

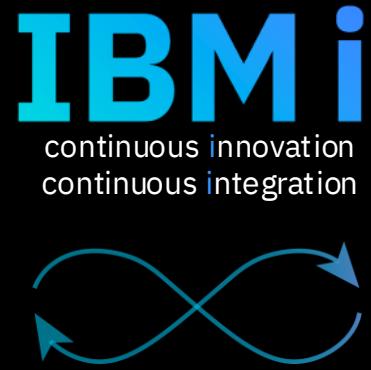
IBM Bob : la puissance de l'IA pour le développement et la modernisation des applications IBM i



Webinar IBM / COMMON France

30 janvier 2026

Philippe Bourgeois
Consultant IBM i
IBM France
pbourgeois@fr.ibm.com



WARNING



- 1. Au 30 janvier 2026 IBM Bob n'est pas encore disponible en GA :
 - Il est en **Technology Preview**
 - La GA est normalement prévue pour le 1er trimestre 2026
- *1. Les informations présentées dans ce webinar sont donc susceptibles d'évoluer dans les semaines et les mois à venir, entre-autres les spécificités IBM i*
- 2. Ce webinar ne dure qu'1h15 ... impossible de faire le tour de toutes les fonctionnalités de Bob
- *D'autres webinars et événements sur IBM Bob auront lieu dans les semaines et les mois à venir*

Plan de la présentation



- **A. IBM Bob – Introduction**
 - Qu'est-ce que IBM Bob – Positionnement – Différentiateurs
 - Que peut apporter Bob pour les équipes IBM i ?
 - Installation, démarrage, configuration, fonctionnalités principales
- **B. Modernisation** des applications IBM i avec Bob
 - 1. Compréhension, explication, documentation de code et d'applications
 - 2. Amélioration, optimisation de code
 - 3. Restructuration, transformation, conversion de code et d'applications
- **C. Développement** de programmes et d'applications IBM i avec Bob
 - 1. Développement de code
 - 2. Développement d'applications
- **D. IBM Bob – Pour aller plus loin**
 - 1. Bilan des tests effectués
 - 2. Comment tester Bob, se préparer à Bob, et comment IBM peut vous aider ?
 - 3. Documentation – Prochains webinars et événements COMMON France

IBM i

continuous innovation
continuous integration



A. IBM Bob – Introduction

Qu'est-ce que IBM Bob ?



- IBM Bob est un **partenaire logiciel** basé sur l'**IA** qui peut vous aider dans toutes les tâches :
 - Du cycle de vie du **développement logiciel** (planification, analyse, conception, développement, tests, déploiement, maintenance)
 - De **modernisation des applications** existantes (nettoyage, documentation, normalisation, restructuration, conversion, migration...)
- Il s'adresse à **tous les profils** IT travaillant sur les applications :
 - Analystes
 - Architectes
 - Développeurs
 - Chefs de projets
 - Responsables applicatifs...

Caractéristiques d'IBM Bob



- Bob a une **compréhension** complète du code, des standards, des dépendances et de l'architecture des applications
- Bob transforme les intentions en modifications logicielles multifichiers, structurées et documentées, via un **cycle itératif** et transparent de planification / exécution / validation
- Bob permet de **déetecter** les problèmes, il peut suggérer des améliorations, refactoriser et optimiser le code
- Bob peut générer de la **documentation** structurée, des plans de **modernisation** à grand échelle et des plans de **tests**
- Bob n'est pas lié à un LLM particulier, il fait de l'orchestration **multi-modèles** (Claude, Granite, Mistral... + des modèles spécialisés)
- Bob peut exécuter des commandes, lancer des tests, appeler des API, inspecter des logs et interagir avec des services externes via le **protocole MCP** (IA **agentique**)

Caractéristiques d'IBM Bob



- Bob n'est pas un produit spécifique à l'IBM i, il a une compréhension de **multiples langages et technologies**
 - Java, Python, JavaScript, TypeScript, Go, C++, Kotlin, Bash, PHP, Ansible...
 - RPG (II, III, IV, Free, Full Free), COBOL, CL, SQL...
- Il permet donc à **toutes** les équipes IT :
 - D'accélérer les cycles de développement
 - De réduire la dette technique

IBM Bob – Positionnement



Extensions IA dans les IDE

- Cline
 - Github Copilot
 - Kilo Code
 - Roo Code
- *Extensions IA dans VS Code ou JetBrains*

IDE avec IA native

- Antigravity
 - Cursor
 - **IBM Bob**
 - Windsurf
- *Conçues pour l'IA*
 - *Multi fichiers*
 - *Multi modèles*
 - *Gestion d'agents*

Générateurs "Prompt-to-App" (Vibe Coding)

- Bolt
 - Lovable
 - Replit Agent
 - V0
- *Génération d'applications complètes à partir de prompts*

Agents IA CLI

- Claude Code
 - Gemini CLI
 - OpenCode
 - **IBM Bob Shell**
- *Mode ligne de commande (terminal)*
 - *Automatisation*
 - *Intégration CI/CD*

Agents "Autonomous Software Engineers"

- Devin
 - Magic.dev
 - OpenHands
- *Création, correction, déploiement d'applications en complète autonomie*

IBM Bob – Différentiateurs



- Bob a un positionnement centré sur l'**entreprise** et la **sécurité** :
 - Bob (a commencé à être et) sera entraîné pour comprendre et moderniser du code **legacy** à grande échelle :
 - RPG et COBOL sous IBM i
 - COBOL sous IBM z
 - Ancien code Java
 - Bob intègre des contrôles de **sécurité** et de **conformité** directement dans son workflow
 - Normalisation des prompts
 - Protection des données sensibles
 - Vérification de la conformité (respect des exigences réglementaires)
 - Bob intégrera des outils de **modernisation** dédiés aux plateformes **IBM i** et **IBM z**

Que peut apporter Bob pour les équipes IBM i ?



- 1. La **documentation** des applications IBM i existantes
 - Pour faciliter l'intégration des nouveaux arrivants... mais pas seulement...
 - *Modèles d'architecture*
 - *Spécifications fonctionnelles et techniques*
 - *Diagrammes des relations...*
- 2. L'amélioration de la **productivité** de développement
 - *Explication de code,*
 - *Génération rapide de code*
 - *Utilisation de normes de développement*
 - *Optimisation du code existant...*
- 3. La **modernisation** des applications IBM i existantes
 - Pour réduire la dette technique et faciliter l'intégration des nouveaux arrivants...
 - *Utilisation de conventions de nommage*
 - *Conversion DDS → SQL*
 - *Conversion RPG fixe → RPG Full Free*
 - *Conversion des accès DB2 RLA → accès SQL*
 - *Extraction de la logique métier*
 - *Refactoring pour utilisation de procédures*
 - *Création de procédures de test*
 - *Création d'une nouvelle interface utilisateur*
 - ...

IBM Bob pour les équipes IBM i – Quelques dates



■ Actuellement

- Bob est en Technology Preview
- Pas de spécificités IBM i dans l'IDE (mais il sait analyser des applications IBM i)

