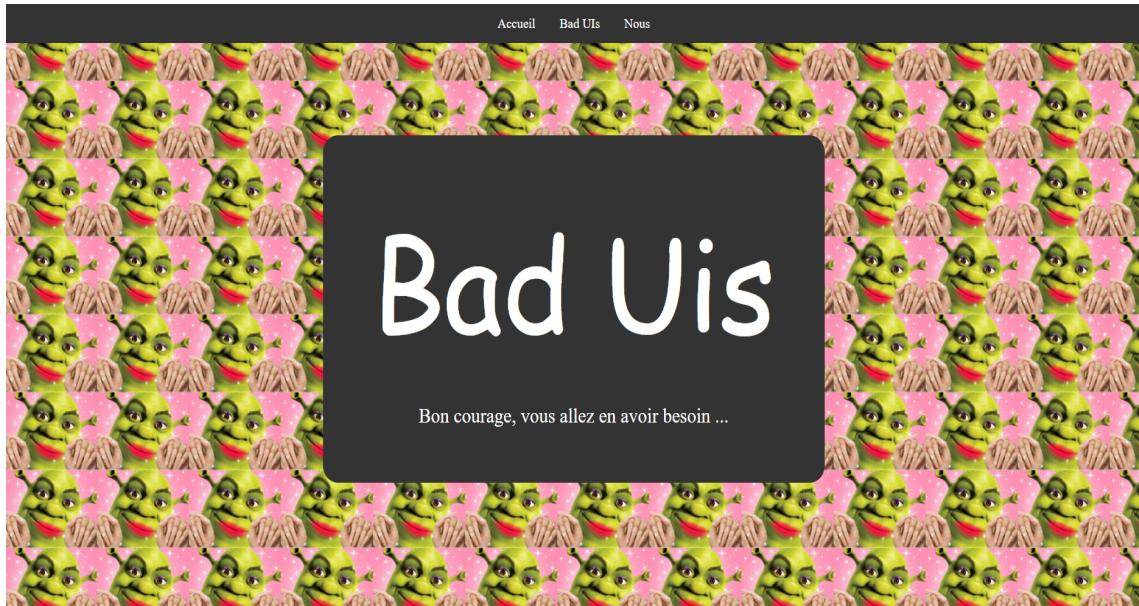
```
> writtenChars.push('a')
← 5
> writtenChars.push('b')
← 6
> writtenChars.push('d')
← 7
> input.innerHTML = writtenChars.join("");
← 'wwwabbd'
>
```

5.7 Conclusion

Comme nous avons pu le constater, il existe un nombre incalculable de Bad UIs différentes. D'autant plus avec la création du r/badUIbattles sur reddit par un célèbre tiktoker qui se spécialise dans le code, beaucoup de développeurs ont profité de cette opportunité pour laisser libre cours à leur imagination. C'est sur ce subreddit que nous avons, en majorité puisé nos idées. Cependant, la difficulté de notre projet ne réside pas uniquement dans l'imagination, mais aussi dans la possibilité de bypasser ces Bad Uis. En effet, pour la plupart d'entre elles, le code entier se trouve côté client, ce qui rend le bypass plus simple. De notre côté, il fallait le rendre compliqué, voire impossible.

Nos BadUIs

6.0.1 Interface graphique



La première étape de notre travail a été de penser à la forme que prendront nos idées. En effet, comme évoqué au début de ce rapport, une BadUI se caractérise dans le côté non-fonctionnel mais aussi dans le côté (peu) esthétique. C'est pour cela qu'ne feuilletant des inspirations pinterest et tiktok et en faisant fonctionner notre imagination. Nous sommes donc partis sur une interface dans le thème de Shrek revisité, en mosaïque pour créer une atmosphère inconfortable. Elle est cependant optimisée pour toutes les tailles d'écran.

