ionic-vue

m2iformation.fr

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Ionic Vue.js

Sommaire

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Introduction

Présentation du framework

- Ionic est un **framework JavaScript** open-source créé en **2013** par Max Lynch, Ben Sperry, et Adam Bradley
- Il permet de créer des **applications** iOS et Android **natives**, ainsi que des **PWA** prêtes pour le mobile
- Il est compatible avec **Angular**, **Vue**, et **React**

Capacitor

- Capacitor.js est le **moteur d'exécution multi-plateformes** d'Ionic
- Il permet de créer des applications Web Natives en donnant **accès aux composants natifs** des différentes plateformes (Android, iOS)
- Capacitor permet de transformer une app web en application native à l'aide de son API tout en utilisant du JavaScript moderne

Les composants Ionic

- Ionic est une **bibliothèque de composants** construits avec les standards du web en utilisant HTML, CSS et JavaScript
- Les composants sont **hautement personnalisables** afin que les applications puissent s'approprier chaque composant

Web View

- Une WebView est un **navigateur** qu'une application native peut utiliser pour afficher du contenu web
- Une application native est une application écrite dans un langage et un un framework conçus spécifiquement pour une plateforme donnée :
  - Android: Kotlin, Java
  - iOS: Swift, Objective-C
- Le moteur de Ionic permet d'utiliser la WebView et communiquer avec les APIs spécifiques de chaque plateformes

Architecture WebView

Les pré-requis

- Node.js & npm
- Visual Studio Code
- Git
- Extension VsCode

Démarrage d'un projet

Initialiser un projet

Pour installer Ionic on installe le package suivant :

- `npm install -g @ionic/cli@latest`

On créé ensuite un projet avec la commande :

- `ionic start <NomProjet> blank --type vue`

Pour lancer le projet depuis le répertoire créé :

- `ionic serve`

Le fichier main.ts

Lors de la création de notre application Vue.js, deux plugins sont ajoutés : `IonicVue` ainsi que le `router`

```
// ...
import { IonicVue } from "@ionic/vue";
const app = createApp(App).use(IonicVue).use(router);

router.isReady().then(() => {
  app.mount("#app");
});
```

Le composant App.vue

Une application Ionic se comporte de la même façon

```
<template>
  <ion-app>
    <ion-router-outlet />
  </ion-app>
</template>

<script setup lang="ts">
import { IonApp, IonRouterOutlet } from "@ionic/vue";
</script>
```

Exemple de composant Ionic

```
<script setup lang="ts">
import {
  IonContent,
  IonHeader,
  IonPage,
  IonTitle,
  IonToolbar,
} from "@ionic/vue";
</script>
<style scoped>
#container strong {
  font-size: 20px;
}
</style>
```

```
<template>
  <ion-page>
    <ion-header :translucent="true">
      <ion-toolbar>
        <ion-title>Mon titre</ion-title>
      </ion-toolbar>
    </ion-header>

    <ion-content :fullscreen="true">
      <div id="container">
        <p>Voici un exemple de composant créé avec Ionic !</p>
      </div>
    </ion-content>
  </ion-page>
</template>
```

IonPage

- Le composant `IonPage` enveloppe chaque vue dans une application Ionic Vue et permet aux transitions de page et à la navigation par pile de fonctionner correctement
- Chaque vue vers laquelle on navigue à l'aide du routeur **doit** inclure un composant `IonPage`

IonIcons

- `Icon` est un composant simple mis à disposition par la bibliothèque **Ionicons**, qui est fournie par défaut avec toutes les applications Ionic Framework
- Il peut être utilisé pour afficher n'importe quelle icône de la bibliothèque Ionicons, ou un SVG personnalisé
- Il permet également de modifier la taille et la couleur de l'icône

```
<template>
  <ion-icon :icon="logoIonic" size="large" color="primary"></ion-icon>
</template>
```

Appeler une méthode de composant

Pour appeler les méthodes d'un composant Ionic, il est nécessaire de récupérer sa ref via le template ref : `<ion-content ref="content">`

On créé ensuite une variable réactive portant le même nom:

`const content = ref();`

On peut ensuite appeler une méthode du composant avec l'attribut `$el` :

```
const scrollToBottom = () => {
  content.value.$el.scrollToBottom(300);
};
```

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Ionic Router

Fonctionnement du routeur

- Le composant `IonRouterOutlet` utilise la bibliothèque Vue Router sous le capot
- Ionic et Vue Router permettent de créer des applications multi-pages possédant des transitions avancées
- Tout ce qui était utilisé avec Vue Router se retrouve dans Ionic Vue