■ Q1 2026

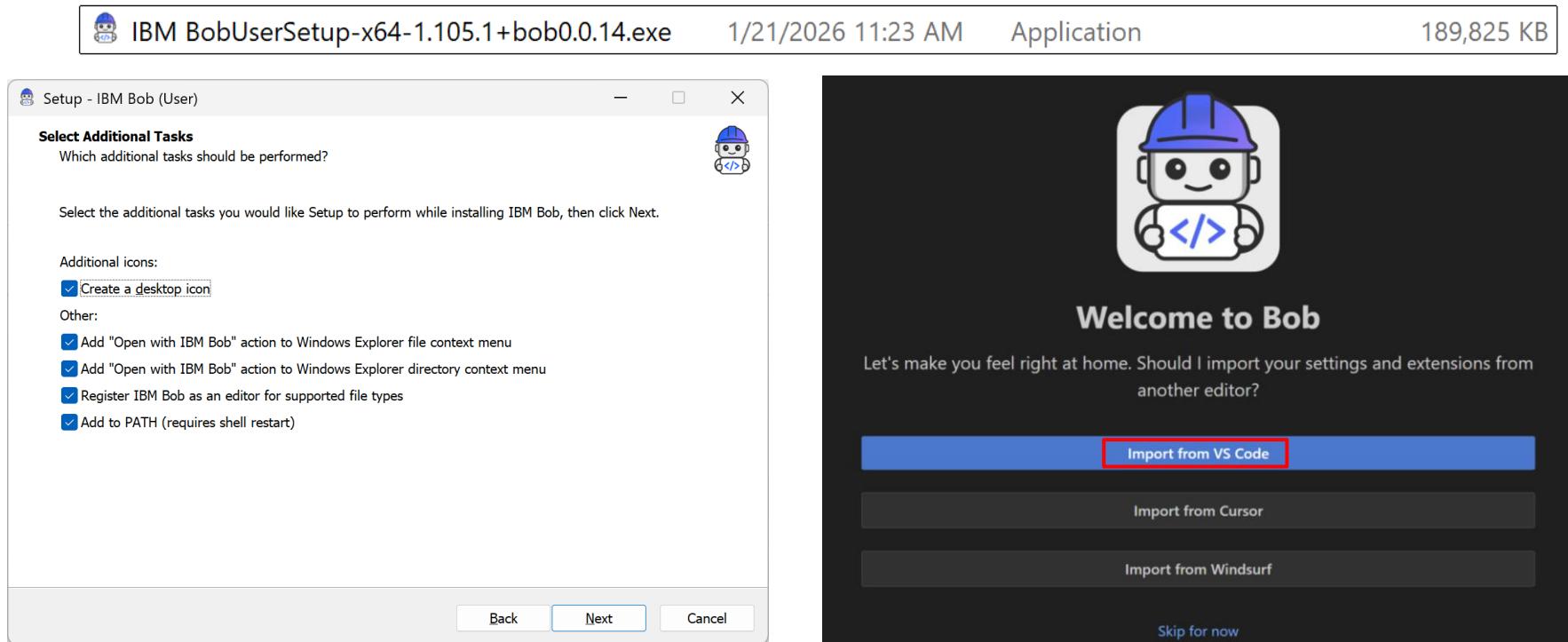
- **GA** de Bob en mode SaaS
- Fonctionnalités spécifiques IBM i dans l'IDE

- Enhanced modernization workflows for RPG, COBOL
- Enable on-system development
- Customized prompts & commands
- Exploit IBM i RAG

Bob – Installation, démarrage, configuration



- Bob est un fork de VS Code



The image shows two screenshots of the IBM Bob User Setup process. The left screenshot is the 'Select Additional Tasks' screen, showing options like creating a desktop icon and adding context menu entries. The right screenshot is the 'Welcome to Bob' screen, prompting the user to import settings and extensions from another editor, with a red box highlighting the 'Import from VS Code' button.

IBM BobUserSetup-x64-1.105.1+bob0.0.14.exe 1/21/2026 11:23 AM Application 189,825 KB

Setup - IBM Bob (User)

Select Additional Tasks

Which additional tasks should be performed?

Additional icons:

Create a desktop icon

Other:

Add "Open with IBM Bob" action to Windows Explorer file context menu

Add "Open with IBM Bob" action to Windows Explorer directory context menu

Register IBM Bob as an editor for supported file types

Add to PATH (requires shell restart)

Back Next Cancel

Welcome to Bob

Let's make you feel right at home. Should I import your settings and extensions from another editor?

Import from VS Code

Import from Cursor

Import from Windsurf

Skip for now

Bob – Installation, démarrage, configuration



Vous récupérez vos extensions spécifiques IBM i

EXPLORER
NO FOLDER OPENED

You have not yet opened a folder.
Open Folder

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

Start

- New File...
- Open File...
- Open Folder...
- Connect to...

Walkthroughs

- Get started with IBM Bob**
Customize your editor, learn the basics, and start coding
- Learn the Fundamentals**
- Get Started with ...** Updated
- Getting started w...** Updated
- Code for IBM i Ac...** Updated

Recent

IBM Bob Announcements

There are no current announcements.

Show welcome page on startup

© Copyright IBM Corp. 2025

IBM BOB OUTLINE

Hi, I'm IBM Bob

Ask me questions or let me code for you.

What would you like me to help with?

(@ to add context, / for commands, hold shift to drag in files/images)

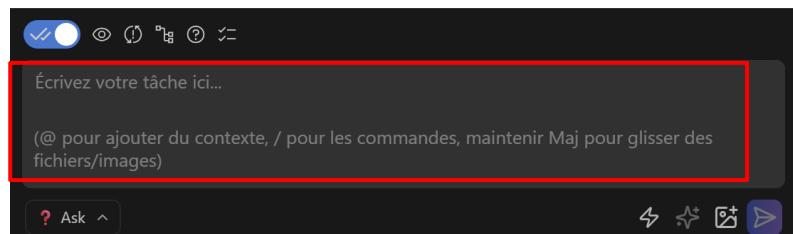
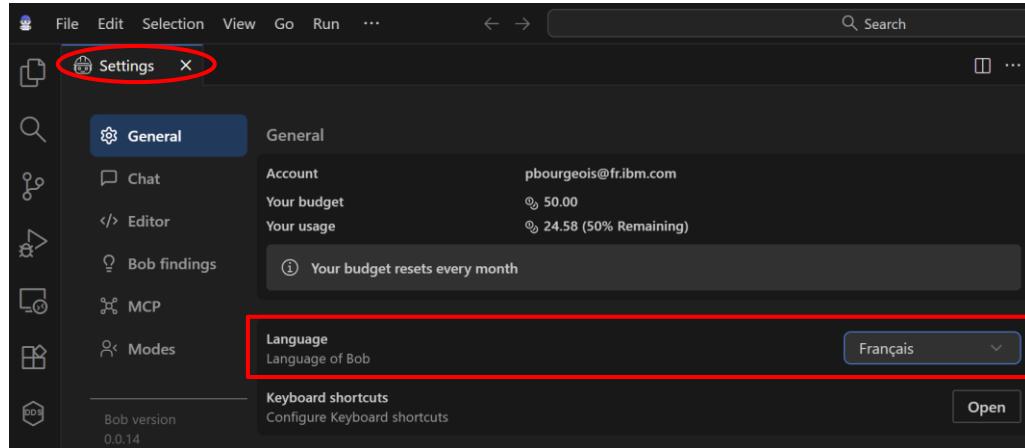
Ask

Bob - Settings

Bob – Installation, démarrage, configuration



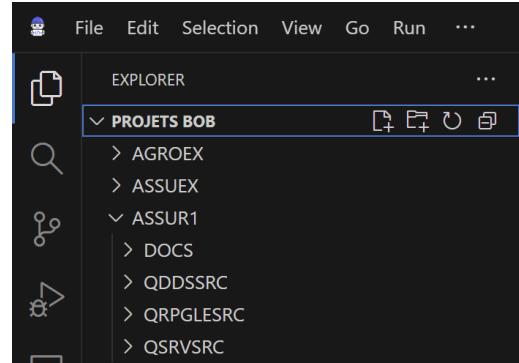
- Pour travailler en français



Bob – L'analyse des applications IBM i



- Bob, dans sa version Technology Preview, ne permet pas d'accéder directement aux systèmes de fichiers de l'IBM i
- En attendant la GA, deux solutions :
 - Solution 1** : copier les sources en local ou cloner un repository GIT localement
 - Solution 2** : installer un serveur **MCP pour IBM i**, pour pouvoir accéder directement aux systèmes de fichiers de l'IBM i
 - Non encore testé

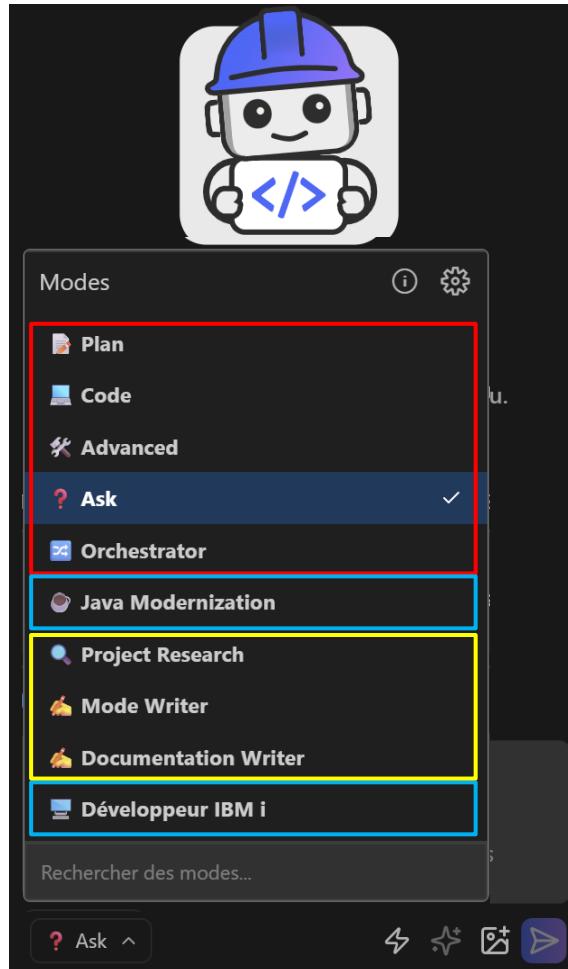


Ici, pour la démo, plusieurs projets dans le même workspace mais il est préférable d'ouvrir un seul "dossier applicatif" à la fois