6.0.2 Numbertragedy



Notre première réalisation de BadUI, Numbertragedy, s'est avérée être un exemple relativement simple. Dans un premier temps, notre objectif était de permettre à l'utilisateur d'ajuster un nombre à la hausse et à la baisse, tout en reflétant ces changements côté serveur. Ensuite, nous avons jugé nécessaire d'empêcher l'utilisateur d'entrer directement une valeur numérique, le contraignant ainsi à utiliser nos boutons dédiés pour ajuster le nombre. Cette restriction était cruciale pour maintenir l'intégrité de notre BadUI, car l'envoi continu de la valeur du nombre à chaque clic utilisateur aurait permis une manipulation facile de celle-ci, contrecarrant ainsi l'essence même de notre concept.

Les boutons destinés à l'augmentation et à la diminution du nombre sont inversés, ajoutant ainsi une complexité supplémentaire à l'exercice. De plus, nous avons envisagé la possibilité de renverser l'ordre des boutons toutes les 5 secondes, pour compliquer davantage l'expérience de l'utilisateur, mais cette possibilité n'ayant été évoquée qu'à la fin du projet, nous avons décidé de ne pas l'implémenter.

Dans un second temps, nous avons ajouté des boutons supplémentaires en haut de l'interface, censés introduire un niveau de difficulté supplémentaire. Cependant, ces ajouts sont superflus, car les deux premiers boutons ne font que modifier le texte affiché au-dessus, tandis que le troisième bouton n'a pas d'effet différent de celui résultant de l'action directe de l'utilisateur sur les boutons initiaux.

Il est aussi important de noter que nous avons changer la durée de vie du cookie session, donc au bout de 30 secondes, si l'utilisateur n'a pas "sauvegardé" son nombre, il perd sa progression.

Les nombres qui sont validés sont stockés côté serveur dans un fichier json.

Regardons d'un peu plus près le code pour comprendre le côté technique :

```

function ChangeNumber(change) {
    console.log("Changing the Number");
    const formData = new FormData();
    formData.append('operation', change);
    fetch('php/numbertragedyservercode.php', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {console.log(data);document.getElementById("numberField").innerHTML = data.number})
    .catch(error => console.error('Error:', error));
}

function sendToServer() {
    const formData = new FormData();
    formData.append('operation', 3);
    fetch('php/numbertragedyservercode.php', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {console.log(data);document.getElementById("valnum").innerHTML=data.validatenumber;})
    .catch(error => console.error('Error:', error));
}

function difficulty(difficulty) {
    const formData = new FormData();
    formData.append('operation', difficulty+3);
    fetch('php/numbertragedyservercode.php', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {console.log(data);console.log(data.difficultymessage);document.getElementById("difficulty_message").innerHTML=data.difficultymessage;})
    .catch(error => console.error('Error:', error));
}

```

Comme expliqué précédemment, nous pouvons nous apercevoir que le code JS ne contient que les fonctions qui permettent aux fonctionnalités graphiques du site de fonctionner et de récupérer les valeurs transmises par le côté serveur. C'est du côté serveur que le code est plus intéressant :

```

<?php
if ($_SERVER['REQUEST_METHOD'] == "POST") {
    $operation = $_POST['operation'];
}

if($operation!=4 and $operation!=5){
    $oldgetsessionstart=session_start(['cookie_lifetime' => 30,'name'=>'YouShouldn\'tbelookinghere']);
    if (!isset($_SESSION['count'])) {
        $_SESSION['count'] = 0;
    }
}

switch ($operation) {
    case 4:
        $numberresponse['difficultymessage']="Really ?... Do you really think it exists here?";
        break;
    case 5:
        $numberresponse['difficultymessage']="Why would you waste your time on that difficulty?";
        break;
    case 6:
        $numberresponse['difficultymessage']="So you have chosen a difficulty, you should not be reading that, you do not have the time, you should pump up that number instead";
        $_SESSION['difficulty_chosen']=true;
        break;
    case 1:
        $_SESSION['count']--;
        break;
    case 2:
        $_SESSION['count']++;
        break;
    case 3:
        $numberresponse['validatenumber']="You got up to \'\'. $_SESSION['count'].\'' I'm sure you can do better.';
        $file=file_get_contents('validated_numbers.json');
        $jsondata = json_decode(file_get_contents('./validated_numbers.json'), true);
        $newkey = count($jsondata)+1;
        $jsondata[$newkey]=$_SESSION['count'];
        $jsondata=json_encode($jsondata);
        file_put_contents('validated_numbers.json', $jsondata);
        fclose($jsonfile);
        break;
    default:
        $numberresponse['error']=" Invalid operation. Please try again.";
        break;
}
if($operation!=4 and $operation!=5 and $operation!=6){
    $numberresponse['number']= $_SESSION['count'];
}
}

// Send the new count back to JavaScript
header('Content-Type: application/json');
echo json_encode($numberresponse);
}

