Introduction à Django REST Framework

Xavier Dutreilh

Rencontres Django 2015

Qu'est-ce que REST ?

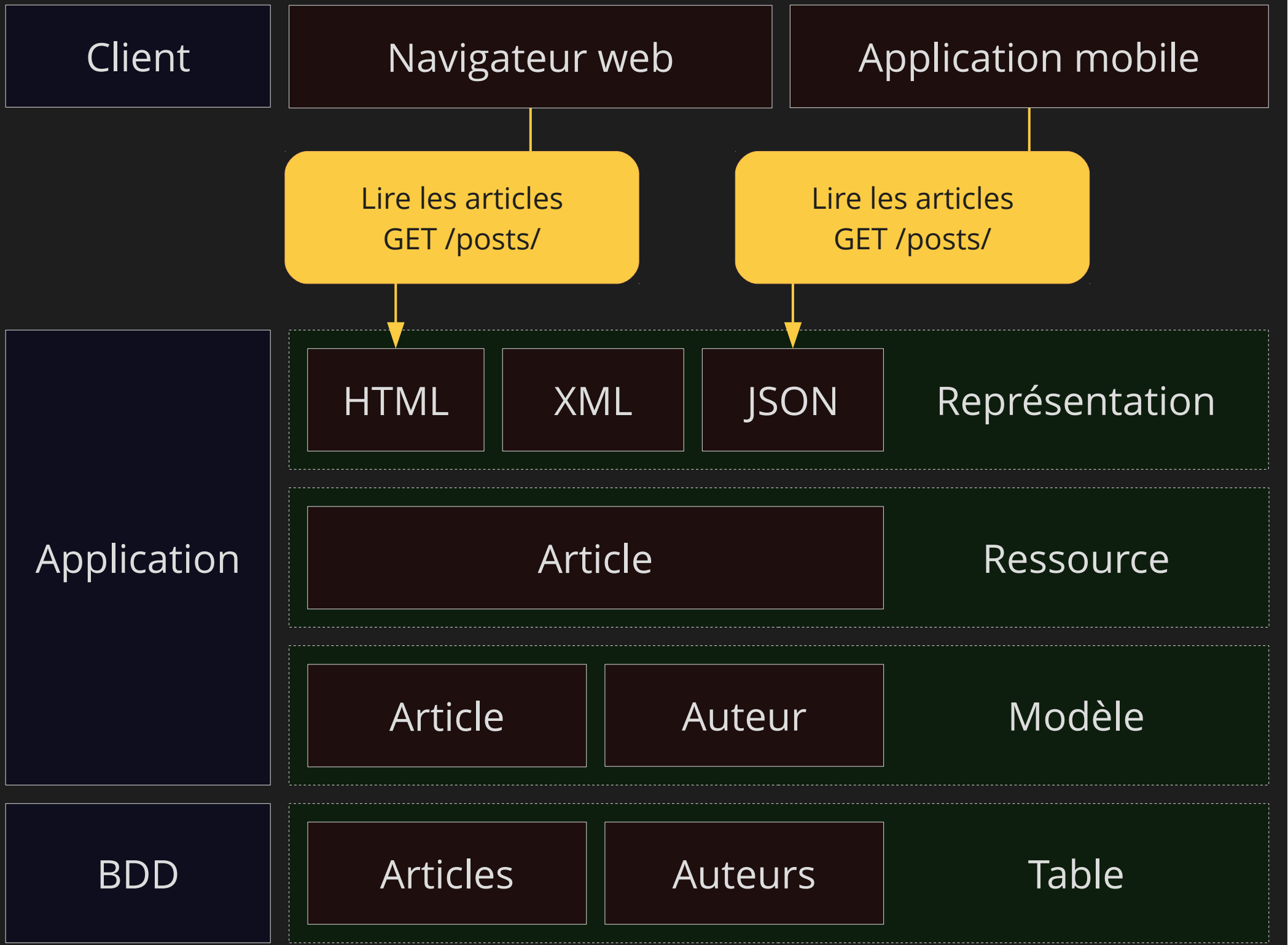
acronyme de
REpresentational
State **T**ransfer

