Langage colibri

Qu'est-ce que c'est?

Colibri est un langage informatique harmonieux qui améliore les langages existants. Colibri est proche d'un CMS (content management system), d'un website builder et d'un logiciel de présentation (diaporamas). Il est voué à être très abordable parce qu'il est son propre framework. Il est facile d'utilisation et accessible. Mais ce langage n'a pas de version gratuite à cause du serveur embarqué.
Par ailleurs, il sera d'abord réfléchi et sera le mieux adapté au site web La Licorne Filante.

Software

Des applications seront conçues pour simplifier le code :
- un éditeur pour le front-end et le back-end qui inclut la prise en charge de la base de données
- un simulateur pour visualiser la forme du site sans forcément utiliser un navigateur

Colibri a la particularité d'intégrer plusieurs sites dynamiques sur une seule page.
Ainsi, à la différence des iFrame et plutôt que d'ouvrir plusieurs onglets, les sources seront encastrées avec fluidité dans la page d'appropriation pour améliorer la transparence et optimiser le stockage de données (exemples: un générique, une citation).
C'est aussi sur quoi La Licorne Filante fonctionnera. En effet, la structure du site est majoritairement composée d'un corps entouré d'espaces latéraux à droite et à gauche, d'une barre des tâches en haut et d'une barre de notifications en bas. Chaque composant du site sera par conséquent, selon le programme, soit indépendant l'un de l'autre, soit complémentaire.

Hardware

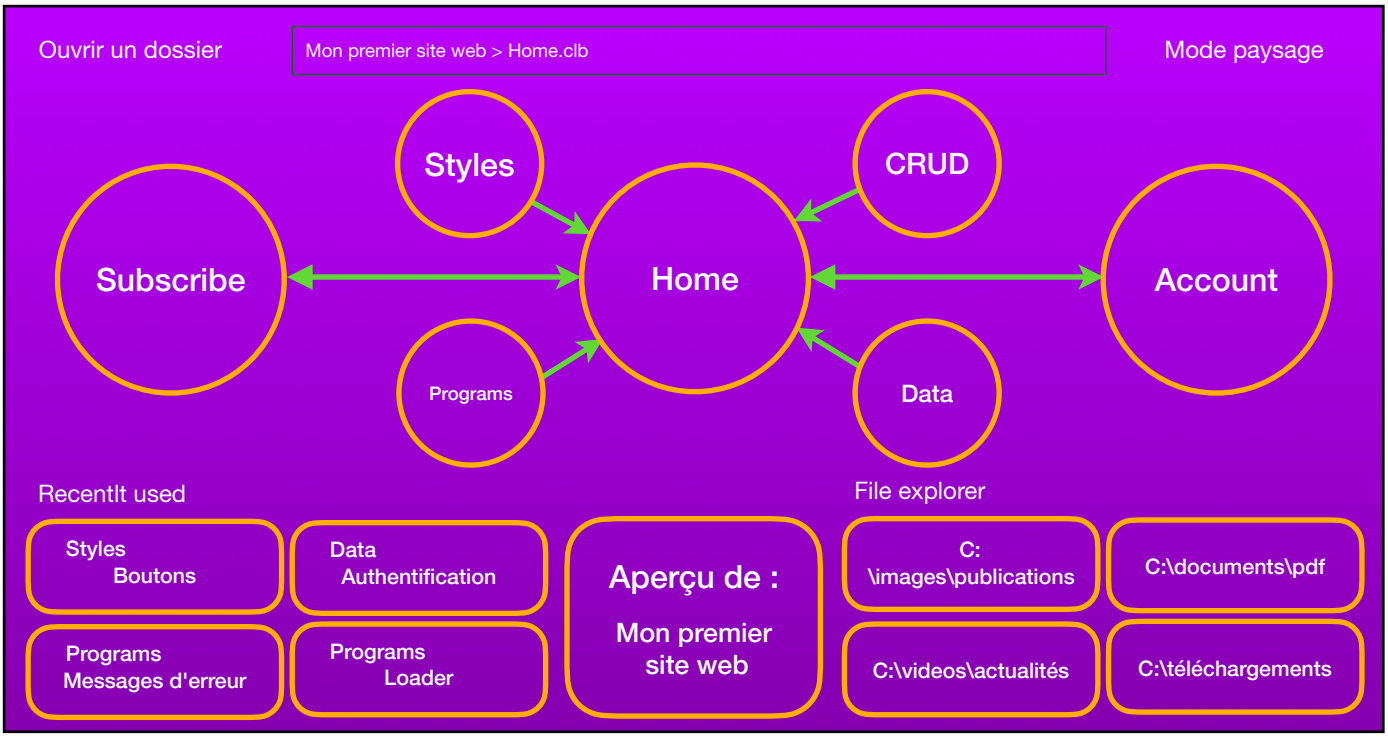
Colibri a pour but de faire évoluer les langages déjà existants. Ainsi le grand public pourra gagner en indépendance car il décentralisera internet. En effet, la nouveauté est l'émergence du hardware côté développeur. Un boitier oeuvrera harmonieusement avec les applications citées précédemment afin d'accéder vers un serveur vraiment local.