```

C'est avec cet exercice que nous avons été confrontés au défi du contournement. Cette première BadUI nous a demandé un certain temps, car nous devions comprendre le mécanisme permettant de stocker toutes les informations côté serveur pour empêcher l'utilisateur de les modifier. Établir cette connexion entre le code JavaScript et le PHP a été complexe, et c'est ce processus qu'on peut voir dans le code ci-dessus. Le fait que tout le code contenant les valeurs (qui sont stockées dans une fichier json auxiliaire) soit côté serveur permet de le rendre inaccessible aux yeux du l'utilisateur, qui est alors obligé d'utiliser l'UI par la fonctionnalité que nous avons prévu.

6.0.3 Phonecall



Le concept de phonecall est de permettre à l'utilisateur d'enregistrer son numéro de téléphone. Cependant, son utilisation rend la tâche quelque peu complexe. L'utilisateur doit saisir son numéro en binaire, utilisant quatre interrupteurs pour contrôler les quatre bits de chaque chiffre. Ainsi, un total de 36 interrupteurs est nécessaire. Il était crucial de garantir que chaque interrupteur affectait correctement le bon chiffre.

Ensuite, nous avons décidé d'ajouter une couche de difficulté supplémentaire. Ainsi, si l'utilisateur dépasse 9 sur l'un des chiffres, tous les interrupteurs se réinitialisent, ramenant ainsi le numéro à 000 000 000.

```
// Function to clear the phone number field
function clearField() {
    var phoneNumberField = document.getElementById('phoneInput');
    phoneNumberField.innerHTML = '000 000 000';
}

function clearAll() {
    var test = document.getElementById('phoneForm');
    var buttons = test.querySelectorAll('.digit-input');
    buttons.forEach(function(button) {
        console.log("One down");
        button.classList.remove('on');
        button.classList.add('off');
        button.innerHTML = 'Off';
    })
    const formData = new FormData();
    formData.append('clear', true);
    fetch('php/phonecall.php', {
        method: 'POST',
        body: formData
    })
    .then(response => response.text())
    .then(data => (console.log(data)))
    .catch(error => console.error('Error:', error));
    clearField();
}

function saveNumber() {
    const formData = new FormData();
    formData.append('save', true);
    fetch('php/phonecall.php', {
        method: 'POST',
        body: formData
    })
    .then(response => response.text())
    .then(data => (console.log(data)))
    .catch(error => console.error('Error:', error));
}
```