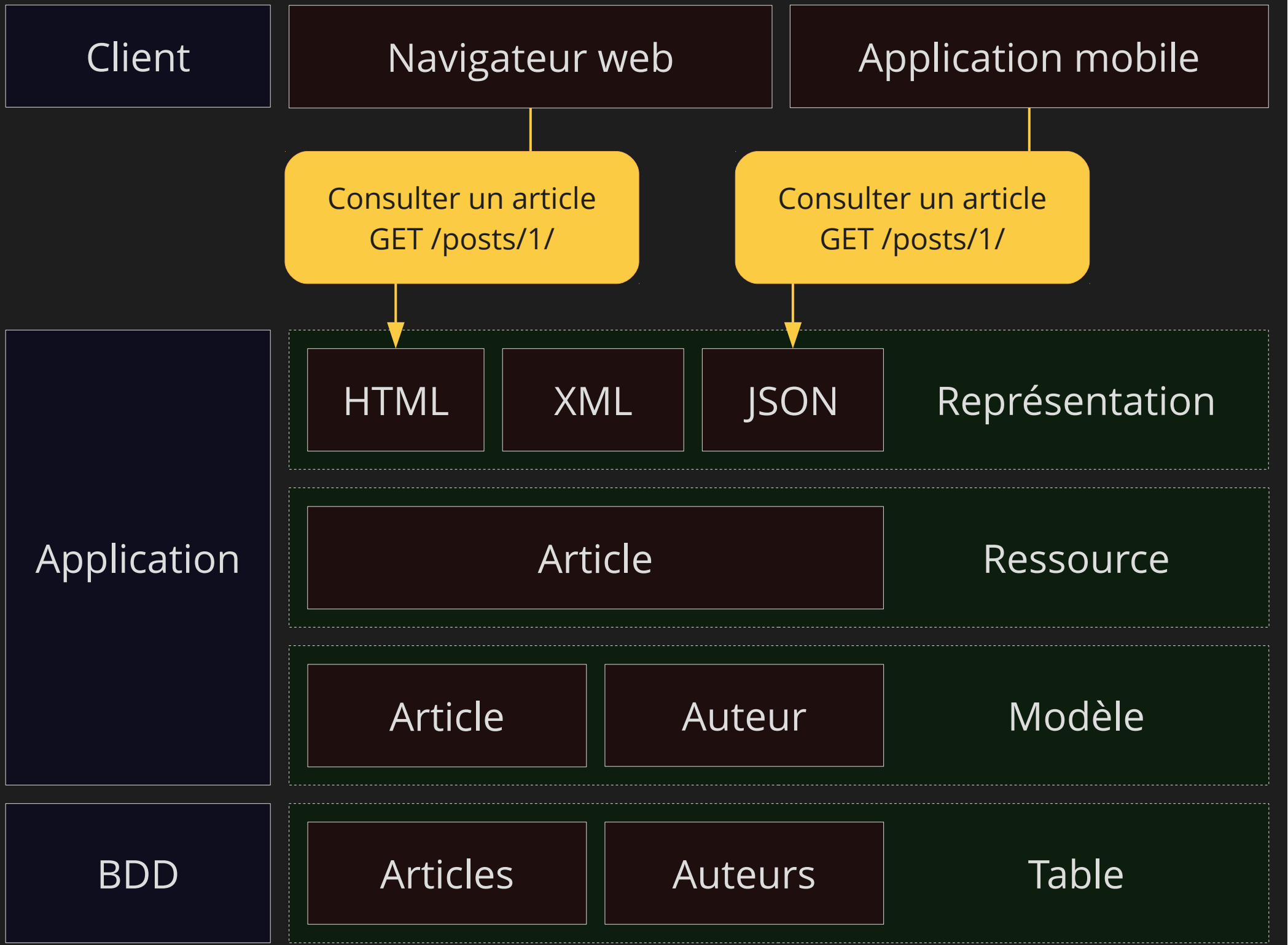
style d'architecture pour les systèmes hypermédia distribués

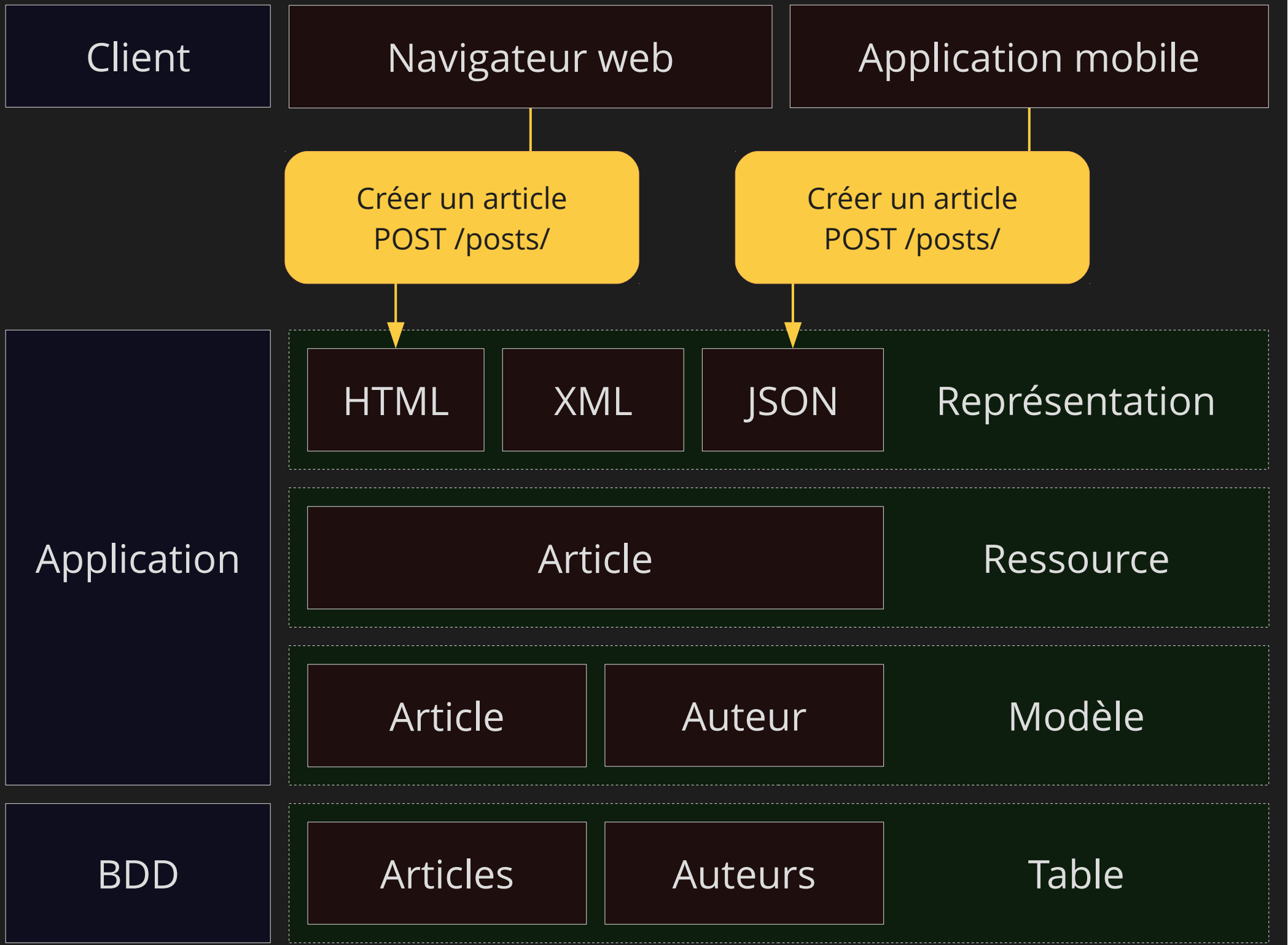
invention de
Roy Fielding
dans sa thèse
de doctorat