Bob – Fonctionnalités principales

▪ Modes

- Bob s'adapte aux besoins spécifiques grâce à différents **modes** (*chaque mode permettant de choisir le "rôle" de Bob ainsi que les outils auxquels il aura accès (lecture de fichiers, édition de fichiers, exécution de commandes...)*)
- Il existe 5 modes prédéfinis :
 - **Ask** : explications, assistance générale
 - **Plan** : planification et conception avant implémentation
 - **Code** : génération, modification, restructuration de code
 - **Advanced** : identique à Code + accès complet aux outils
 - **Orchestrator** : pour coordonner les projets multi domaines
- Ces modes peuvent être **personnalisés**, il est possible de se **créer** ses propres modes ou d'en **télécharger** d'autres



Bob – Fonctionnalités principales



■ Personnalisation des modes

Modes

← **Code**

Description
Write and modify code

Role definition
You are IBM Bob, a highly skilled software engineer with extensive knowledge in many programming languages, frameworks, design patterns, and best practices.

When to use
Use this mode when you need to write, modify, or refactor code. Ideal for implementing features, fixing bugs, creating new files, or making code improvements across any programming language or framework. Does not support MCP or Browser tools.

Available Tools

Read Files Edit Files Execute Commands

Modes

← **Développeur IBM i**

Description
Développement, maintenance et modernisation des programmes et applications IBM i

Scope
Global

Role definition
Tu es Bob, un expert développeur IBM i (AS/400, iSeries) spécialisé dans:

- Développement en RPG (RPG II, RPG III, RPG IV, RPG Free-Format, RPG Full Free)
- Développement en COBOL (COBOL et COBOL ILE)

Pour les conversions en RPG Full Free:

- Tu indiques toujours **free en première ligne
- Tu ajoutes ctl-opt actgrp(*new) si le code contient des procédures
- Sauf si demandé explicitement, tu conserves les accès natifs aux fichiers (pas de conversion en SQL)
- Tu respectes la logique métier existante

Pour la modernisation des applications IBM i:

- Tu ne proposes pas de migrer la base de données DB2 for i vers d'autres bases de données
- Tu ne proposes pas de migrer la logique métier backend développée en RPG ou COBOL vers d'autres langages
- Tu peux proposer de migrer l'interface utilisateur 5250 vers un nouveau frontend développé dans des technologies récentes

Available Tools

Read Files Edit Files Execute Commands Use MCP Switch Modes

Bob – Fonctionnalités principales



- **Exemples de modes personnalisés**

- IBM i Developer
- IBM i DB2 for i Expert
- IBM i Dependency Analyst
- IBM i RPGUnit Tester
- IBM i DevOps Engineer
- IBM i Modernization Specialist

- Disponibles en GA ?

- Enhanced modernization workflows for RPG, COBOL
- Enable on-system development
- **Customized prompts & commands**
- Exploit IBM i RAG

Bob – Fonctionnalités principales



■ Chat – Conversations

- Pour discuter avec Bob et poser des questions

Dans IBM Bob à quoi sert le mode "Orchestrator" ?

Ask ^

⚡ ⚡+ ⚡+ ⚡

Le mode "⚡ Orchestrator" (orchestrator) dans IBM Bob est un mode spécialisé conçu pour gérer des projets complexes et multi-étapes qui nécessitent une coordination entre différentes spécialités.

⌚ Objectif Principal

Ce mode sert à **orchestrer et coordonner** des tâches complexes qui dépassent le cadre d'une simple implémentation de code ou d'une question ponctuelle.

Dans le source MAIN04.SQLRPGLE ouvert, expliquer à quoi sert la sous-routine BLOGR. Stocker l'explication dans le fichier EXPLAIN_MAIN04_BLOGR2.MD

Advanced ⚡ ⚡+ ⚡+ ⚡

```
main04.sqlrpgle x
ASSUEX > QRPGLESRC > main04.sqlrpgle > BLOGR
2179  C  BLODEC      BEGSR
2221  *  *
2222  C  ENDDO
2223  C  ENDNR
2224  ****
2225  C  BLOGR      BEGSR
2226  ****
2227  C  MOVE        *BLANKS   UTILP
2228  ? *
2229  ? * Blocage Groupement inexistant
2230  | * L'ADHERENT appartient à un GROUPE
2231  B01 C  GRPINDE   IFEQ    'G'
2232  C  GRPE        CHAIN    INFOGRP
2233  B02 C  *IN99      IFEQ    *ON
2234  C  MOVE        LIBBL    WTPREJ
2235  C  Z-ADD      25       WREJL
2236  C  MOVE        *OFF    OKLIG
2237  C  ELSE
```

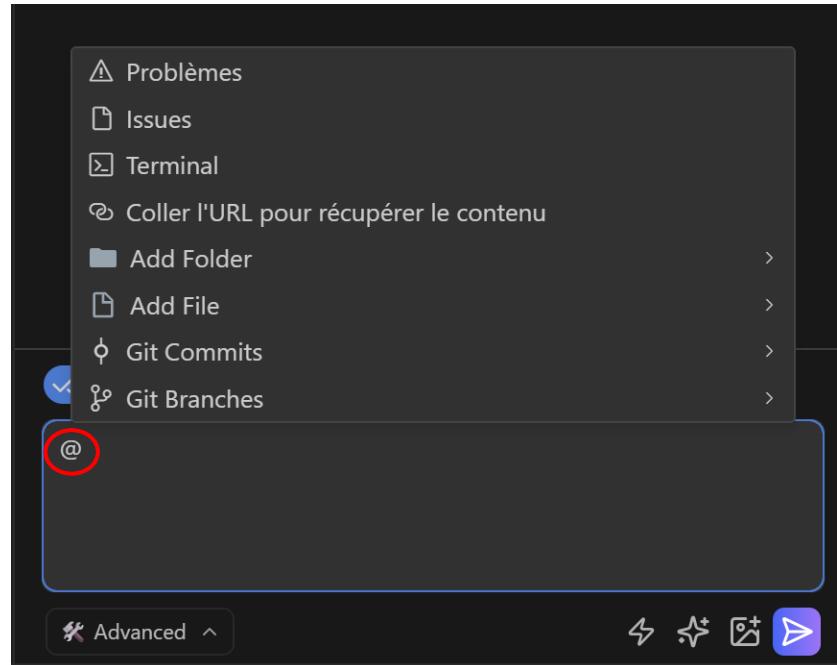
Bob – Fonctionnalités principales



■ Contexte

- Pour référencer des éléments spécifiques dans les conversations

Mention Type	Format	Description	Example Usage
File	@/path/to/file.ts	Includes file contents in request context	"Explain the function in @/src/utils.ts"
Folder	@/path/to/folder	Includes contents of all files directly in the folder (non-recursive)	"Analyze the code in @/src/components"
Problems	@problems	Includes IBM Bob Problems panel diagnostics	"@problems Fix all errors in my code"
Terminal	@terminal	Includes recent terminal command and output	"Fix the errors shown in @terminal"
Git Commit	@a1b2c3d	References specific commit by hash	"What changed in commit @a1b2c3d?"
Git Changes	@git-changes	Shows uncommitted changes	"Suggest a message for @git-changes"
URL	@https://example.com	Imports website content	"Summarize @https://docusaurus.io/!"



Bob – Fonctionnalités principales



■ Questions / réponses pour clarification

J'ai identifié plusieurs problèmes de gestion des dates dans HSR210.RPG. Avant de créer le plan détaillé, j'aimerais clarifier quelques points :

1. **Stratégie de conversion** : Préférez-vous utiliser des années sur 4 caractères (CCYYMMDD) partout, ou maintenir la compatibilité avec les années sur 2 caractères en utilisant une logique de fenêtre glissante (1940-2039) ?
2. **Impact sur les fichiers** : Les fichiers de base de données (HSLCLAMA, HSLORDHA, HSLINVTA, etc.) utilisent-ils des champs de date avec des années sur 2 ou 4 caractères ? Cela déterminera si nous devons modifier uniquement la logique du programme ou aussi les structures de données.
3. **Autres programmes** : Y a-t-il d'autres programmes RPG dans le projet (HSR200, HSR215, HSR220, etc.) qui devraient également être mis à jour avec la même logique de dates ?