Figure 6.1: Implémentation d'un chiffre

Figure 6.2: fonction "clear" et envoi au JS

```
function toggleButton(button,digit) {
    var phoneNumberField = document.getElementById('phoneInput');
    if (button.classList.contains("on")) {
        button.classList.remove("on");
        button.classList.add("off");
        button.innerHTML = 'Off';
    } else {
        button.classList.remove("off");
        button.classList.add("on");
        button.innerHTML = 'On';
    }
    var newdigit = 0;
    var position_digit=0;
    var casetogo=0;
    if (digit>0 && digit<5) {
        casetogo=1;
    }else if (digit>4 && digit<9) {
        casetogo=2;
    }else if (digit>8 && digit<13) {
        casetogo=3;
    }else if (digit>12 && digit<17) {
        casetogo=4;
    }else if (digit>16 && digit<21) {
        casetogo=5;
    } else if (digit>20 && digit<25) {
        casetogo=6;
    } else if (digit>24 && digit<29) {
        casetogo=7;
    } else if (digit>28 && digit<33) {
        casetogo=8;
    } else if (digit>32 && digit<37) {
        casetogo=9;
    }
```

Figure 6.3: séparation des cas (chiffres)

En ce qui concerne le code, nous avons appliqué le même mécanisme de communication entre JavaScript et PHP que celui utilisé dans l'interface utilisateur précédente. Cette approche vise à empêcher l'utilisateur de modifier directement le nombre sans recourir au fonctionnement spécifique de l'interface, qui est, il faut le dire, assez contraignant. Par conséquent, l'utilisateur n'a aucun moyen de contourner la sécurité de cette interface utilisateur. Comme illustré dans les images ci-jointes, bien que les premiers traitements soient effectués dans le code JavaScript, c'est le PHP qui a le dernier mot après sa vérification des inputs avant l'enregistrement, ce qui rend le contournement en changeant directement les valeurs côté client impossible.

