??? ????????: interface GenerateMenuAndRoutesOptions { fetchMenuListAsync?: () => ...

```
interface GenerateMenuAndRoutesOptions {
  fetchMenuListAsync?: () => Promise<RouteRecordStringComponent[]>;
  forbiddenComponent?: RouteRecordRaw['component'];
  layoutMap?: ComponentRecordType;
  pageMap?: ComponentRecordType;
  roles?: string[];
  router: Router;
  routes: RouteRecordRaw[];
}
```

Это **TypeScript-интерфейс**, который описывает **объект с настройками** для функции (скорее всего `generateAccessible`).
Разберём **что означает КАЖДАЯ строка и зачем это нужно** □□

????? ?????

```
interface GenerateMenuAndRoutesOptions { ... }
```

□□ Это **контракт**:

- какие поля **можно / нужно** передать
- какие из них **обязательные**
- какие **опциональные**
- каких **типов** они должны быть

?????? ?? ?????

? `fetchMenuListAsync?: () => Promise<RouteRecordStringComponent[]>;`

- `?` → **необязательное**
- тип → **функция**
- без аргументов
- возвращает **Promise**
- Promise резолвится в массив `RouteRecordStringComponent`

◫ То есть:

```
fetchMenuListAsync: async () => {
  return getAllMenusApi()
}
```

◫ **callback**, который:

- асинхронно получает меню / маршруты
- обычно с бэкенда

? `forbiddenComponent?: RouteRecordRaw['component'];`

Это:

> ❝ «возьми тип поля `component` из `RouteRecordRaw`»

Фактически:

```
Component | (() => Promise<Component>)
```

Используется как:

- компонент **403 / Forbidden**
- или fallback при отсутствии прав

? `layoutMap?: ComponentRecordType;`

Карта layout-компонентов:

```
{
  BasicLayout: BasicLayout,
  IFrameView: IFrameView,
}
```

Используется, чтобы:

- сопоставлять строку `"BasicLayout"`
- с реальным Vue-компонентом

? `pageMap?: ComponentRecordType;`

Карта страниц:

```
import.meta.glob('../views/**/*.vue')
```

Используется, чтобы:

- по строке `"Users"`
- найти компонент `Users.vue`

? `roles?: string[];`

Массив ролей пользователя:

```
['admin', 'editor']
```

Используется для:

- фильтрации маршрутов
- проверки доступа

? `router: Router;` ? (??????????)

Экземпляр **Vue Router**:

```
const router = createRouter(...)
```

Используется для:

- `router.addRoute()`
- управления навигацией

? `routes: RouteRecordRaw[];` ? (?????????????)

Массив **базовых маршрутов**:

```
[
  { path: '/login', component: Login },
  { path: '/404', component: NotFound }
]
```

Это:

- статические маршруты
- которые существуют всегда

????????????? vs ??????????????

Поле	Обязательное
`router`	□
`routes`	□
`fetchMenuListAsync`	□
`layoutMap`	□
`pageMap`	□
`roles`	□
`forbiddenComponent`	□

??? ??? ????????????? ?? ????????

```
generateAccessible(mode, {
  router,
  routes,
  pageMap,
  layoutMap,
  fetchMenuListAsync,
})
```

TypeScript:

- проверяет, что объект корректный
- подсказывает поля
- ловит ошибки на этапе разработки

???????

> Этот интерфейс описывает **набор параметров**,
> которые нужны для **генерации меню и маршрутов с доступами**
> (часто — динамически и с бэкенда).

Revision #2
Created 2026-01-20 09:06:36 UTC by Nikolay
Updated 2026-01-20 09:13:13 UTC by Nikolay