Utiliser des années sur 4 caractères (CCYYMMDD) partout et modifier les structures de données si nécessaire pour une solution pérenne

Maintenir la compatibilité avec les années sur 2 caractères en utilisant une fenêtre glissante (1940-2039) avec une logique robuste de conversion

Commencer par HSR210.RPG uniquement avec une fenêtre glissante, puis évaluer les autres programmes après

Analyser d'abord les structures de fichiers DDS pour comprendre le format des dates avant de décider



Bob – Fonctionnalités principales



■ Todo List

Todo List Created

- Analyser les fichiers sources existants
- Créer le répertoire QSRVSRP pour les sources de programmes de service
- Créer le source du programme de service UTIL_SRV.RPGL avec la procédure ENQHSINFF
- Créer le source de liage UTIL_BND.BND dans QSRVSRP
- Créer le nouveau programme principal DCLSINP1.RPGL avec l'appel à la procédure
- Vérifier la cohérence de tous les fichiers créés

Todo List Created

- Analyser le programme HSR210.RPGL et les structures de fichiers DDS
- Identifier les problèmes de gestion des dates avec années sur 2 caractères
- Documenter la solution proposée avec la logique de fenêtre glissante 1940-2039
- Corriger la logique de conversion d'année dans la subroutine *INZSR (lignes 413-419)
- Vérifier et valider que toutes les utilisations de CYMD sont correctes
- Tester la logique avec des cas limites (1939, 1940, 2039, 2040)
- Documenter les changements dans les commentaires du programme
- Créer un document récapitulatif des modifications

Todo List Updated

- Analyser la structure des tables SQL et les fichiers logiques
- Créer le nouveau fichier VIDEOFFSQL.RPGL avec les structures de données
- Définir les variables indicatrices NULL (INT(5)) comme variables isolées
- Remplacer les opérations READ/READE/SETLL par des curseurs SQL
- Remplacer les opérations CHAIN par des SELECT SQL
- Adapter la gestion des valeurs NULL avec les variables indicatrices

Bob – Fonctionnalités principales



■ Approbations

- Demandes d'approbation
 - Approbations automatiques possibles par type d'opération

Maintenant, je vais créer le source de liaige (binder source) UTIL_BND.BND.

4+

IBM Bob veut créer un nouveau fichier

QSRVSRC/UTIL_BND.BND

Auto-approval disabled

Approve

Reject

✓ Approbation automatique

 Lecture

Écriture

 Navigateur

 Réessayer

 MCP  M

ode Sous-tâches

Exécuter

Question

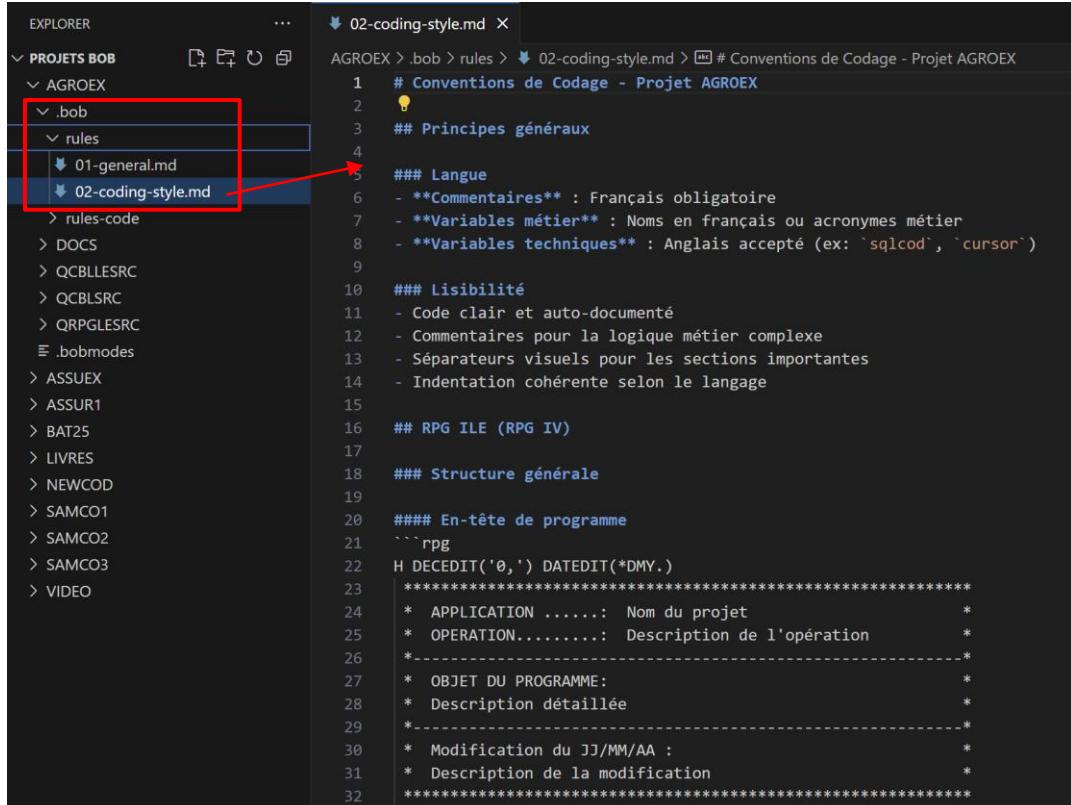
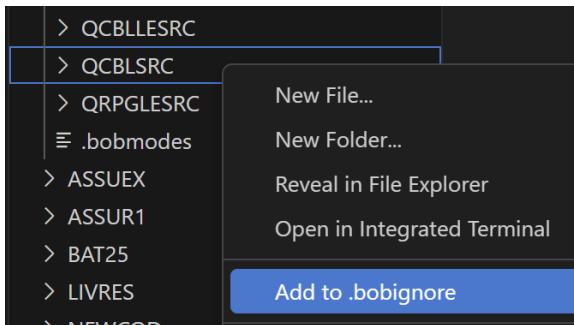
— Todo



Bob – Fonctionnalités principales



- Définition de règles
 - Globales
 - OU
 - Pour le projet en cours
- Ignorer des fichiers
 - Ajout à .bobignore



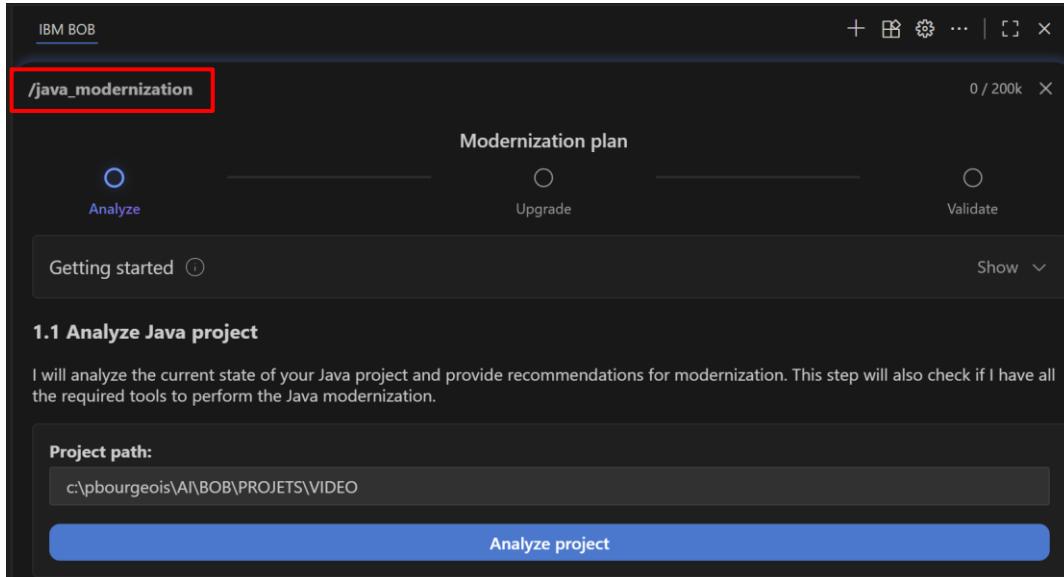
```
EXPLORER ...  
PROJETS BOB ...  
AGROEX ...  
└ .bob  
    └ rules  
        └ 01-general.md  
        └ 02-coding-style.md  
    > rules-code  
    > DOCS  
    > QCBLLSRC  
    > QCBLSRC  
    > QRPGLESRC  
    & .bobmodes  
    > ASSUEX  
    > ASSUR1  
    > BAT25  
    > LIVRES  
    > NEWCOD  
    > SAMCO1  
    > SAMCO2  
    > SAMCO3  
    > VIDEO  
02-coding-style.md x  
AGROEX > .bob > rules > 02-coding-style.md > # Conventions de Codage - Projet AGROEX  
1 # Conventions de Codage - Projet AGROEX  
2   
3 ## Principes généraux  
4  
5 ### Langue  
6 - **Commentaires** : Français obligatoire  
7 - **Variables métier** : Noms en français ou acronymes métier  
8 - **Variables techniques** : Anglais accepté (ex: `sqlcod`, `cursor`)  
9  
10 ### Llisibilité  
11 - Code clair et auto-documenté  
12 - Commentaires pour la logique métier complexe  
13 - Séparateurs visuels pour les sections importantes  
14 - Indentation cohérente selon le langage  
15  
16 ## RPG ILE (RPG IV)  
17  
18 ### Structure générale  
19  
20 ##### En-tête de programme  
21   
22 H DECEDIT('0,') DATEDIT(*DMY.)  
23 ****  
24 * APPLICATION .....: Nom du projet *  
25 * OPERATION.....: Description de l'opération *  
26 *-----*  
27 * OBJET DU PROGRAMME: *  
28 * Description détaillée *  
29 *-----*  
30 * Modification du JJ/MM/AA : *  
31 * Description de la modification *  
32 ****
```