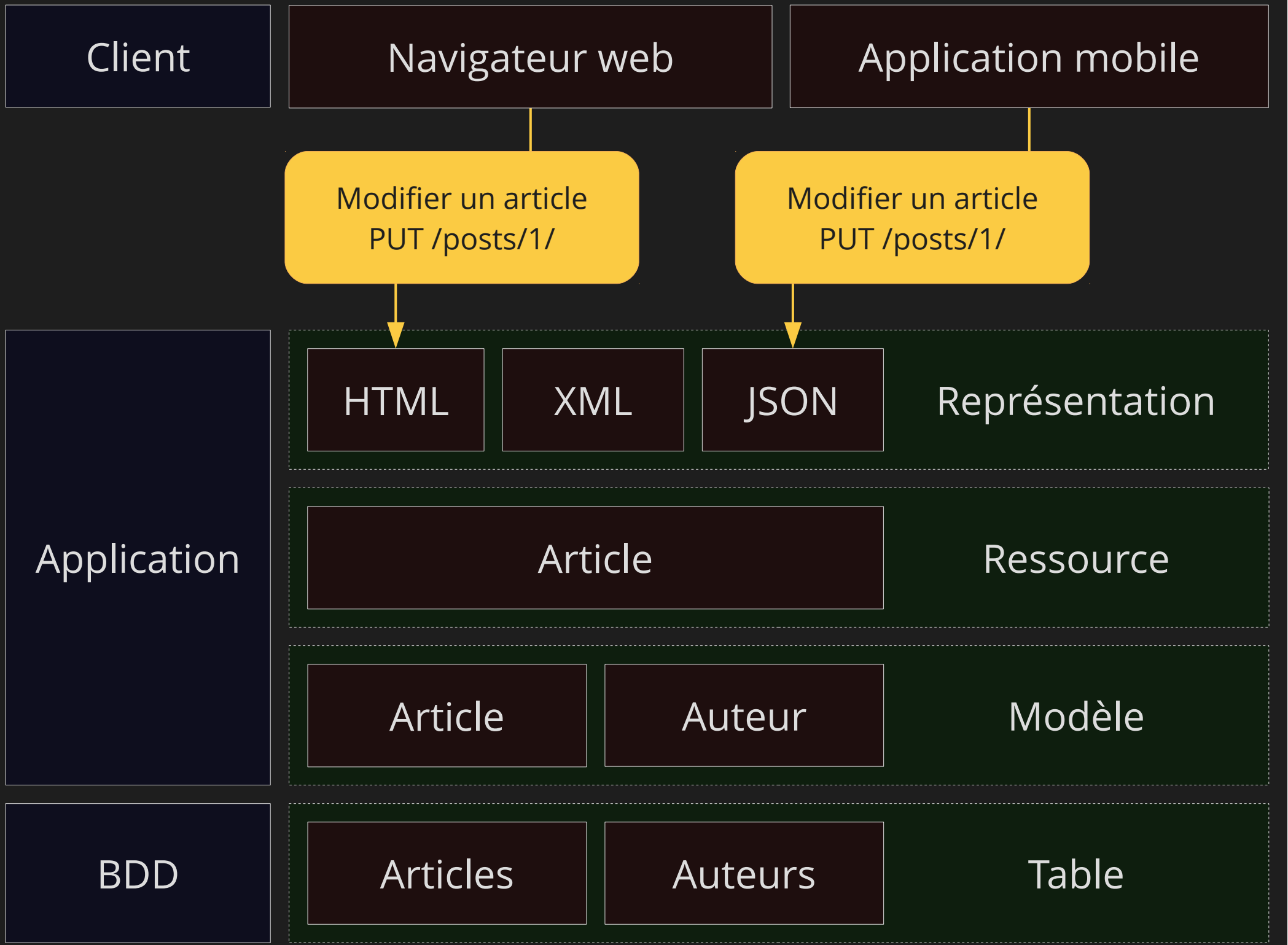
application à
de nombreux
domaines,
dont le web