Création du routeur

- Les routes sont généralement définies dans un fichier nommé `router.ts` :

```ts
import { createRouter, createWebHistory } from "@ionic/vue-router";
import { RouteRecordRaw } from "vue-router";
import HomePage from "@/views/Home.vue";

const routes: Array<RouteRecordRaw> = [
  {
    path: "/",
    name: "Home",
    component: HomePage,
  },
];
```

```ts
const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes,
});

export default router;
```

Gérer les redirections

Si l'on souhaite faire une redirection sur l'origine, on peut utiliser la clé `redirect` pour envoyer sur une autre route

```
const routes: Array<RouteRecordRaw> = [
  {
    path: "/",
    redirect: "/home",
  },
  {
    path: "/home",
    name: "Home",
    component: HomePage,
  },
];
```

Options de navigations

Pour naviguer vers d'autres pages on peut utiliser l'attribut :

```
<ion-button router-link="/detail">Go to detail</ion-button>
```

Ou encore l'instance du router via la méthode `.push`:

```
<template>
  <ion-button @click="() => router.push('/detail')">Page détail</ion-button>
</template>
<script lang="ts">
const router = useRouter();
</script>
```

router-link

- Au lieu d'utiliser des balises `<a>` ordinaires, on utilise l'attribut personnalisé `router-link` pour créer des liens
- Cela permet à Vue Router de modifier l'URL sans recharger la page, de gérer la génération de l'URL ainsi que son encodage

```
<ion-button
  router-link="/page2"
  router-direction="back"
  :router-animation="myAnimation"
>Cliquer ici</ion-button>
```

Navigation avec useIonRouter

useIonRouter permet d'exécuter du code avant de naviguer vers une autre page, chose que l'on ne peut pas pas faire avec l'attribut router-link

```
import { defineComponent } from "vue";
import { useIonRouter } from "@ionic/vue";
import { customAnimation } from "@/animations/customAnimation";

const ionRouter = useIonRouter();

ionRouter.push("/page2", customAnimation);
```

Navigation avec router.go

Vue Router dispose d'une méthode router.go qui permet aux développeurs d'avancer ou de reculer dans l'historique de l'application

`/pageA -> /pageB -> /pageC`

- Si vous appelez `router.go(-2)` sur `/pageC`, vous serez ramené à `/pageA`
- Si vous appelez ensuite `router.go(2)`, vous serez ramené à la page `/pageC`

Lazy loading

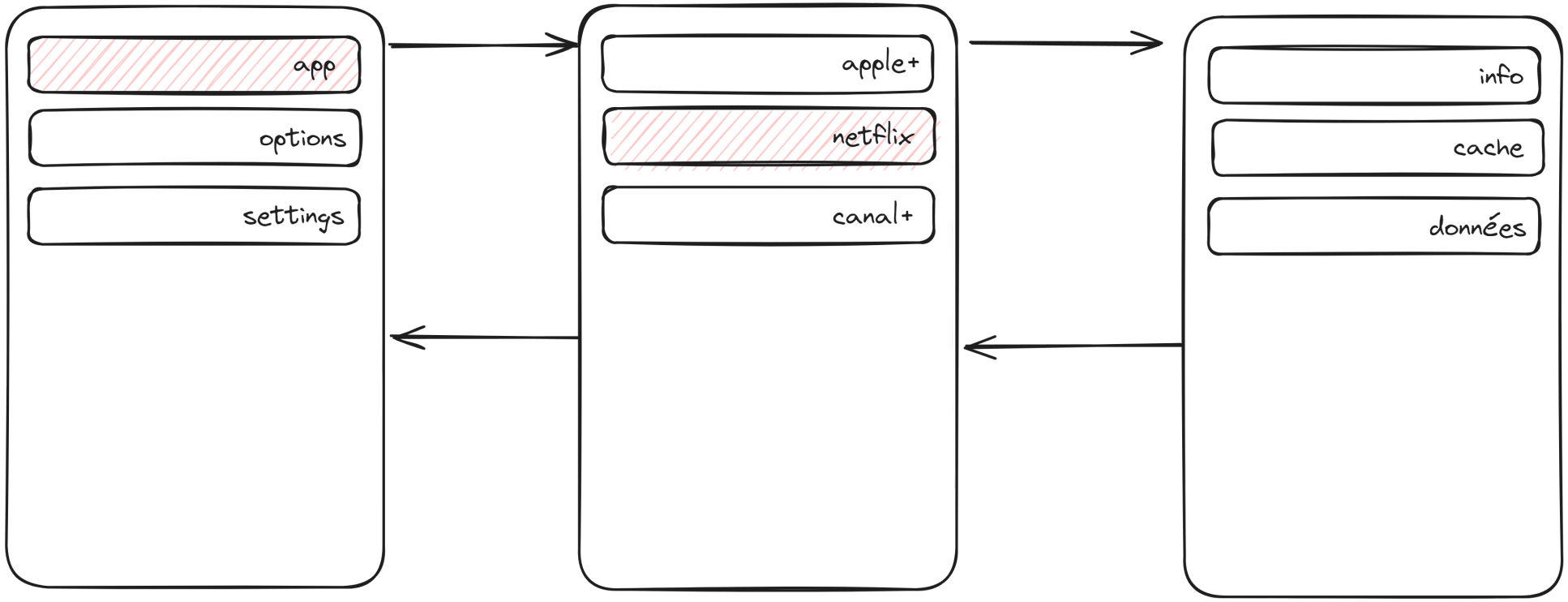
Pour optimiser le chargement de notre application, il est intéressant d'utiliser une méthode appelée le lazy loading