Bob – Fonctionnalités principales



■ Workflows prédéfinis

- Actuellement : workflow "Java Modernization"
- Avec la GA : workflows "RPG/COBOL Modernization"

A screenshot of the IBM BOB (Business Object Designer) interface. The top navigation bar shows 'IBM BOB' and the URL '/java_modernization'. The main title is 'Modernization plan'. Below it are three circular buttons: 'Analyze' (selected), 'Upgrade', and 'Validate'. A 'Getting started' button is visible. The main content area is titled '1.1 Analyze Java project' with the sub-instruction: 'I will analyze the current state of your Java project and provide recommendations for modernization. This step will also check if I have all the required tools to perform the Java modernization.' A 'Project path:' field contains 'c:\pbourgeois\AI\BOB\PROJETS\VIDEO'. A large blue 'Analyze project' button is at the bottom.

/java_modernization 0 / 200k X

Modernization plan

Analyze Upgrade Validate

Getting started

1.1 Analyze Java project

I will analyze the current state of your Java project and provide recommendations for modernization. This step will also check if I have all the required tools to perform the Java modernization.

Project path:

c:\pbourgeois\AI\BOB\PROJETS\VIDEO

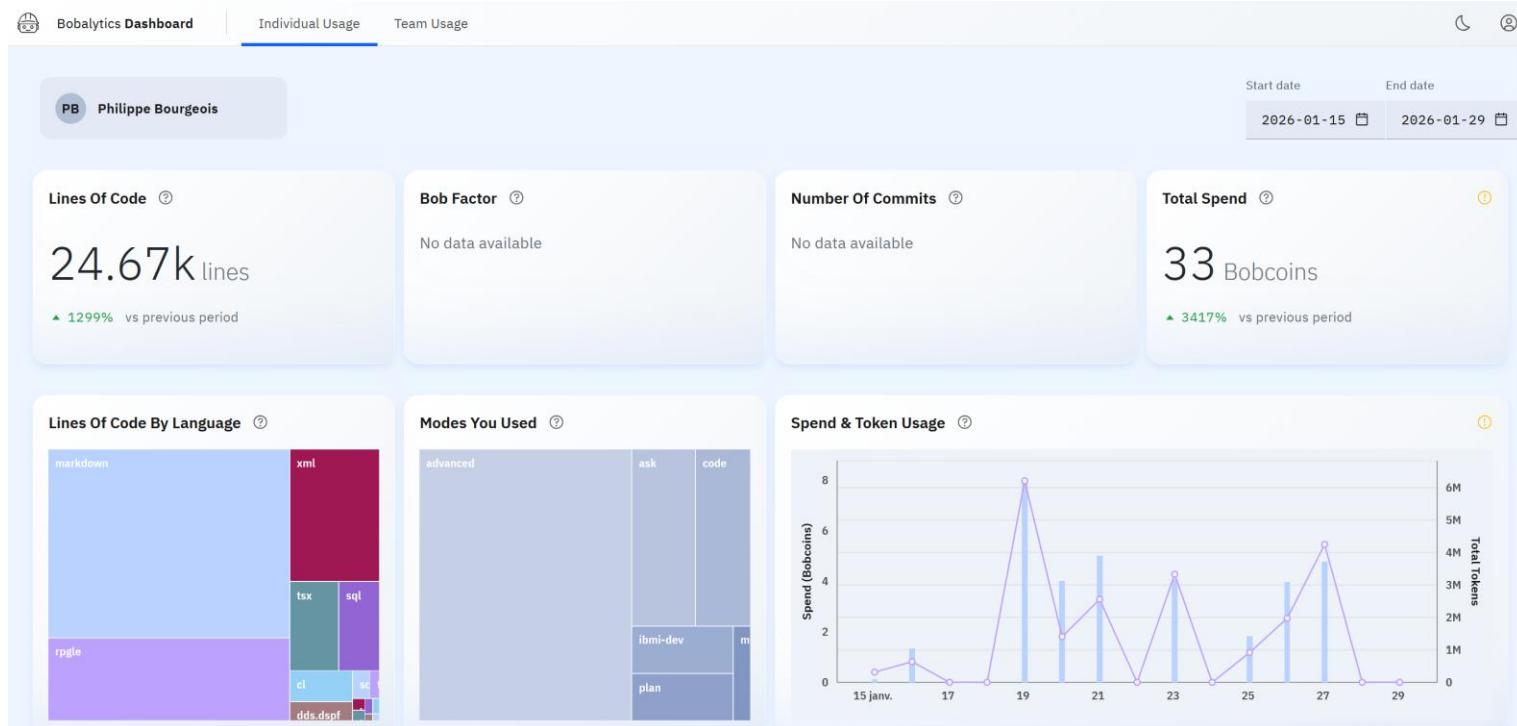
Analyze project

- Enhanced modernization workflows for RPG, COBOL
- Enable on-system development
- Customized prompts & commands
- Exploit IBM i RAG

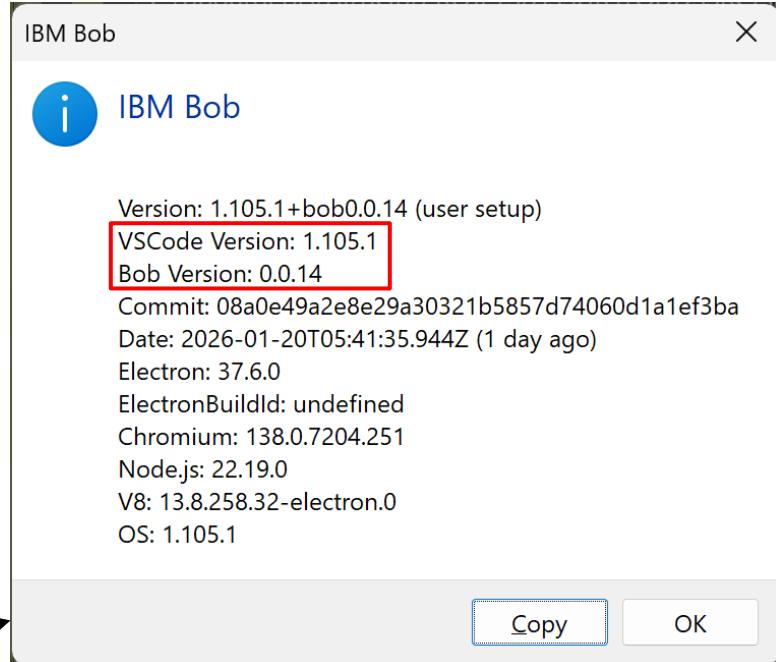
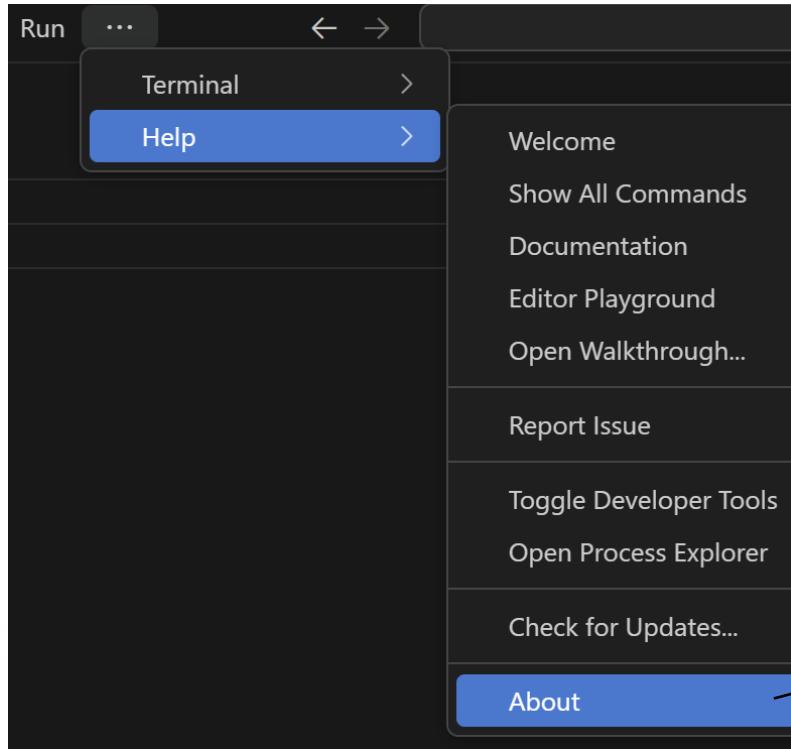
Bob – Fonctionnalités principales