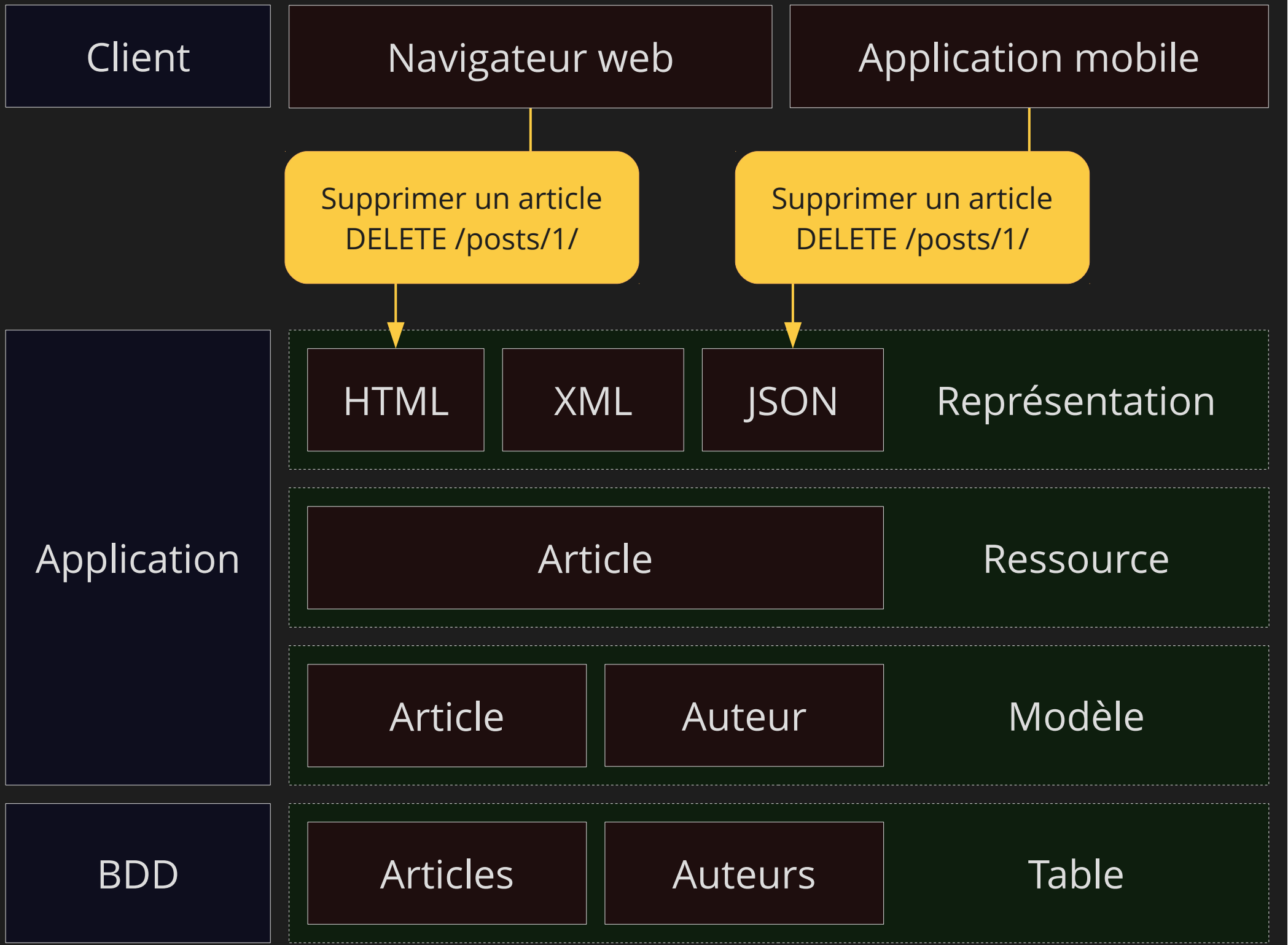
Comment peut-on appliquer REST à un blog ?