Le boîtier Colibri est fabriqué pour dépendre de son socle (bluetooth, induction) ou d'un mobile (partage de connexion, batterie externe). Et pour cause, dans ses fonctions primaires, il ne sera équipé que d'un espace de stockage. Les avantages seront toutefois multiples. Dans sa fonction secondaire, il agira comme une alarme. En cas de vol, perte ou accident, le boîtier est paré d'équipements : une pile bouton, un émetteur GPS et une alarme. Trop éloigné du mobile appareillé, il sonnera automatiquement.

Voici les autres atouts :
- Premièrement, on peut travailler en mode hors-ligne dès que les données seront récupérées;
- Deuxièmement, des flux de données accessibles au public seront échangées en insérant des puces en forme de carte micro SIM (vendues séparément). Cela concernera : l'heure, la météo, les finances, la position géographique, le traffic routier,... on pourrait en théorie s'en servir comme une carte bancaire, un titre de transport ou une carte membre;
- Troisièmement, la sécurité est très fiable car il faut d'abord infiltrer le socle ou un mobile pour accéder aux données du boîtier, si bien sûr celui-ci est en contact avec l'un d'eux;
- Quatrièmement, l'idée est de d'activer le cloud que lorsque nécessaire et que le boîtier est fonctionnel pour des raisons écologiques;
- Cinquièmement, la protection des données personnelles et de la vie privée sera meilleure et l'utilisateur sera moins envahi par les publicités ciblées;
- Sixièmement, le boîtier Colibri permet de créer son propre site intranet depuis un PC ou un mobile, ce qui diminue les frais d'abonnements (mais aussi, malheureusement, potentiellement les coûts de main d'oeuvre).

Présentation de l'éditeur

Lorsque nous sélectionnons un projet, voici comment se présente le **menu principal** de l'éditeur.



Commençons d'abord par la barre multifonctionnelle, située en haut de la fenêtre.

En survolant cette barre, 4 options apparaissent. Lorsque que l'on clique sur l'une d'elle, une fenêtre glisse depuis le haut jusqu'au milieu et prend la moitié centrée de la largeur de l'écran.
Arborescence : permet de voir et naviguer sur les chemins qui compose le projet
Rechercher et remplacer : soit par objet (les 2 espaces à côté de la fenêtre sont réservés au split, par glisser-déposer), soit par mot; ici, nous sommes dans le menu principal, donc il s'agit, dans cet exemple, d'une recherche sur tout le dossier Home.
Modèles : codes du projet et sur internet à copier, plutôt que d'aller sur le navigateur.
Paramètres : accès aux périphériques qui mènent au bon fonctionnement du site web.

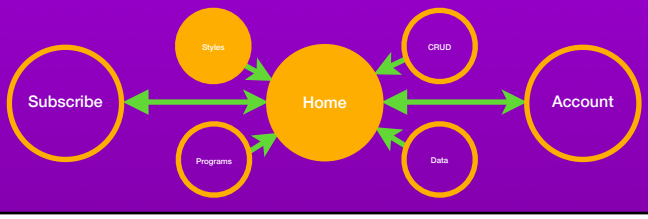
Il y a deux modes de présentation : paysage et portrait.

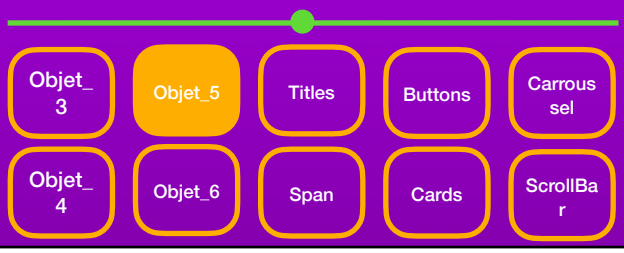
Ouvrir un dossier

Mon premier site web > Home > Styles > Styles_index.clb > Objet_5

Mode paysage

#Le code est est semblable au CSS mais est sensible à l'indentation#
Objet_5 : {
position {x,y,z,p;} #La position détermine le calibre d'un bloc#
border {size, color, radius;}
background {repeat, color, URL;}
font {size, color, family;}
flex-wrap {width;}
icon {<a>, }
shadow {} ;
}