■ Analytique : Bobalytics Dashboard



Version de Bob utilisée pour cette présentation





B. Modernisation des applications IBM i avec IBM Bob

Modernisation applicative IBM i avec Bob

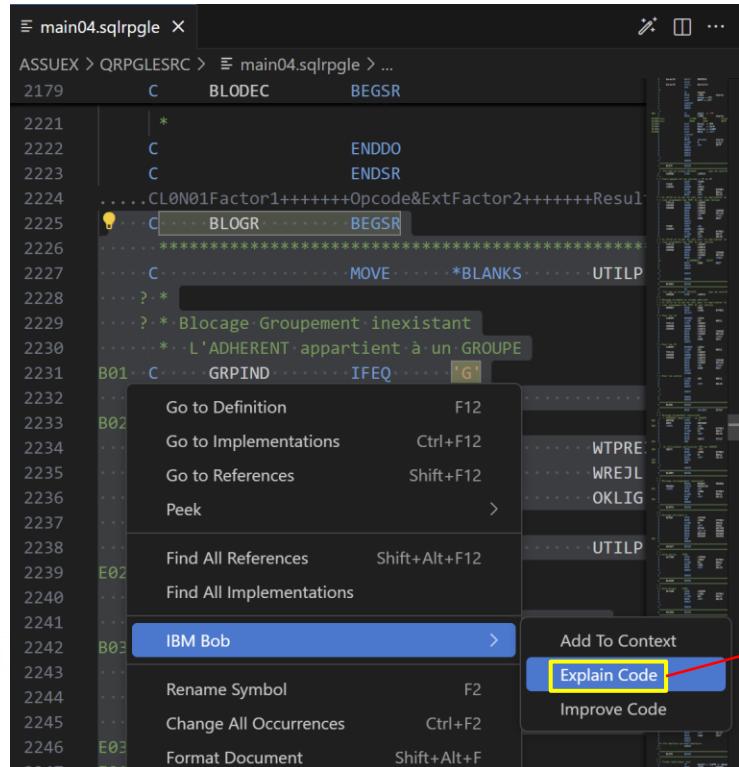


- 1. Compréhension, explication et documentation de code
- 2. Amélioration, optimisation de code
- 3. Restructuration, transformation, conversion de code



1. Compréhension, explication et documentation de code

Explication d'une partie de programme



```
ASSUEX > QRPGLESRC > main04.sqlrpgle > main04.sqlrpgle
2179 C BLODEC BEGSR
2221 * C ENDDO
2222 C ENDSR
2223 ....CL0N01Factor1+++++0pcode&ExtFactor2+++++Result
2224 ....CL0N01Factor1+++++0pcode&ExtFactor2+++++Result
2225 C BLOGR BEGSR
2226 ****MOVE *BLANKS UTILP
2227 C
2228 ? *
2229 ? * Blocage·Groupement·inexistant
2230 * L'ADHERENT·appartient·à·un·GROUPE
2231 C GRPIND IFEQ 'G'
2232 B01 Go to Definition F12
2233 B02 Go to Implementations Ctrl+F12
2234 B03 Go to References Shift+F12
2235 B04 Peek >
2236 B05 Find All References Shift+Alt+F12
2237 B06 Find All Implementations
2238 B07
2239 E02
2240 E03
2241 B08 IBM Bob >
2242 B09 Add To Context
2243 B10 Explain Code
2244 B11 Improve Code
2245 B12 Rename Symbol F2
2246 B13 Change All Occurrences Ctrl+F2
2247 B14 Format Document Shift+Alt+F
```

IBM BOB OUTLINE

Explain the following code from file path QRPGLESRC\main04.sqlrpgle:2225-2249 *** C BLOGR BEGSR ***... 18k / 200k 0.12 X

Explication du code BLOGR (lignes 2225-2249)

1. Objectif et Fonctionnalité

La sous-routine BLOGR (BLOcage GRouement) est un contrôle de validation qui vérifie la validité d'un groupement d'adhérents et les droits d'accès associés. Elle effectue deux vérifications principales :

1. Vérification de l'existence du groupement dans la base de données
2. Contrôle des droits d'accès pour les utilisateurs ayant un statut particulier ('SPE')

2. Composants Clés et Leurs Interactions

Variables principales :

- GRPIND : Indicateur de type de groupement (valeur 'G' = Groupe)
- GRPE : Clé du groupement à vérifier
- INFOGRP : Fichier contenant les informations des groupements

Principe de sécurité :

Application du principe de **moindre privilège** : même si le groupement existe, un utilisateur avec statut 'SPE' est bloqué, nécessitant des droits supplémentaires pour accéder aux données de groupe.

Explication d'une partie de programme



```
main04.sqlrpgle x
ASSUEX > QRPGLESRC > main04.sqlrpgle > BLOGR
2179      C     BL0DEC      BEGSR
2221      *          END00
2222      C          END0R
2223      ****
2224      C     BLOGR      BEGSR
2225      ****
2226      C          MOVE      *BLANKS      UTILP
2227      ? *
2228      ? * Blocage Groupement inexistant
2229      ? * L'ADHERENT appartient à un GROUPE
2230      B01 C     GRPIND      IFEQ      'G'
2231      C     GRPE      CHAIN      INFOGRP
2232      B02 C     *IN99      IFEQ      *ON
2233      C          MOVE      LIBBL      WTPREJ
2234      C          Z-APP      25      WTPREJ
2235      ****
2236
2237 Dans le source MAIN04.SQLRPGLE ouvert, expliquer à
2238 quoi sert la sous-routine BLOGR. Stocker l'explication
2239 dans le fichier EXPLAIN_MAIN04_BLOGR2.MD
2240
2241
2242
2243
2244
2245      C          MOVE      *OFF      OKLIG
2246      E03 C          ENDIF
2247      E01 C          END
2248      *
2249      C          END0R
2250      ****
```

Preview EXPLAIN_MAIN04_BLOGR2.MD x

Explication de la sous-routine BLOGR - MAIN04.SQLRPGLE

Vue d'ensemble

La sous-routine **BLOGR** (lignes 2225-2249) est responsable de la **validation de l'existence et de l'accès d'un groupement** pour un adhérent. Elle effectue des contrôles de blocage liés au groupement auquel appartient l'adhérent.