Client
Navigateur web
Application mobile
Lire les articles
GET /posts/
Lire les articles
GET /posts/
Application
HTML
XML
JSON
Représentation
Article
Ressource
Article
Auteur
Modèle
BDD
Articles
Auteurs
Table


Client
Navigateur web
Application mobile
Consulter un article
GET /posts/1/
Consulter un article
GET /posts/1/
Application
HTML
XML
JSON
Représentation
Article
Ressource
Article
Auteur
Modèle
BDD
Articles
Auteurs
Table


Client
Navigateur web
Application mobile
Créer un article
POST /posts/
Créer un article
POST /posts/
Application
HTML
XML
JSON
Représentation
Article
Ressource
Article
Auteur
Modèle
BDD
Articles
Auteurs
Table

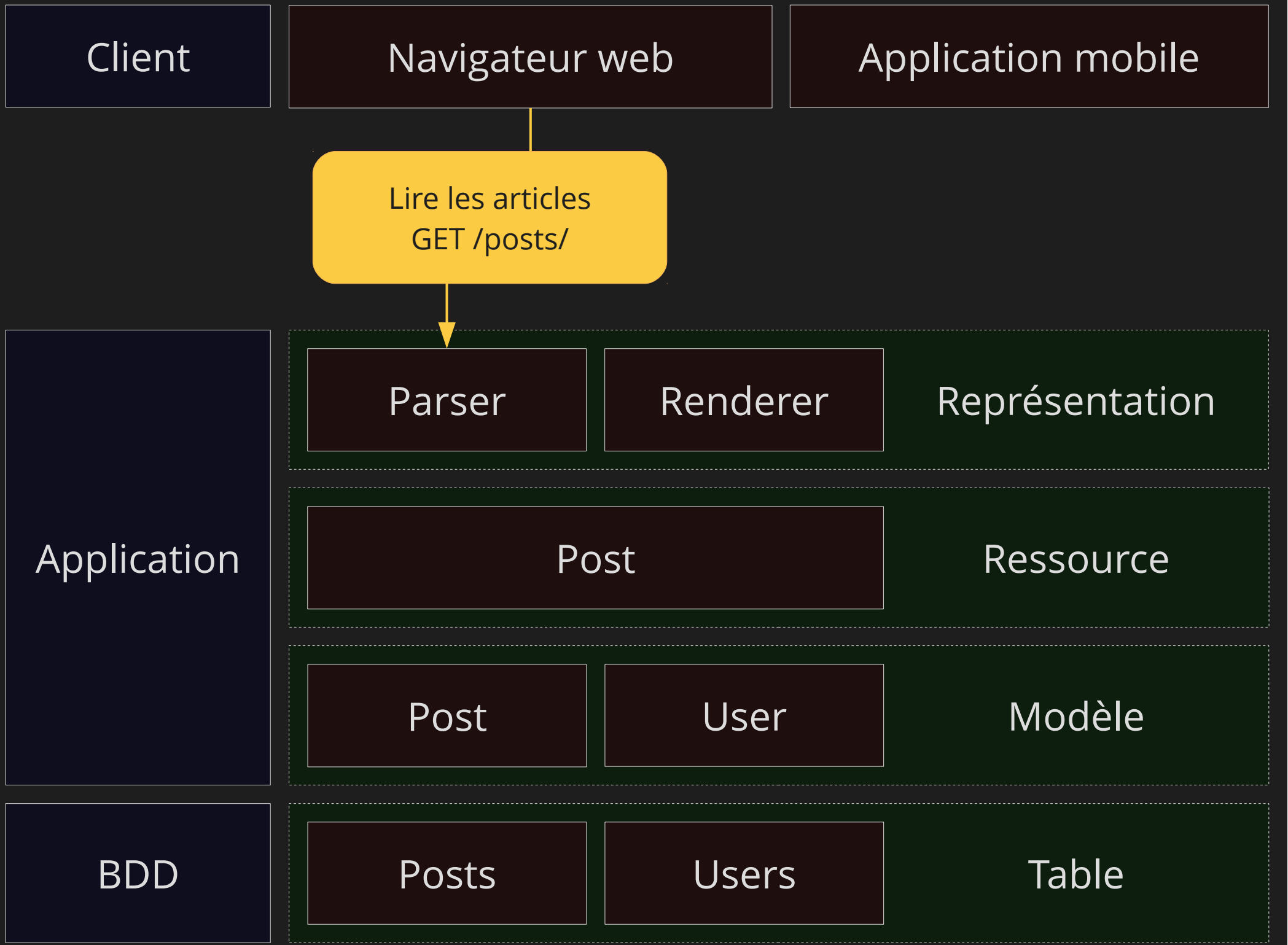

Client
Navigateur web
Application mobile
Modifier un article
PUT /posts/1/
Modifier un article
PUT /posts/1/
Application
HTML
XML
JSON
Représentation
Article
Ressource
Article
Auteur
Modèle
BDD
Articles
Auteurs
Table