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $Didwegetasessionstart=session_start(['cookie_lifetime' => 3000, 'name'=>'AreYouStillTrying?',]);}
if(isset($_POST['digit'])){
    $digit = $_POST['digit'];
    if (!isset($_SESSION['digitstate'])) {
        $_SESSION['digitstate'] =[ '0000','0000','0000','0000','0000','0000','0000','0000'];
    }
    if($_SESSION['digitstate'][ceil($digit/4)-1][$digit%4-1]==0){
        $_SESSION['digitstate'][ceil($digit/4)-1][$digit%4-1]=1;
    }
    else{
        $_SESSION['digitstate'][ceil($digit/4)-1][$digit%4-1]=0;
    }
    foreach ($_SESSION['digitstate'] as $digit) {
        $test=($digit[0]*1+$digit[1]*2+$digit[2]*4+$digit[3]*8);
        if($test>8){
            $_SESSION['digitstate'] =[ '0000','0000','0000','0000','0000','0000','0000','0000'];
        }
    }
    $response['digitstate'] = $_SESSION['digitstate'];
}

```

Figure 6.4: Code PHP pour le calcul des digits

6.0.4 Labymage

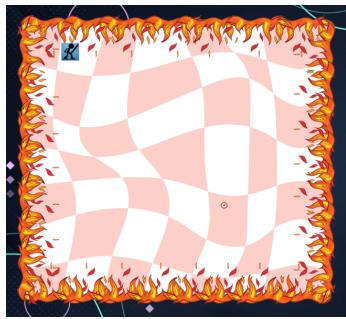


Figure 6.5: Niveau 1



Figure 6.6: Niveau 2

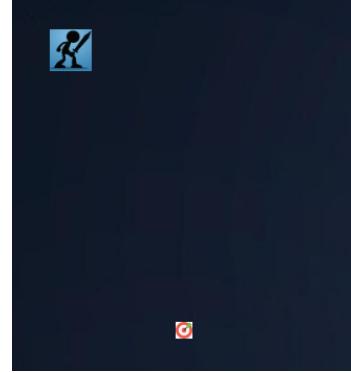


Figure 6.7: Niveau 3

Ce BadUI nous a demandé davantage de temps que les précédents en raison de sa complexité accrue pour le sécuriser et le mettre en place. Dans ce BadUI, un labyrinthe est présenté au niveau 1 sous forme d'un simple carré afin de permettre à l'utilisateur de comprendre le principe. Cependant, le niveau 2 est bien plus difficile, tandis que le niveau 3, bien que légèrement plus simple, présente le défi supplémentaire d'avoir des murs invisibles, rendant ainsi l'exercice plus ardu.

L'interface met en place une image que l'utilisateur doit glisser jusqu'à atteindre un objectif. De plus, aucune information concernant la position des murs n'est disponible côté client, et l'image qui sert à afficher les murs n'est qu'une simple image que nous avons créée. Elle ne contient aucune information, si ce n'est sa position qui est fixée à (0;0). Dès que le curseur, et donc l'image, approche trop près d'un mur, sort du labyrinthe ou se déplace trop rapidement (par exemple, lors d'une tentative de téléportation), le niveau est réinitialisé et l'utilisateur doit recommencer depuis le début.

Une autre difficulté réside dans le fait que le cookie utilisé pour attacher l'image au curseur n'est valide que pendant 30 secondes, ce qui signifie qu'il est impératif de ne pas perdre de temps.

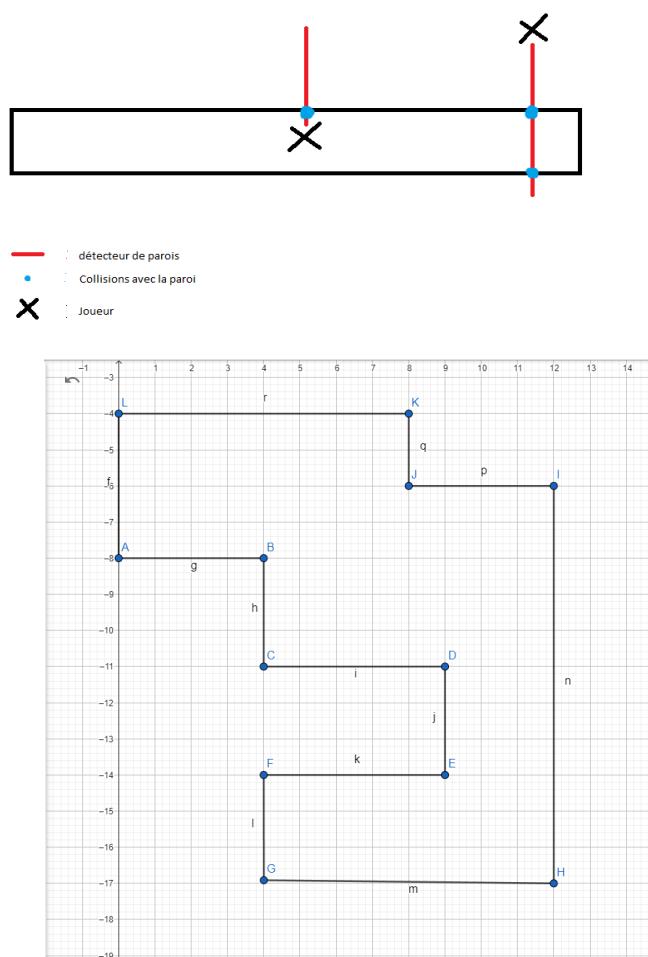
Au Niveau du code, la première étape a été de coller l'image au curseur.

```
image.addEventListener('mousemove', moveImage);