Objectif principal

Cette sous-routine vérifie deux conditions critiques qui peuvent bloquer le traitement d'une ligne :

1. **L'existence du groupement** dans la table **INFOGRP**
2. **Le type d'utilisateur** associé au groupement (utilisateur particulier "SPE")

Logique détaillée

1. **Initialisation (ligne 2227)**

```
MOVE *BLANKS UTILP
```

 - Initialise la variable **UTILP** (type d'utilisateur) à blanc
2. **Vérification si l'adhérent appartient à un groupe (lignes 2231-2239)**



Explication de requêtes SQL

Go Run ... ← → 🔍 Projets BOB (Workspace)

main01.sqlrpgle X

```
ASSUEX > QRPGLESRC > main01.sqlrpgle > getActeFromSubstitution
367 Dcl-Proc getListCodeLPOOptique EXPORT;
395     Select
396         | PARDCLPP.CDLPP
397     From PARDCLPP
398     Where PARDCLPP.PANIER = :PANIER_p
399         And 1 = Case
400             When :SPHERE =
401                 And (PARDCLPP.SPHERE1 IS NULL OR
402                     :SPHERE Between PARDCLPP.SPHERE1 And
403                         PARDCLP
404                     And (PARDCLPP.CYLINDRE1 IS N
405                         :CYLINDRE Between PARDCLPP.CYLINDRE1 And
406                             PARDCLP
407
```

Dans le source MAIN01.SQLRPGLE ouvert dans l'espace de travail, expliquer à quoi sert la requête SQL entre les lignes 395 et 445

Advanced ^



Ln 61, Col 18 (420 selected) Spaces: 2 UTF-8 CRLF {} RPGLE Bob - Settings

Explication de requêtes SQL

Scénario 3 : Sphère = 0 ET Cylindre ≠ 0 (lignes 415-420)

```
When :SPHERE = 0 And :CYLINDRE <> 0  
And (PARDCLPP  
:SPHERE  
And :CYLINDRE
```

Usage : Correction cylindrique uniquement (astigmatisme pur).

Explication de la requête SQL (lignes 395-445) - MAIN01.SQLRPGLE

Vue d'ensemble

Cette requête SQL sélectionne un code LPP (Liste des Produits et Prestations) depuis la table `PARDCLPP` en fonction de paramètres optiques complexes pour des verres correcteurs.

Objectif

La requête détermine le code de remboursement approprié pour des verres optiques en fonction de :

- La **sphère** (correction de la myopie/hypermétropie)
- Le **cylindre** (correction de l'astigmatisme)
- Le **type de verre**
- Le **panier** (catégorie de remboursement)

Résultat

La requête retourne une liste de codes LPP correspondant aux critères de prescription optique, permettant ainsi de déterminer le remboursement applicable selon la réglementation de la Sécurité Sociale française.

Explication de programmes RPG



Expliquer à quoi sert le programme CLI01_sans_cartouche.SQLRPGLE ouvert dans l'espace de travail.
Stocker l'explication dans le fichier EXPLAIN_CLI01_sans_cartouche_V2.MD

Task Completed

Advanced ^

Explication complète du programme CLI01_sans_cartouche.SQLRPGLE créée avec succès dans le fichier EXPLAIN_CLI01_sans_cartouche_V2.MD.

Le document détaille :

- Vue d'ensemble et objectif du programme
- Architecture technique (fichiers, paramètres)
- Flux de traitement complet
- Requête SQL principale
- Système de gestion des couleurs par priorité
- Options et touches de fonction disponibles
- Contrôles et validations métier
- Liste complète des programmes appelés
- Gestion de la LDA
- Particularités techniques
- Messages d'erreur
- Points d'attention importants

Explication de programmes RPG



File Edit Selection View Go Run ... ← → Projets BOB (Workspace) 0: Preview EXPLAIN_CLI01_sans_cartouche_V2.MD X

ExPLICATION du Programme CLI01_sans_cartouche.SQLRPGLE

Vue d'ensemble

CLI01_sans_cartouche.SQLRPGLE est un programme RPG ILE (Integrated Language Environment) qui gère l'affichage et la manipulation des rôles contractuels d'une personne physique dans un système de gestion d'assurance. Il s'agit d'un programme interactif avec sous-fichier qui permet de consulter et gérer les contrats associés à une personne.

Objectif principal

Le programme affiche une liste des contrats (rôles contractuels) associés à une personne identifiée par son `IdPersonne`. Il :

- Visualiser les contrats d'une personne
- Modifier des contrats existants
- Créer de nouveaux contrats
- Supprimer/radier des contrats
- Créer des contrats de surcomplémentaire
- Accéder à diverses fonctions de gestion (adresses, téléphones, emails, etc.)

Architecture technique

Fichiers utilisés

Fichiers d'affichage

- CLI0036E** : Fichier écran principal avec sous-fichier (FSFL)

Fichiers de base de données

Contrôles et validations

Contrôle de blocage sinistre (SRCtlSin)

Avant toute modification (F08, F10, F11, F13, F15, F17), le programme vérifie via **MODP001G** s'il existe des sinistres en cours. La modification est bloquée avec le message :

"MAJ impossible, sinistre en cours de règlement"

Contrôle surcomplémentaire (SRCtlSurco)

Lors de la création d'un contrat surco (option 6), le programme :

- Récupère les informations du contrat de base via **ADHG003G**
- Vérifie l'existence de surcomplémentaires disponibles via **MOD204G**
- Bloque l'opération si aucune surco n'est disponible pour le barème

Vérification du statut de la personne (SREtatPers)

Le programme vérifie via **CLIG032G** si la personne est un souscripteur actif :

- 1** : Actif → Accès à F20 (compte adhérent)
- 0** : Inactif → Pas d'accès F20
- 2** : Non identifié → Pas d'accès F20

Explication de programmes RPG



Spécification Fonctionnelle - CLI01.SQLRPGLE

1. IDENTIFICATION DU PROGRAMME

Élément	Valeur
Nom du programme	CLI0036G
Type	SQLRPGLE
Application	CLIENT - Référentiel Client
Fonction principale	Fiche personne physique
Auteur initial	PBL
Date de création	11/04/2019
Version actuelle	00.04.47
Dernière modification	10/12/2021 (SBO)

2. OBJECTIF FONCTIONNEL

Ce programme constitue l'écran central de gestion d'une personne physique dans le référentiel client. Il permet de :

- Visualiser les informations synthétiques d'une personne
- Accéder aux différentes fiches de détail (adresses, téléphones, emails, etc.)
- Gérer les rôles contractuels de la personne (souscripteur, bénéficiaire, etc.)
- Consulter l'historique des contacts et réclamations
- Gérer le compte espace adhérent

Spécification Technique - CLI01.SQLRPGLE

1. Vue d'ensemble

1.1 Identification

- **Programme** : CLI0036G (CLI01.SQLRPGLE)
- **Type** : SQLRPGLE
- **Application** : CLIENT - Référentiel Client
- **Fonction** : Fiche personne physique avec gestion des rôles contrat

1.2 Description

Programme de gestion et d'affichage des informations d'une personne physique dans le référentiel client. Il permet de visualiser et gérer les différents rôles contractuels d'une personne (souscripteur, bénéficiaire, etc.) ainsi que toutes ses coordonnées associées.

2. Architecture Technique

2.1 Paramètres d'entrée (PEntrée)

```
Dcl-Ds wEntree Qualified;
  OptExc  char(60);      // Opérations exclues
  Cop     zoned(2:0);    // Code opération
  LOp    char(20);      // Libellé opération
  IdPers  zoned(9:0);    // Id personne
End-Ds;
```

Explication de programmes RPG



Fonctionnalité principale :

- Crée, modifie et supprime les fiches adhérents pour plusieurs gammes d'assurance (Santé, Prévoyance, Obsèques, ...)
- Gère le traitement bi-annuel (année en cours + année précédente simultanément)
- Permet la duplication de fiches (simple ou jumelée)
- Calcule les cotisations et gère les tarifs
- Gère les informations bancaires pour prélèvement automatique (SEPA/RIB)
- Suit les bénéficiaires et les commissions

La subroutine `CONTRO` effectue plus de 50 validations :

- Cohérence dates (prélèvement, inscription, radiation)
- Situation familiale vs nombre de bénéficiaires
- Compatibilité garantie/périodicité/mode de paiement
- Règles fiscales (Madelin, CMU, crédit d'impôt)
- Contrôles spécifiques par association (ASAFA, AFPS, MS)

10. Spécificités métier

- **Contrats responsables/non-responsables** : Gestion transition
- **IJH** (Indemnités Journalières Hospitalisation) : Base, montant, c...
- **Éclatement de contrat** (`WECLAT='E'`) : Division d'un contrat g...
- **Carte verte/blanche** : Gestion des cartes de tiers payant
- **CMU** (Couverture Maladie Universelle) : Traitement spécifique a...
- **RGPD/CNIL** : Consentements marketing via `ADHF116`

Explication de programmes RPG



Task Completed



Documentation créée avec succès dans `EXPLAIN_STOCKS.MD`

Le fichier contient une explication complète du programme STOCKS.SQLRPGLE, incluant :

