

# Modèle C4

4 niveaux de zoom :

- Context
- Container
- Component
- Code.

Représente le même système à plusieurs niveaux de détail.

Editeur: <https://playground.structurizr.com>

vue très large : où se place ton système ?

## Les 4 C

### Context

Vue très large : Ton système vu de l'extérieur, "Ce système sert à quoi, pour qui, et avec quoi il parle ?". On y voit :

- les utilisateurs,
- les systèmes externes,
- ton application au milieu,
- les grandes interactions.
- 

Exemple :

- Client
- Admin
- Shop Platform
- Stripe
- Service de mails

### Container

Vue intermédiaire : quels gros blocs le composent (blocs exécutable ou stockant des données), "Techniquement, mon système est découpé en quels gros morceaux?".

Exemple:

- Frontend Web App (Nuxt/Vue)
- Backend API (Node.js ou PHP)
- MariaDB
- Worker d'envoi d'e-mails

## Component

Vue fine : comment un bloc est organisé, "À l'intérieur de ce bloc, quels sont les modules importants et leurs responsabilités ?"?

Exemple pour une API :

- Auth Controller
- Order Service
- Payment Service
- Product Repository
- Email Gateway

## Code

Vue très fine : à quoi ressemble le code / design interne ?

Niveau "Code" prévu, mais en pratique il est souvent optionnel, "Comment ce composant est structuré en classes, interfaces, modules, etc. ?".

## Exemple

```
workspace "JustNote Reloaded" "Système de gestion de carnets pour institutions pénitentiaires belges" {  
  
    model {  
        user = person "Utilisateur Prison" "Membre du personnel pénitentiaire authentifié"  
        admin = person "Administrateur" "Utilisateur avec privilèges élevés"  
  
        ldapServer = softwareSystem "Serveur LDAP/Active Directory" "Microsoft Active
```