function moveImage() {
  if(on==0){
    document.addEventListener('mousemove', moveImageWithCursor, {passive: true});
    on=1;
    timeoutID = setTimeout(() => {
      document.removeEventListener('mousemove', moveImageWithCursor, {passive: true});
    }, 30000);
  }
}
```

Ensuite, nous avons dû mettre en œuvre les labyrinthes, qui étaient en réalité des polygones, ainsi que les fonctionnalités pour empêcher l'image de sortir du labyrinthe et pour limiter les mouvements trop rapides afin de prévenir toute tentative de téléportation.

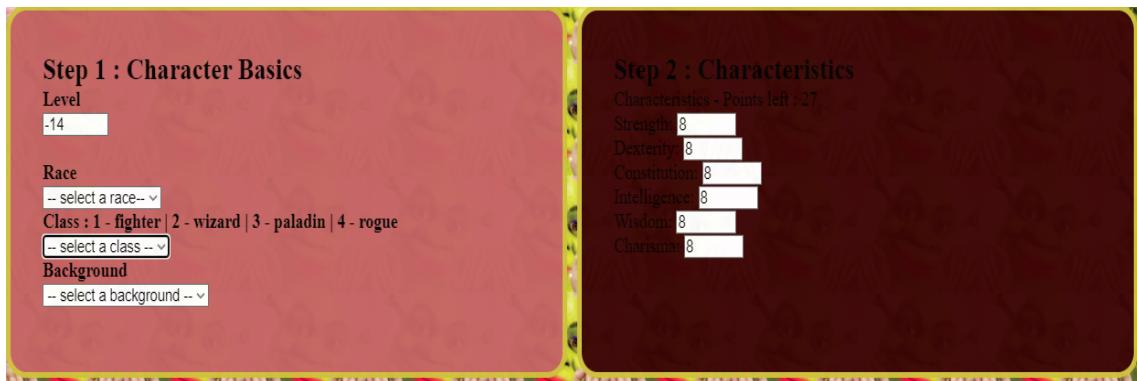
Ces fonctionnalités ont dû être gérées côté serveur pour éviter toute tentative de triche. De la même manière ce code détecte si l'image arrive bien sur sa cible. Pour détecter la sortie du labyrinthe, nous avons mis en place un mécanisme de détection qui compte le nombre de parois que devait traverser le personnage avant de sortir du polygone. Si ce nombre est pair, cela indique que le personnage à déplacer est à l'extérieur de la zone, comme illustré dans le schéma ci-dessous. Le code correspondant à ces fonctionnalités est trop long pour être inclus ici, mais il se trouve dans le fichier update-image-position.php dans notre projet.



Voici le schéma du labyrinthe 3.

6.0.5 BadForm

Ce dernier BadUI porte sur la création d'un formulaire très très mal fait pour plusieurs raisons. Nous n'avons pas eu le temps d'exploiter tout ce qu'il est possible de faire pour ce BadUI ni de le sécuriser côté serveur. Il ne s'agit que d'un début où nous avons commencé par rendre le formulaire peu intuitif et le rendant difficile à lire.



Problèmes & Résolutions

7.0.1 Changement de Logiciel

Nous avons initialement travaillé sur Replit, mais avons rencontré des limitations, telles que l'utilisation de plusieurs fichiers dans des langages différents, ainsi que l'implémentation d'un git, qui demandaient l'adhésion payante à la version premium du site, nous poussant ainsi à passer sur Visual Studio Code avec XAMPP pour le développement local. Ce changement nous a permis d'avoir un meilleur contrôle sur notre environnement de développement et une meilleure gestion des fichiers et des services nécessaires à notre projet.

7.0.2 Bypass

Nous avons identifié le risque de contournement facile ainsi que les mesures de sécurité à mettre en place côté client en étudiant des Bad UIs créées par d'autres développeurs.

Pour contrer cela, nous avons mis en place des vérifications systématiques des informations envoyées par l'utilisateur côté serveur, ce qui garantit l'intégrité des données et prévient les tentatives de contournement. Théoriquement facile à comprendre, la mise en place de solution pour parer le souci nous a pris beaucoup de temps, notamment la communication entre le code JavaScript et le code PHP.

7.0.3 Maladie

Pendant la durée du projet, nous avons tous les deux été malades pendant une durée relativement longue (1 mois+), ce qui a considérablement fait baisser notre productivité et notre temps de concentration. Cela nous a donc forcé à travailler à distance et à revoir nos horaires et méthodes de travail pour continuer à être efficaces en préservant au maximum notre santé mentale et physique, mise à rude épreuve.

7.0.4 Expérience en PHP/JavaScript