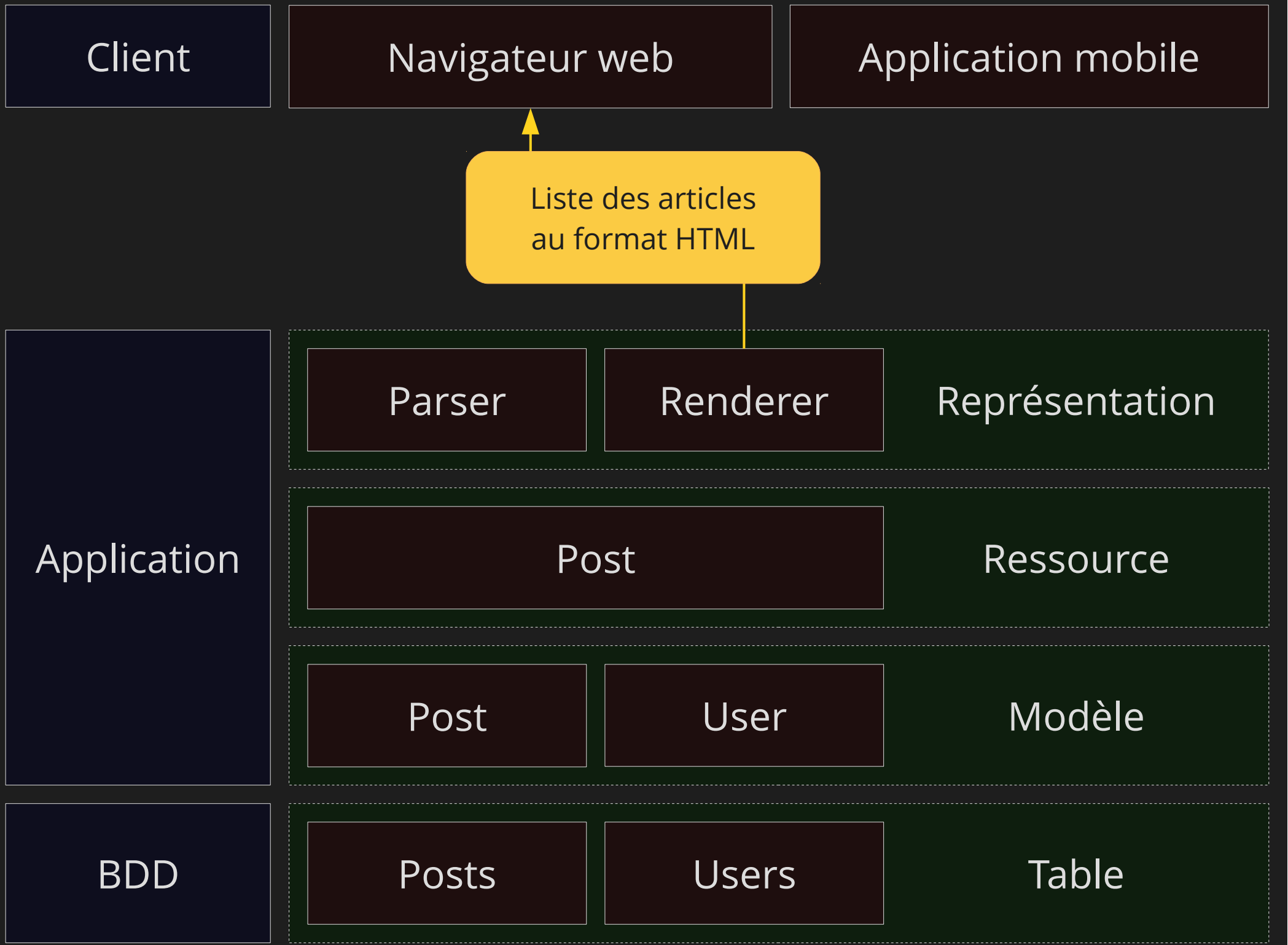

Client
Navigateur web
Application mobile
Supprimer un article
DELETE /posts/1/
Supprimer un article
DELETE /posts/1/
Application
HTML
XML
JSON
Représentation
Article
Ressource
Article
Auteur
Modèle
BDD
Articles
Auteurs
Table