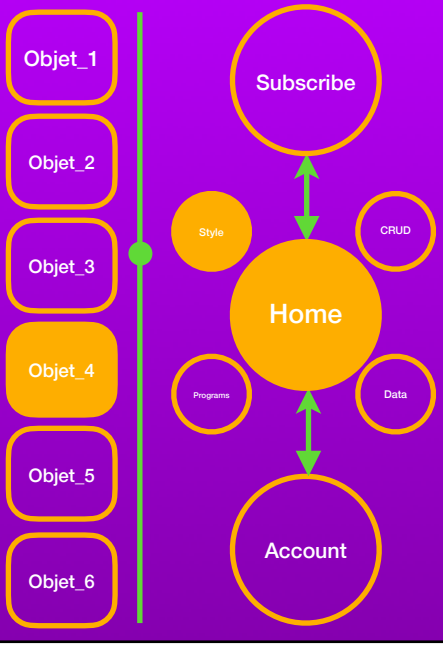
Ouvrir un dossier

Mon premier site web > Home > Styles > Styles_index.clb > Objet_4

Mode portrait

#Le code est est semblable au CSS mais est sensible à l'indentation#
Objet_4 : {
position {x,y,z,p;} #La position détermine le calibre d'un bloc#
border {size, color, radius;}
background {repeat, color, URL;}
font {size, color, family;}
flex-wrap {width;}
icon {<a>, }
shadow {} ;
}

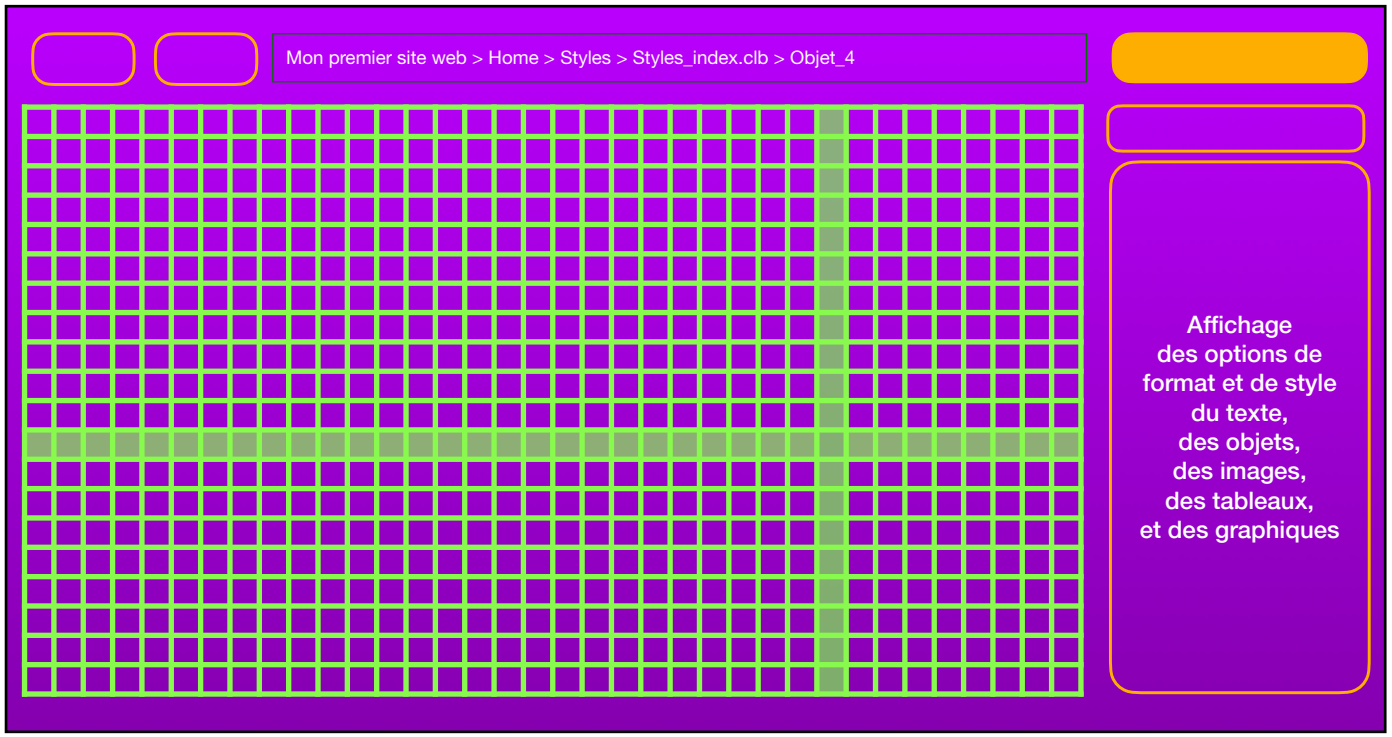
Objet_5 : {
position {x,y,z,p;}
border {size, color, radius;}
background {repeat, color, URL;}
font {size, color, family;}
flex-wrap {width;}
icon {<a>, }



Nous avons visuellement 3 éléments.
Un schéma avec : des grands cercle pour les dossiers et des petits cercles pour les fichiers ; un sommaire avec le nom des objets ; une barre de défilement.

L'endroit où nous sommes est indiquée dans la barre d'adresse et est en surbrillance.
En cliquant sur un des éléments en surbrillance, un autre affichage apparaît. Il est spécifique aux dossiers, à chacun des 4 fichiers et aux objets.