Initialement, nous avions peu d'expérience avec JavaScript et PHP, les langages de programmation principaux pour notre projet. Pour surmonter cette lacune, nous avons adopté une approche d'apprentissage autodidacte, en explorant les ressources disponibles en ligne et en testant nos connaissances au fur et à mesure de notre progression. Ce processus nous a permis d'acquérir les compétences nécessaires pour réaliser notre projet. Cependant, cette quête de connaissances nous a elle aussi, pris beaucoup de temps.

7.0.5 Méthodes de travail

Nos approches de travail et nos horaires efficaces étaient très différents, ce qui a parfois rendu difficile la coordination de nos efforts. En effet, nous avons deux personnalités plutôt opposées, et nos modes de vie sont très différents, ainsi nous coordonner pour travailler ensemble efficacement n'a pas été simple. De plus, il est important de préciser qu'en parallèle, nous avions beaucoup de projets que d'autres matières nous imposait, ce qui a réduit le temps disponible pour travailler sur ce projet, notamment lorsqu'on apptochait de sa fin.

7.0.6 Divergence des compétences

En outre, nos niveaux de compétences dans chaque domaine requis pour le projet étaient également très variés. Nous ne disposons pas exactement des mêmes compétences en apprentissage et en compréhension des langages utilisés. Nous avons donc pris du temps pour faire en sorte que nous comprenions tous deux chacuns des procédés du projet.

Nos autres idées de BadUI

Dans cette section, nous présentons d'autres idées de BadUIs auxquelles nous avons pensé :

- Remplacement de l'alphabet : Une idée consiste à remplacer l'alphabet standard par un autre lors de la saisie au clavier ou même dans l'affichage du site. Par exemple, nous pourrions utiliser l'alphabet elfique, comme illustré ci-dessous :

ELVISH SCRIPT: SAMPLE ALPHABET														
A	B	C	D	E	F	G	H	I	J	K	L	M		
ø	ø	ð	ð	ø	ç	ø	ȝ	ɔ	ø	ø	ø	h	v	
N	O	P	Q	R	S	T	U	V	W	X	Y	Z		
ñ	ð	ȝ	l	ð	ø	ð	ð	ø	ø	ñ	ȝ	ɔ	ø	

- Système de lettre aléatoire : Les utilisateurs pourraient être confrontés à une seule lettre et auraient le choix entre l'ajouter ou rafraîchir pour en obtenir une autre.
- Navigation aléatoire sur le site : Nous pourrions créer un système de navigation aléatoire où le serveur renvoie un lien aléatoire à suivre à chaque fois que l'utilisateur souhaite accéder à une nouvelle page.
- Contrôle audio déroutant : Une idée serait d'avoir deux vidéos, et lorsque l'utilisateur lance l'une d'elles, l'audio de l'autre se déclenche simultanément, créant ainsi une expérience désorientante.
- Calculatrice à layout changeant : Nous pourrions concevoir une calculatrice dont la disposition des touches change aléatoirement à chaque pression, ou encore un clavier virtuel avec le même principe.

- Réglage du volume : Nous avions envisagé de concevoir une barre permettant d'ajuster le volume à l'aide d'un curseur. Cependant, nous avons eu l'idée d'utiliser l'icône du haut-parleur située à droite de la barre du curseur comme déclencheur pour le curseur lui-même. En restant appuyé sur l'icône du haut-parleur pendant une certaine période, le curseur serait lancé plus loin sur la barre, augmentant ainsi le volume audio, à l'image d'un lanceur automatique de balles de tennis.



Conclusion du Projet

9.1 Fin du Projet

Ce projet a été très enrichissant : Il nous a permis d'en apprendre plus sur le développement web et plus précisément sur la conception d'interfaces utilisateurs. Nous avons aussi dû nous adapter au php et au JavaScript afin d'en comprendre les spécificités. Nous aurions bien aimé pouvoir en faire plus, mais étant limités en temps et ayant 5 autres projets sur lesquels nous devions travailler en parallèle, il a été compliqué pour nous de nous investir encore plus.