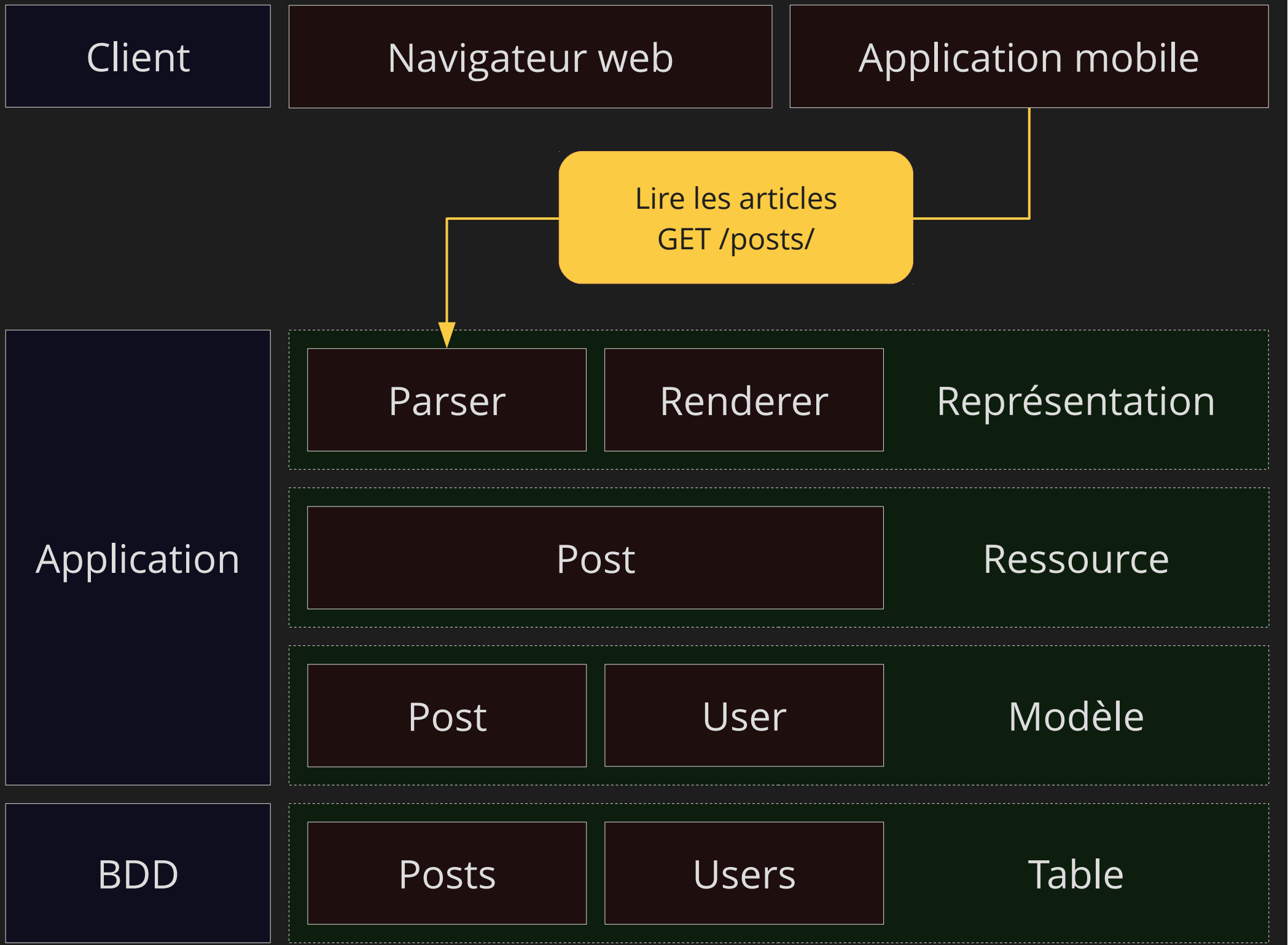
Qu'est-ce que Django REST Framework ?

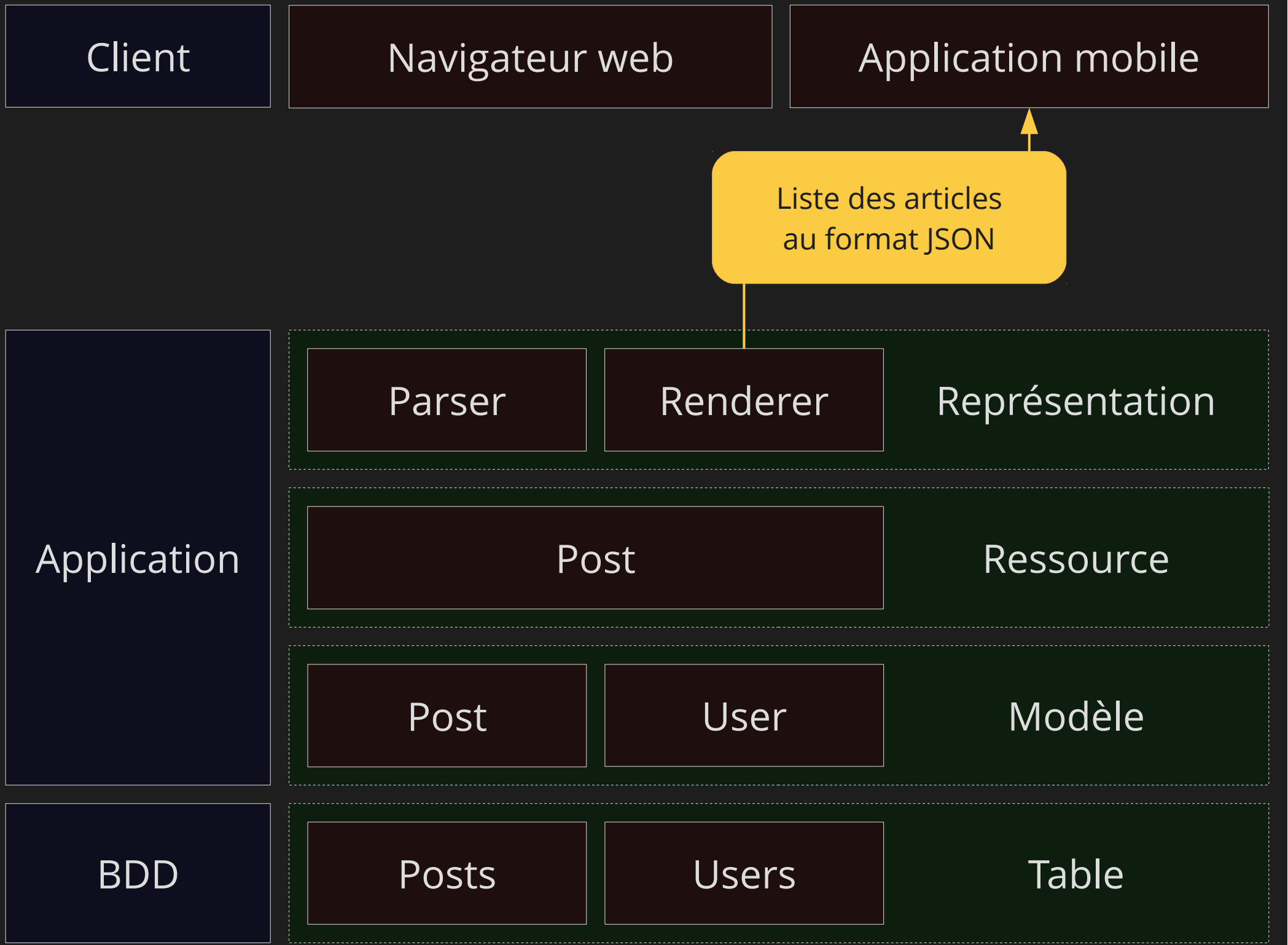
boîte à outils
de création
d'architecture
REST avec Django