Directory"

```
smtpServer = softwareSystem "Serveur SMTP" "SMTP"
```

```
justNoteSystem = softwareSystem "JustNote Reloaded" "Système de gestion de carnets" {
```

```
  frontendApp = container "Application Web Frontend" "Application SPA" "Nuxt 4, Vue  
3, Pinia" {
```

```
    frontendAuth = component "Module Auth" "Gestion authentification" "Frontend  
module"
```

```
    frontendRpcClient = component "Client RPC" "Client JSON-RPC" "Frontend  
service"
```

```
    frontendStore = component "State Management" "Gestion état centralisée" "Pinia  
store"
```

```
    pages = component "Pages" "Composants de page" "Vue components"
```

```
    composables = component "Composables" "Logique métier" "Vue composables"
```

```
    uiComponents = component "Composants UI" "Interface utilisateur" "Vue  
components"
```

```
    pages -> frontendAuth "Utilise"
```

```
    pages -> composables "Utilise"
```

```
    pages -> uiComponents "Affiche"
```

```
    frontendAuth -> frontendRpcClient "Appelle"
```

```
    frontendStore -> frontendRpcClient "Gère tokens"
```

```
    composables -> frontendRpcClient "Appelle RPC"
```

```
  }
```

```
  backendApi = container "API Backend RPC" "API JSON-RPC 2.0" "PHP 8.4, Apache" {
```

```
    rpcDispatcher = component "RPC Dispatcher" "Gestion protocole JSON-RPC" "PHP  
component"
```

```
    authMiddleware = component "Auth Middleware" "Validation JWT" "PHP middleware"
```

```
    serviceContainer = component "Container DI" "Injection dépendances" "PHP  
service container"
```

```
    authController = component "Auth Controller" "login, logout, refresh" "PHP  
controller"
```

```
    usersController = component "Users Controller" "CRUD utilisateurs" "PHP  
controller"
```

```
    notebooksController = component "Notebooks Controller" "CRUD carnets" "PHP  
controller"
```

```
    entriesController = component "Entries Controller" "CRUD entrées" "PHP  
controller"
```

```
    sectionsController = component "Sections Controller" "CRUD sections" "PHP"
```

```
controller"
    absencesController = component "Absences Controller" "CRUD absences" "PHP
controller"
    remindersController = component "Reminders Controller" "CRUD rappels" "PHP
controller"
    adminControllers = component "Admin Controllers" "Méthodes admin" "PHP
controller"
    publicControllers = component "Public Controllers" "Méthodes publiques" "PHP
controller"
    dbService = component "Database Service" "Wrapper ORM Medoo" "PHP service"
    jwtService = component "JWT Service" "Gestion tokens JWT" "PHP service"
    ldapService = component "LDAP Service" "Authentification AD" "PHP service"
    refreshTokenService = component "Refresh Token Service" "Rotation tokens" "PHP
service"
    accessControlService = component "Access Control Service" "Évaluation
permissions" "PHP service"
    mailerService = component "Mailer Service" "Envoi emails" "PHP service"

    rpcDispatcher -> authMiddleware "Passe par"
    authMiddleware -> serviceContainer "Résout services"
    authController -> jwtService "Utilise"
    authController -> ldapService "Utilise"
    authController -> refreshTokenService "Utilise"
    authController -> dbService "Utilise"
    usersController -> dbService "Utilise"
    usersController -> accessControlService "Vérifie permissions"
    notebooksController -> dbService "Utilise"
    notebooksController -> accessControlService "Vérifie permissions"
    entriesController -> dbService "Utilise"
    entriesController -> accessControlService "Vérifie permissions"
    sectionsController -> dbService "Utilise"
    absencesController -> dbService "Utilise"
    absencesController -> accessControlService "Vérifie permissions"
    remindersController -> dbService "Utilise"
    remindersController -> accessControlService "Vérifie permissions"
    remindersController -> mailerService "Envoie notifications"
    adminControllers -> dbService "Utilise"
    adminControllers -> accessControlService "Vérifie permissions"
    publicControllers -> dbService "Utilise"
}
```

```

database = container "Base de données" "Stockage relationnel" "MariaDB 11.8" {
    tags "Database"

    usersTable = component "Table users" "Comptes utilisateurs" "Table"
    rolesTable = component "Table roles" "Définitions rôles" "Table"
    userRolesTable = component "Table user_roles" "Assignations rôle" "Table"
    orgsTable = component "Table organizations" "Organisations" "Table"
    placesTable = component "Table places" "Établissements" "Table"
    notebooksTable = component "Table notebooks" "Carnets" "Table"
    sectionsTable = component "Table sections" "Sections" "Table"
    entriesTable = component "Table entries" "Entrées" "Table"
    absencesTable = component "Table absences" "Absences" "Table"
    absenceTypesTable = component "Table absence_types" "Types absences" "Table"
    remindersTable = component "Table reminders" "Rappels" "Table"
    translationsTable = component "Table translations" "Traductions" "Table"
    groupsTable = component "Table groups" "Groupes" "Table"
    permissionsTable = component "Table permissions" "Permissions" "Table"
    logsTable = component "Table logs" "Audit" "Table"
    refreshTokensTable = component "Table refresh_tokens" "Tokens refresh" "Table"
}

dbAdmin = container "Admin DB" "Interface admin DB" "phpMyAdmin"

frontendApp -> backendApi "Appelle JSON-RPC" "HTTP/JSON"
backendApi -> frontendApp "Retourne réponses" "HTTP/JSON"

backendApi -> database "Lit/écrit" "MySQL"
database -> backendApi "Retourne données" "MySQL"

backendApi -> ldapServer "Authentifie" "LDAPS"
ldapServer -> backendApi "Retourne résultat" "LDAPS"

backendApi -> smtpServer "Envoie emails" "SMTP"
dbAdmin -> database "Administre" "MySQL"
}

user -> justNoteSystem "Utilise"
admin -> justNoteSystem "Administre"

user -> frontendApp "Utilise"
admin -> frontendApp "Utilise"

```

```
frontendApp -> user "Affiche le dashboard"
}

views {
  systemContext justNoteSystem "SystemContext" "Vue contexte système" {
    include *
    autolayout lr
  }

  container justNoteSystem "ContainerView" "Vue conteneurs" {
    include *
    autolayout lr
  }

  component frontendApp "FrontendComponents" "Vue composants Frontend" {
    include *
    autolayout tb
  }

  component backendApi "BackendComponents" "Vue composants Backend" {
    include *
    autolayout tb
  }

  component database "DatabaseComponents" "Vue composants Database" {
    include *
    autolayout tb
  }

  dynamic justNoteSystem "AuthFlow" "Flux authentication" {
    user -> frontendApp "1. Saisit identifiants"
    frontendApp -> backendApi "2. Appelle Auth.login"
    backendApi -> ldapServer "3. Valide LDAPS"
    ldapServer -> backendApi "4. Retourne résultat"
    backendApi -> database "5. Charge utilisateur"
    database -> backendApi "6. Retourne données"
    backendApi -> frontendApp "7. Retourne tokens"
    frontendApp -> user "8. Redirige dashboard"
  }

  styles {
```

```
    element "Person" {
      background #08427b
      color #ffffff
    }
    element "Software System" {
      background #1168bd
      color #ffffff
    }
    element "Container" {
      background #438dd5
      color #ffffff
    }
    element "Component" {
      background #85bbf0
      color #000000
    }
    element "Database" {
      shape cylinder
    }
  }

  theme default
}
}
```

---

Révision #4

Créé 2026-03-03 13:28:25 CET par Larananas

Mis à jour 2026-03-03 14:37:03 CET par Larananas