Second affichage des fichiers Styles

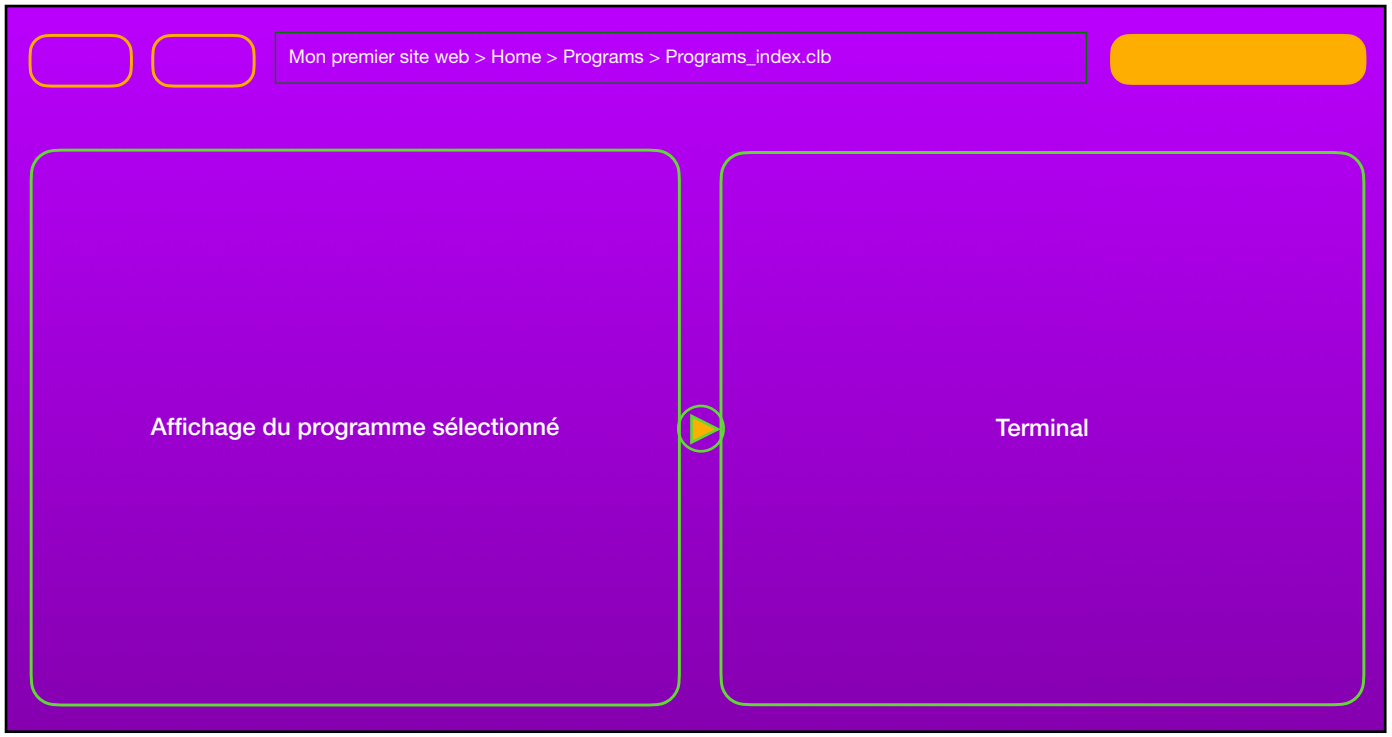


Versions : sélectionner PC (36:20) ou mobile (20:36) ; Axes (ici: abscisses = 57 et ordonnées = 28); Barre d'adresse (partie multifonctionnelle désactivé)e ; Retourner au code : appuyer sur Styles ; Options de format et de style : liste déroulante et affichage ; Quadrillage : wireframe.