Nous sommes tout de même très contents du résultat final, même si nous aurions aimé pouvoir passer plus de temps dessus. En y réfléchissant, nous aurions sûrement dû nous mettre à coder et à tester des idées plus tôt au lieu de réfléchir à plus d'idées et de se renseigner sur les langages de programmation.

9.2 Ce que nous avons appris

Comme énoncé plus tôt, nous avons renforcé nos connaissances en php et JavaScript, ainsi que sur la conception d'interfaces utilisateurs web. Nous avons aussi commencé à apprendre comment contourner des interfaces web (surtout côté client) afin de les manipuler et d'obtenir le résultat que l'on souhaite.

9.3 Commentaires personnels

9.3.1 Léo

J'avais choisi ce projet afin de pouvoir en apprendre plus sur le développement web et la conception d'interfaces utilisateurs, et notamment pour savoir si c'était un domaine où j'aurais pu souhaiter aller après mes études.

Il s'est avéré que bien que le projet ait été très intéressant, ce n'est finalement pas ce genre de projets que j'aimerai gérer plus tard ou même développer à pleins temps. Je suis donc très content d'avoir pu m'en rendre compte dès maintenant et je continuerai à explorer de nouvelles possibilités et de nouveaux domaines afin de trouver ce qui me convient le mieux.

Je tiens aussi à remercier Katia pour avoir fait le projet avec moi. Je continuerai néanmoins à m'intéresser au côté "bypass" de ces interfaces notamment avec le site [Newbie Contest](#) que M.Lagrange nous a présenté au cours du projet.

9.3.2 Katia

J'ai opté pour ce projet en raison de son originalité. Bien que je doive avouer que la nature exacte du projet m'échappait à la lecture de son résumé. L'idée de créer des interfaces utilisateur complexes à utiliser avec un esthétique discutable m'attirait énormément.

De plus, dans le cadre de mon développement professionnel futur, j'aspire à me spécialiser dans le design web, et ce projet semblait être le plus en phase avec mes aspirations, même si j'ai rapidement réalisé que le style n'était pas l'élément central de ce projet.

J'ai beaucoup appris en collaborant avec Léo, que je tiens à remercier chaleureusement pour avoir partagé ce projet avec moi. Léo possède des compétences en programmation que je n'oserais jamais revendiquer, et il a toujours été disposé à expliquer de manière claire le fonctionnement du code, même s'il perdait du temps sur son propre travail.

Maintenant, je ne pense pas poursuivre ce type de projet à l'avenir, car le développement back-end ne correspond pas vraiment à mes aptitudes. Je suis convaincue d'avoir confirmé mon choix de me concentrer davantage sur le design plutôt que sur les fonctionnalités proprement dites. Cependant, je ne regrette en aucun cas d'avoir choisi ce projet, qui m'a permis de progresser considérablement et de découvrir des langages que je ne maîtrisais pas du tout, tels que JavaScript et PHP.

Remerciements & bibliographie

10.1 Remerciements

Nous tenons à remercier M.Lagrange pour avoir proposé ce projet et nous avoir suivi durant toute la durée de celui-ci. Nous remercions aussi tous les développeurs que nous avons cité dans notre rapport en présentant leurs BadUIs, ainsi que nos amis pour nous avoir apporté un regard neuf et un avis constructif lors des tests de notre projet.

10.2 Bibliographie / Sitographie

- [user-789](#)
- [userinyerface website](#)
- [goulartnogueira](#)
- [i01000101](#)
- [im ocean](#)
- [Reddit Community BadUI Battle](#)