Comment peut-on construire un blog avec Django REST Framework ?


Client
Navigateur web
Application mobile
Lire les articles
GET /posts/
Application
Parser
Renderer
Représentation
Post
Ressource
Post
User
Modèle
BDD
Posts
Users
Table


Client
Navigateur web
Application mobile
Liste des articles
au format HTML
Application
Parser
Renderer
Représentation
Post
Ressource
Post
User
Modèle
BDD
Posts
Users
Table


Client
Navigateur web
Application mobile
Lire les articles
GET /posts/
Application
Parser
Renderer
Représentation
Post
Ressource
Post
User
Modèle
BDD
Posts
Users
Table


Client
Navigateur web
Application mobile
Liste des articles
au format JSON
Application
Parser
Renderer
Représentation
Post
Ressource
Post
User
Modèle
BDD
Posts
Users
Table

```python
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

```
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

```
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

```
class Post(Model):

    title = CharField(max_length=100)
    author = ForeignKey(User)
    content = TextField()
    date_created = DateTimeField(auto_now_add=True)
    date_updated = DateTimeField(auto_now=True)

    def __str__(self):
      return self.title
```

Activation du filtrage et du tri de la vue

```
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

Configuration des critères de filtrage de la vue

```
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

Définition des critères de filtrage de la vue

```
class PostFilter(FilterSet):

    title = CharFilter(lookup_type='icontains')
    content = CharFilter(lookup_type='icontains')

    class Meta:

        model = Post
        fields = ('title', 'content')
```

Définition des critères de tri de la vue

```
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

Définition des critères de tri par défaut de la vue

```
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

Configuration de la pagination de la vue

```
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

Définition de la pagination de la vue

```
class PostPagination(PageNumberPagination):

    page_size = 10
```

Configuration des permissions de la vue

```
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

Définition des permissions de la vue

```
class IsAuthorOrReadOnly(IsAuthenticatedOrReadOnly):

    def has_object_permission(self, request, view, obj):
        return (
            request.method in SAFE_METHODS or
            obj.author == request.user
        )
```

Configuration du sérialiseur de la vue

```
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

Définition du sérialiseur de la vue

```
class PostSerializer(HyperlinkedModelSerializer):

    author = SerializerMethodField()

    class Meta:

        model = Post
        fields = ('url', 'title', 'author', 'content', 'date_created',
            'date_updated')

    def get_author(self, obj) :
      return obj.author.get_full_name()
```

Affectation automatique de l'auteur

```
class PostViewSet(ModelViewSet):

    queryset = Post.objects.all()
    filter_backends = (DjangoFilterBackend, OrderingFilter)
    filter_class = PostFilter
    ordering_fields = ('title', 'date_created', 'date_updated')
    ordering = ('-date_updated',)
    pagination_class = PostPagination
    permission_classes = (IsAuthorOrReadOnly,)
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)
```

Où peut-on
voir une démo ?

Django REST framework v 3.1.1 xavier

Api Root / Post List

Post List

OPTIONS GET

```
GET /posts/
```

```
HTTP 200 OK
Vary: Accept
Content-Type: application/json
Allow: GET, POST, HEAD, OPTIONS

{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "url": "http://localhost:8000/posts/1/",
            "title": "Hello, world!",
            "author": "Xavier Dutreilh",
            "content": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. In vulputate ipsum a tincidunt venenatis. Aliquam eu turpis a lorem finibus
            "date_created": "2015-05-04T07:55:51.605879Z",
            "date_updated": "2015-05-04T07:58:52.012621Z"
        }
    ]
}
```

Raw data | HTML form

Title

Content

POST

code source
disponible
sur GitHub

Merci !

Xavier Dutreilh
xavier@dutreilh.com
http://xavier.dutreilh.com/