Second affichage des fichiers Data

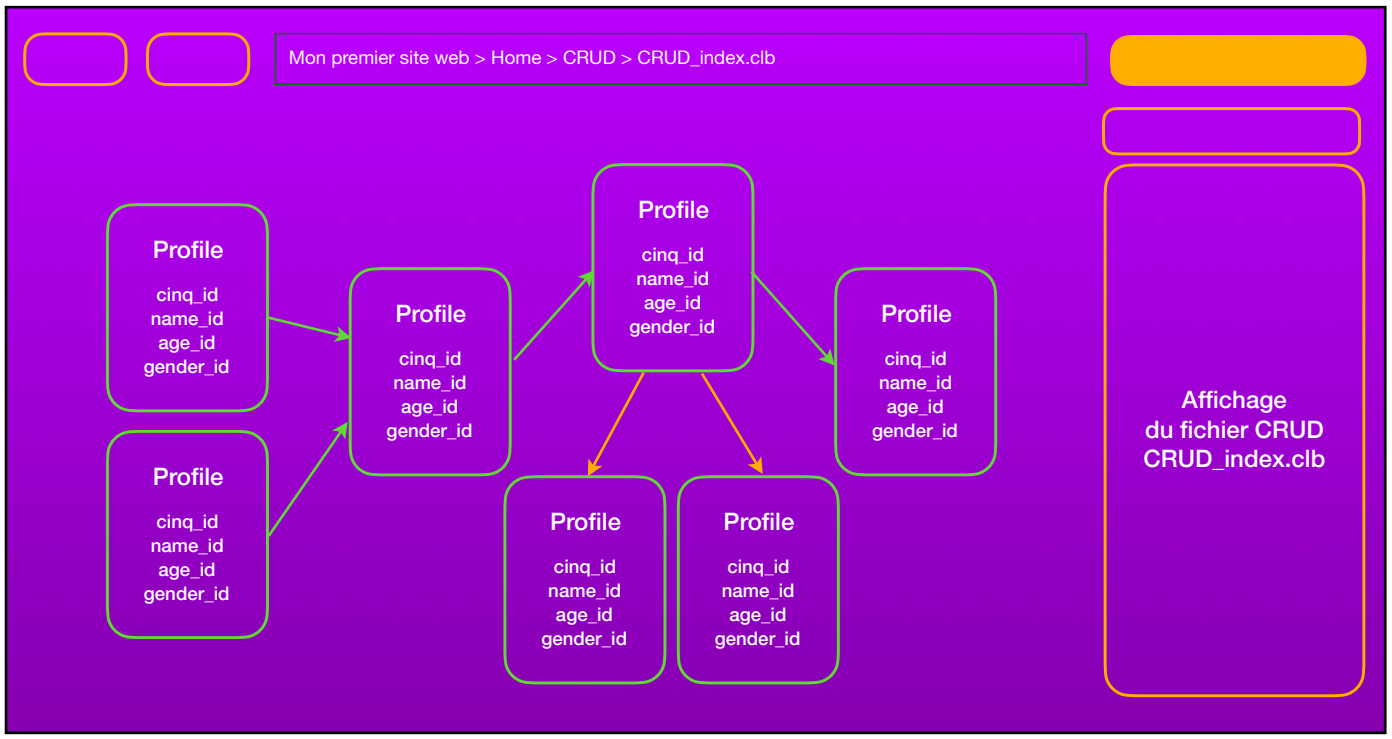
Etat : ON ou OFF (est-ce que les visiteurs ont accès à la base de données?).
Nombre de pages qui utilisent cette base de données : dans l'exemple, il y en a 3.
A l'instar des styles, il y a besoin d'un terminal pour gérer les bases de données.

Second affichage des fichiers Programs



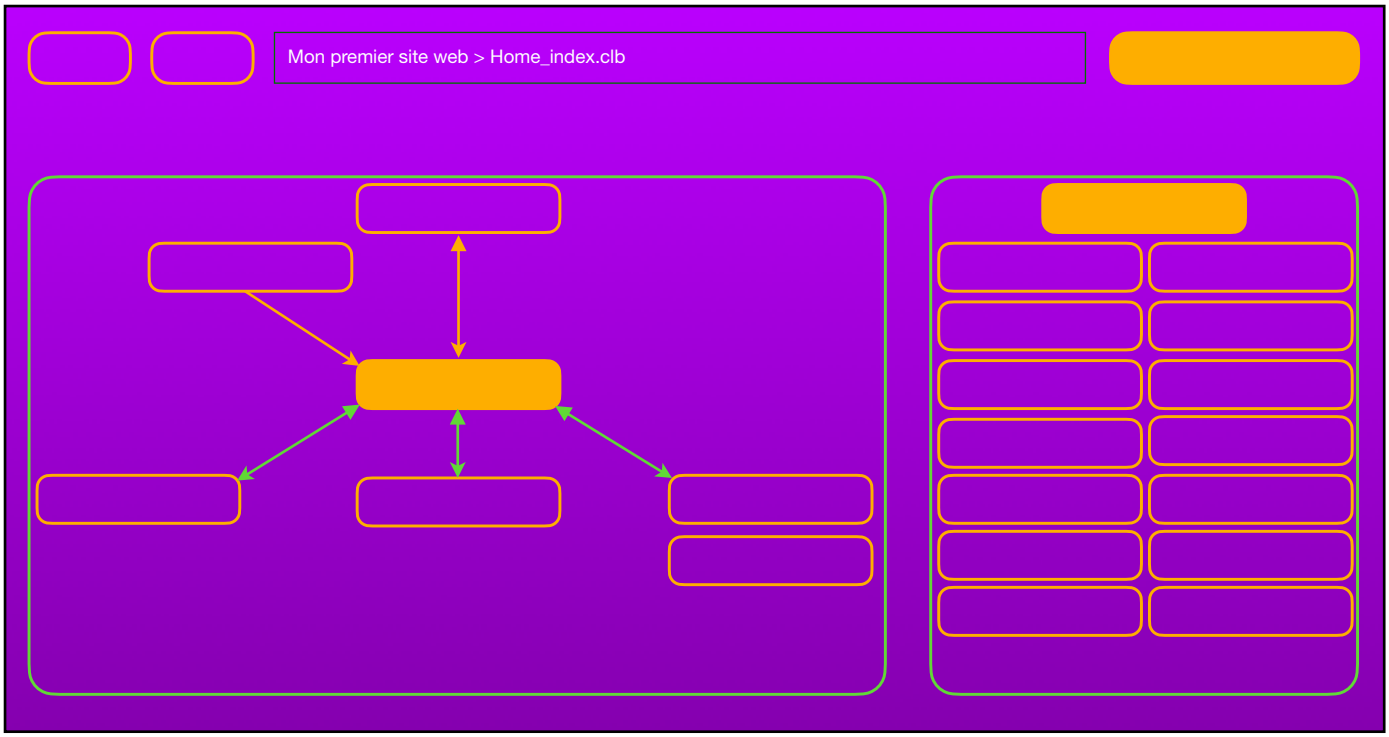
Dans le même état d'esprit, le second affichage propose les fonctions essentielles au fichier.
Le mode split offre une vue sur le terminal et le programme en cours d'excécution.
Nature du dernier programme testé : ici, c'est un float ; Pages qui utilisent ce programme : ici, 2.