L'intérêt est de charger un composant uniquement lorsque la route définie est appelé

```
const routes: Array<RouteRecordRaw> = [
  {
    path: "/",
    name: "home",
    component: () => import("@/views/HomePage.vue"),
  },
];
```

Routing linéaire

- Le routing linéaire permet d'obtenir des comportements simples et prévisibles



Routing non-linéaire

Le routage non linéaire signifie que la vue à laquelle l'utilisateur doit revenir n'est pas nécessairement la vue précédente qui était affichée à l'écran

Quel routing choisir

- Garder l'application aussi simple que possible jusqu'à ce que vous ayez besoin d'ajouter le routage non linéaire
- Le routage non linéaire est très puissant, mais il ajoute également une quantité considérable de complexité aux applications mobiles
- Les deux utilisations les plus courantes du routage non linéaire sont les onglets et les sorties imbriquées de routeurs ioniques

TOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Application Tabs

Les onglets

Lorsque l'on travaille avec des onglets, Ionic Vue a besoin d'un moyen de savoir quelle vue appartient à quel onglet

Le composant IonTabs est très utile dans ce cas

Création d'onglets

`router.ts`

```ts
const routes: Array<RouteRecordRaw> = [
  {
    path: "/tabs/",
    component: Tabs,
    children: [
      {
        path: "tab1",
        component: () => import("@/views/Tab1.vue"),
      },
      // ...
    ],
  },
];
```

`Tabs.vue`

```vue
<ion-tabs>
  <ion-router-outlet></ion-router-outlet>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1" href="/tabs/tab1">
      <ion-icon :icon="triangle" />
      <ion-label>Tab 1</ion-label>
    </ion-tab-button>
  </ion-tab-bar>
  <!-- ... -->
</ion-tabs>
```

Fonctionnement des Tabs

- Chaque onglet est traité comme une pile de navigation individuelle
- Cela signifie que si vous avez trois onglets dans votre application, chaque onglet a sa propre pile de navigation
- Dans chaque pile, vous pouvez naviguer vers l'avant (pousser une vue) et vers l'arrière (sortir une vue)

IonRouterOutlet

- Le composant `IonRouterOutlet` fournit un conteneur pour rendre des vues.
- Il est similaire au composant `RouterView` que l'on trouve dans d'autres applications Vue, sauf que `IonRouterOutlet` peut rendre plusieurs pages dans le DOM dans la même sortie
- Lorsqu'un composant est rendu dans `IonRouterOutlet`, nous considérons qu'il s'agit d'une "page" Ionic Framework
- ⚠️ Rien ne doit être fourni à l'intérieur de `IonRouterOutlet` lorsque vous le configurez dans votre modèle

Hook de cycle de vie

Les hooks

Nom de l'événement	Description
ionViewWillEnter	Déclenché lorsque le composant vers lequel il est routé est sur le point de s'animer en vue
ionViewDidEnter	Déclenché lorsque l'animation du composant vers lequel il est dirigé est terminée
ionViewWillLeave	Déclenché lorsque le composant routing from est sur le point de s'animer
ionViewDidLeave	Déclenché lorsque l'animation du composant routing from est terminée