Second affichage des fichiers CRUD



Ceci est un plan du site sous forme de modèle de conception de données (MCD).
Etat : ON (connecté à une base de données). Bouton "+" : ajouter des entités et propriétés
Flèches oranges : entités accessibles uniquement par l'admin.

Second affichage des dossiers



Dans cette interface, on peut gérer deux choses.

La maquette du site
Aussi appelée organigramme, elle va mettre en évidence l'endroit où nous sommes de 3 façons différentes automatiquement ou lorsque l'on clique sur la page concernée:
- un large espace tout autour;
- le style est en surbrillance (il y a donc 2 façons de revenir à l'affichage principal);
- des flèches (vertes pour tout le monde et oranges pour l'admin uniquement) à sens unique et à double sens indiquent les direction vers les autres pages.

On peut naviguer en faisant défiler l'organigramme dans toutes les directions et en paramétrant le zoom (en pourcentage ou en plein écran).
On peut aussi ajouter une page ici. Les 5 dossiers (Styles, Programs, CRUD, Data et Files) seront automatiquement générés. La nouvelle page sera pré-remplie selon où on la placera dans l'organigramme. Par exemple, si on crée une page "Mon panier" et qu'on la relie avec "Home" par une flèche, il y aura dans "Mon panier" : Home

Le dossier Files
Il ne sera pas affiché dans des bulles car il s'agit de duplicats de la page en question en HTML. Son emplacement rappelle que seul ce dossier ne peut pas être générer automatiquement.
Le principe est le même que pour la maquette du sit: si on clique dans l'encadré droit (mais sur aucune page) puis sur "+", un nouveau fichier sera crée. Dans cet exemple, il s'agit de pages de traduction. Il suffira de supprimer le texte en français et de le remplacer dans la langue que l'on souhaite. Si le code est correctement tapé, une liste défilante "Changer da langue" a déjà été crée dans les fichiers Styles et Programs. En dessous de la barre d'adresse, il y a un espace vide: **C'est ici que Colibri demandera l'autorisation** d'ajouter à cette liste ce qui suit après "Files_".

Introduction aux fonctionnalités
Colibri tire sa puissance dans la taxonomie et dans la structure de l'arborescence. Là où plusieurs langages ont besoin constamment d'un terminal pour taper des commandes, ici, des morceaux de codes seront ajoutés de façons semi-automatiques. Cela évite de débugger inlassablement et on peut plus facilement créer un site plus complexe. Ce sont les **fonctionnalités répertoriées**.
D'une autre part, il y a des plugs-in qui peuvent s'ajouter dans les listes déroulantes de Styles, Data et CRUD pour décupler les possibilités des plus créatifs. Des abonnements modulables à la semaine est souhaitée de la part des collaborateurs. Ce sont les **fonctionnalités additionnelles**.

Fonctionnalités

Fonctionnalités répertoriées

Comme pour tout langage, il existe une documentation. Mais elle est imposante au point que de nombreuses fonctionnalités sont très peu utilisées. Colibri opte pour la rentabilité et l'efficacité. Les codes générés sont souvent accompagnés d'un commentaire pour guider le développeur.

Globalement, la majorité des **pages** Colibri se code de la manière suivante:

```
<div> #Les commentaires sont entre 2 caractères dièse.#
Objet = [fichier_1 ; fichier_2;] : {
        <p>Texte</p> propriété_1 , {propriété_2} ; </p> #Les propriétés avant le point virgule s'appliquent ici à la balise#
        <img src="" alt=""> propriété_1 , propriété_2 ;
}
</div>  #On peut encore simplifier en écrivant dans <head> : groupe = [fichier_1 ; fichier_2].#
```

A la création d'une page, des objets s'affichent d'emblée dans chacun des 4 fichiers pour bien démarrer. Dans le **fichier Styles**, on peut coder en CSS et en Colibri.

```
#Clic-droit puis "Lock as header" pour encadrer et faire une en-tête fixe#
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">
IMPORT = [Data_index.clb] : {(GraphicArt,)} #On importe depuis Data_index.clb une application pour l'utiliser dans le second affichage#
```

```
objet_! : {
        position {x,y,z,p;} #La position détermine le calibre d'un bloc#
        border {size, color, radius;}
        background {repeat, color, URL;}
        font {size, color, family;}
        flex-wrap {width;}
        icon {<a>, }
        shadow { } ;
}
```