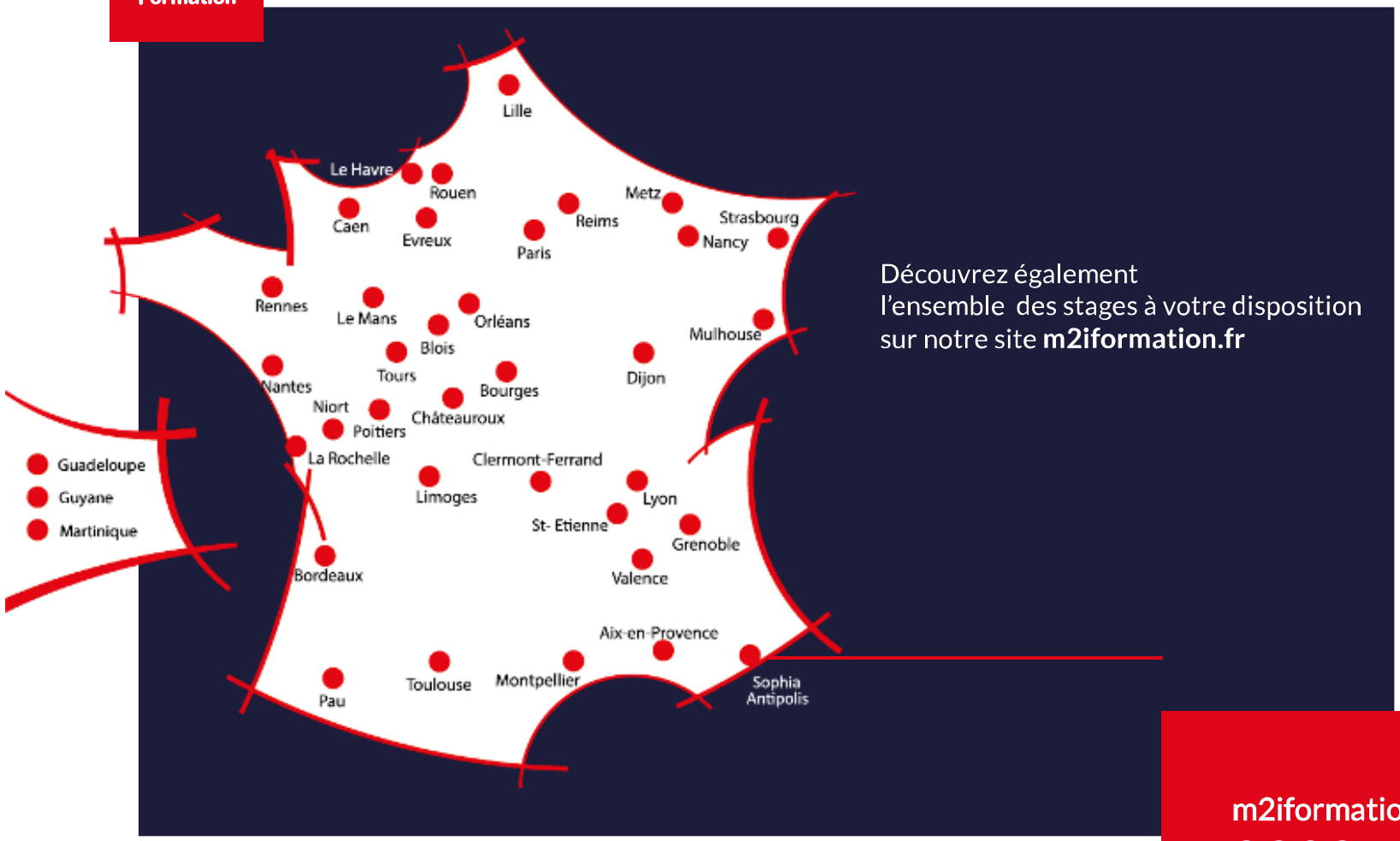
Gestion de vie d'une page

Lorsqu'une application est enveloppée dans `<ion-router-outlet>`, Ionic Framework conserve l'ancienne page dans le DOM existant, mais la cache de la vue et assure la transition vers la nouvelle page, le but étant de:

- Conserver l'état de l'ancienne page (données à l'écran, position de défilement, etc...)
- Assurer une transition plus douce vers la page puisqu'elle est déjà là et n'a pas besoin d'être créée

Merci pour votre attention

Des questions ?



Découvrez également
l'ensemble des stages à votre disposition
sur notre site **m2iformation.fr**

m2iformation.fr