Ceci est une mauvaise pratique.
Dans colibri, le nom des contenus, formes et composants sont directement écrits avec la première lettre en majuscule suivi d'un underscore et l'appelation.
Cette appelation servira dans les autres fichiers. Pas besoin, donc d'écrire id="", class="" ou name="".
> Objet_Nom

```
Objet = [GraphicArt] : {<img src="MonPremierSiteWeb/hummingbird.png" alt="">;} #Insertion de la réalisation ici#
```

```
#animation# #2 façons d'écrire#
Objet_! =  [<img src="" alt="">] : {
        FROM start TO end DURING time ,
        THEN x:-( ) y:-( ) TO  x:-( ) y: ( ) DURING ---- z ( ) ,
        NOW start TO end DURING time ;
}
```

```
Objet = [FROM TO DURING , THEN TO DURING
        (for i=0 ; i<10 ; i++) , NOW TO DURING ] : {
                x:int(float) y:int(float) z(int) int ,
                ... ,
                x:int(float) y:int(float) z(int) int ;
}
```

```
#rotation# #Les fonctions sont écrites en majuscule pour les différencier des propriétés et pour éventuellement alléger le code#
Objet_! : {
        FROM direction TO direction ON degree (float) DURING : position , #virgule pour passer à une autre commande#
        THEN : repeat, no reverse ;  #point virgule pour annoncer la dernière commande avant de l'acolade fermante#
}

<div> #Carroussel# #On extrait Button de Styles.clb et Programs.clb#
Objet_! = [<img src="" alt="">  , <img src="" alt="">  , <img src="" alt="">  , <img src="" alt="">  ; Button] : {
                size : {
                position x : FROM 11( ) TO 31( ) ,  #Quand il n'y a rien entre parenthèses, cela sous entend "1.0"#
                position y : FROM 33( ) TO 37(0.2) ;
                position z : front #Cette fonction assure que le caroussel sera toujours au premier plan#
                position p : relative ;
        }
        FROM left TO right
        DURING : 4850 ;
        transition : {
                swipe , #Définit la nature de la transition#
                WHEN cursor(false) ; #Le caroussel s'arrête quand le curseur passe dessus#
                WHEN Button(true) ; #Le caroussel s'anime quand on appuie sur le bouton#
        }
}
</div>
```

Data

Localhost : 4500 ;
Data : nom_de_la_base_de_données ;
information (source , format) ;
IMPORT :
	calendar ( ) ; datetime ( ) ; hours ( ) ; météo( ) ; maps (latitude, longitude) ; traffic ( ) ; location ( ) ;
	stocks(Y€$£,etc) ; unities (kg<>lbs, m<>miles, °C<>°F, L<>cm^3, etc) ; antivirus ( ) ;
URL :	website ( ) ; notification ( ); RSS ( ); API (); iFrame ( );

Programs

```
WHILE Objet : {
	 true WHEN h THEN setInterval:1500 ;
	false WHEN cursor || time > 1000000 ;
}
```

On peut traduire par : Objet s'activera lorsqu'on appuiera sur h puis se mettra à jour toutes les 1,5 seconde Object se désactivera soit lorsque le curseur balayera sur lui, soit au bout de 1000 secondes.

```
FOR answer_name = ["Quel est ton nom ?" ; string]
FOR answer_age = ["Quel âge as-tu ?" ; int] : {
	IF < 3 DISPLAY " est un bébé."
	ELSE IF>= 3 && < 13  DISPLAY " est un enfant."
	ELSE IF >= 13 && < 18  DISPLAY " est un adolescent."
	ELSE IF >= 18 && < 30  DISPLAY " est un jeune adulte."
	ELSE IF >= 30 && < 59  DISPLAY " est un adulte confirmé."
	ELSE IF >= 59 && < 80  DISPLAY " est un sénior."
	ELSE >= 81  DISPLAY " va bientôt mourir." ;
}
DISPLAY answer_name[ ]  answer_age{ } " parce qu'il a " answer_age[ ] "."
> SMgeek45 est un jeune adulte parce qu'il a 26 ans.
```

Les conditions, les booléens, les boucles et l'affichage s'écrivent en majuscule.
Exceptés les chaînes de caractères, seules les fonctions s'écrivent uniquement en miniscule.
A la différence de Python, "input" ne s'écrit pas parce qu'il est interprété par les crochets [ ].
Les majuscules épargnent bien des affectations, parenthèses et autres ponctuations.
Pour différencier la réponse à la question entre crochets et les conditions entre accollades, (on comprend mieux l'importance de ces caractères), il suffit de les écrire accolés à la fonction.

```
array = [0,1,6,2,4,7,3,8,5,9,10[oiseau,pollinisateur] ; str, int] : {ASC ; 10}
#Ceci est un dictionnaire# #point virgule# #nature des caractères# #fonction en majuscule (ici: trier les chiffres dans l'ordre)#
#Ce n'est pas un tableau parce que le chiffre 10 est définit par 2 éléments#
DISPLAY array
> 0 1 2 3 4 5 6 7 8 9 10
> oiseau pollinisateur
```

```
array === Objet [Data_index]: {"style life"} #Ce programme permet de jumeler ce dictionnaire avec une colonne du tableau du fichier Data_index, donc de manipuler la database (équivalent de Entity et RegistrationForm dans le langage Symfony)#
```

CRUD

```
#Clic-droit puis "Lock as header" pour encadrer et faire une en-tête fixe#
Localhost : nom_du_serveur_local ;
Data : nom_de_la_base_de_données ;
```

```
#Clic-droit puis "Lock as title" pour encadrer et faire une en-tête défilante#
Objet = CREATE_Objet + READ_Objet + UPDATE_Objet + DELETE_Objet
```

```
CREATE_Objet : {
	IF(empty) : {""} , string, upper, lower, int, float, null, require[#@&?!%*$€£¤§-\_/(")]
	ELSE(!empty) : GET Data , DISPLAY("") , SET Data
}
READ_Objet : { IF(!empty) : {CREATE, GET Data}  ELSE : DISPLAY("Erreur de chargement")
}
UPDATE_Objet : {
	IF(empty) : { CREATE }
	ELSE (!empty) : { !READ, SET Data, DISPLAY("C'est ici que tu peux t'exprimer.") }
}
DELETE_Objet : {
	IF(!empty) : { !READ, SET Data, DISPLAY("Nous te confirmons la suppression définitive de objet") }
	ELSE : DISPLAY("Erreur de chargement")
}
#oui, require est bien un tableau dans le code du CRUD!#
```

Comme tous langages, Colibri évoluera. Ce ne sont donc que des listes non exhaustives.
Cependant, le code doit finir par fonctionner en codant aussi simplement que précédemment.

Fonctionnalités additionnelles

Prenons par exemple Py Charm (logiciel pour Python) Il n'est pas possible d'intégrer une autre librairie et encore moins des extensions. A l'inverse, en ce qui concerne Colibri, il est souhaitable de collaborer avec des entreprises afin d'enrichir les librairies et les possibilités.

Bien évidemment, les sites payants le resteront et on pourra passer par le terminal pour les réglements bancaires, un peu comme au temps du minitel mais plus simplement et plus sécurisé que sur internet. L'idée est de tout réunir au même endroit de façon harmonieuse.

Dans les frameworks en front-back, Bootstrap domine. Notons qu'on peut aisément l'intégrer ici.
Actuellement, CodePen est un des leaders des hébergements de code. On peut citer des sites tiers tels que 3dtransforms.desandro.com .
Actuellement, Adobe est le leader des outils informatiques pour l'informatique.

Particularités de Colibri

- Diviser pour mieux régner

Pour commencer, il n'existe pas à l'heure actuelle un seul langage populaire qui a son propre hardware à l'exception de Swift, développé par Apple. Mais cela reste discutable, d'autant plus que l'acessibilité est maigre et assez réducteur. La première particularité est un serveur local. Puisque le serveur est local, nul dira que le seul accès vers le code d'autrui est GitHub. Désormais, on pourra se transférer des données via le même principe que la cryptomonnaie. Les échanges d'informations serviront à la science, l'Etat, un usage pro... d'une outre mesure.

- Une autre façon d'utiliser sa mémoire

De fait, Colibri a pour ambition de démocratiser une technologie qui désactive énergiquement la datacenter où sont stockées nos données dans le cloud. Un paramètre permet de la réactiver. Aussi, lors d'un vol de téléphone, il est moins probable de perdre toutes ses données. Cela donne de la souplesse dans la gestion de stockage en tendant vers une consommation responsable. Par ailleurs, nous aurons tendance à prendre plus de temps pour faire le tri et le nettoyage.

- Anticiper des problèmes de réseau

L'accès vers ces données, une fois téléchargées, seront disponibles en mode hors-ligne. Pourquoi pas importer notre site en ligne préféré en attendant que le réseau revienne... Et si notre magazine hebdomadaire se téléchargeait directement dans la mémoire sans avoir à faire la moindre manipulation? Que dire des étudiants à l'université qui ne peuvent faire des comptes rendus qu'au travers les moteurs de recherche.

- Développer un marché du travail équitable

Le réseau social La Licorne Filante veut arriver à un stade où les vidéastes en ligne, lesbloggeurse, les amateurs et les spécialistes arrivent à travailler d'un commun accord sans être submergé par la publicité ni être distrait et en gagnant en indépendance. En somme : réinventer internet. Plus :
là où la presse et les activistes sont mis sous silence, là où des populations n'ont pas accès à internet ou ne peuvent pas se permettre d'investir, là où les pionniers de la consommation industrielle ne sont pas écoutés ou bien sont isolés, là aussi sera Colibri.