- **Vue d'ensemble** : Programme de calcul de statistiques de stock de veaux en élevage
- **Objectif** : Génération de fiches de stock avec mouvements (entrées, sorties, mortalités) et valorisation
- **Paramètres** : Date début, date fin, type de veau (laitier/autre)
- **Architecture détaillée** :
 - 5 requêtes SQL principales (stock début, entrées, mortalités, sorties)
 - Calcul du stock de fin par différence
 - Utilisation de UNION pour chercher les prix dans deux tables
- **Tables SQL** : OELOVEAU, OESTOKVO, OEACHCOL, OELIVLOL
- **Modifications historiques** : Filtres sur lots ≥ 25388 (2017), inclusion prix à 0
- **Structure des données** : 16 champs calculés par lot (quantités et valeurs)
- **Points techniques** : SQL embarqué, curseurs, conversions de dates, traçabilité
- **Recommandations** : Exécution batch, vérifications, monitoring performance

Explication du Programme STOCKS.SQLRPGLE

Vue d'ensemble

STOCKS.SQLRPGLE est un programme RPG ILE avec SQL intégré conçu pour **calculer et alimenter un fichier de travail avec les statistiques de stock de veaux en élevage** sur une période donnée. Il fait partie du projet SOEINT et prépare les données pour les états I1100 et I1120.

Objectif principal

Le programme génère des fiches de stock détaillées pour les veaux en élevage, en calculant pour chaque lot :

- Le stock de début de période (quantité et valeur)
- Les entrées pendant la période
- Les mortalités pendant la période
- Les sorties (abattages) pendant la période
- Le stock de fin de période (calculé par différence)

Explication de programmes CBL



Task Completed

Explication complète du programme PESEES01.CBL créée dans le fichier EXPLAIN

Le document détaille :

- L'objectif du programme (calcul des poids et taux pour l'atelier R.P.C.)
- Les 3 modes de traitement (tank, journée, période)
- Les calculs détaillés pour chaque composant (EST, MG, ESD, MP)
- La gestion des unités de mesure (G/KG et G/L) avec conversions
- Les règles spéciales (densité fixe du lait cru à 1,033)
- L'historique complet des modifications (1995-2019)
- La validation des analyses et gestion des mouvements
- Les aspects techniques (précision, erreurs, audit trail)

Preview EXPLAIN_PESEE01.MD X

Explication du Programme PESEES01.CBL

Vue d'ensemble

Programme: PESEES01.CBL **Auteur:** xxx **Dernière modification:** 17/07/1996
Langage: COBOL pour IBM AS/400
Contexte: Atelier R.P.C. (Réception et Pesée du Lait Cru)

Objectif principal

Ce programme effectue le **calcul automatique des différents poids et taux** des matières premières (principalement du lait) en fonction des analyses physico-chimiques. Il s'agit d'un programme de traitement batch qui valorise les enregistrements d'apports en calculant les poids dérivés à partir des analyses de laboratoire.

Fonctionnalités principales

1. Calcul des poids selon les analyses

Le programme calcule automatiquement :

- **Poids d'EST (Extrait Sec Total)** : Matière sèche totale
- **Poids de MG (Matière Grasse)** : Teneur en lipides
- **Poids d'ESD (Extrait Sec Dégraissé)** : Matière sèche sans la matière grasse
- **Poids de MP (Matière Protéique)** : Teneur en protéines

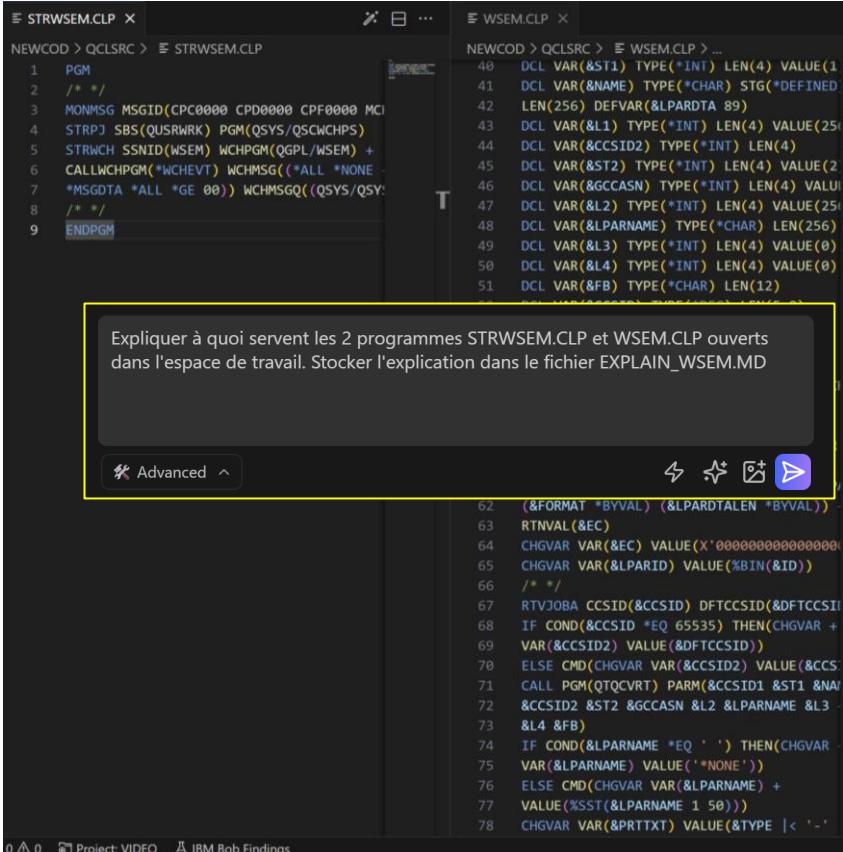
2. Gestion des unités de mesure

Le programme gère deux systèmes d'unités :

- **G/KG** (grammes par kilogramme) : Concentration massique
- **G/L** (grammes par litre) : Concentration volumique

Il effectue automatiquement les conversions nécessaires entre poids et volume en utilisant la densité.

Explication de programmes CL



```
STRWSEM.CLP
NEWCOD > QCLSRC > STRWSEM.CLP
1  PGM
2  /* */
3  MONMSG MSGID(CPC0000 CPD0000 CPF0000 MCI
4  STRPJ SBS(QUSRWRK) PGM(QSYS/QSCNCHPS)
5  STRWCH SSNID(WSEM) NCHPGM(QGPL/WSEM) +
6  CALLWCHPGM(*NCHVT) WCHMSG(*ALL *NONE +
7  *MSGDTA *ALL *GE 00) WCHMSGQ(QSYS/QSY
8  /* */
9  ENDPGM

WSEM.CLP
NEWCOD > QCLSRC > WSEM.CLP > ...
40  DCL VAR(&ST1) TYPE(*INT) LEN(4) VALUE(1
41  DCL VAR(&NAME) TYPE(*CHAR) STG(*DEFINED
42  LEN(256) DEFVAR(&LPARDTA 89)
43  DCL VAR(&L1) TYPE(*INT) LEN(4) VALUE(256
44  DCL VAR(&CCSID2) TYPE(*INT) LEN(4)
45  DCL VAR(&ST2) TYPE(*INT) LEN(4) VALUE(2
46  DCL VAR(&GCCASN) TYPE(*INT) LEN(4) VALUE(1
47  DCL VAR(&L2) TYPE(*INT) LEN(4) VALUE(256
48  DCL VAR(&LPARNAME) TYPE(*CHAR) LEN(256
49  DCL VAR(&L3) TYPE(*INT) LEN(4) VALUE(0
50  DCL VAR(&L4) TYPE(*INT) LEN(4) VALUE(0
51  DCL VAR(&FB) TYPE(*CHAR) LEN(12
52  FMTOUT FMT(*VARY) FMTVAL(*VARY)
53  /* */
54  /* */
55  /* */
56  /* */
57  /* */
58  /* */
59  /* */
60  /* */
61  /* */
62  (&FORMAT *BYVAL) (&LPARDTALEN *BYVAL)
63  RTNVAL(&EC)
64  CHGVAR VAR(&EC) VALUE(X'0000000000000000)
65  CHGVAR VAR(&LPARID) VALUE(X'BIN/&ID)
66  /* */
67  RTWJOBA CCSID(&CCSID) DFTCSSID(&DFTCSSID)
68  IF COND(&CCSID *EQ 65535) THEN(CHGVAR +
69  VAR(&CCSID2) VALUE(&DFTCSSID))
70  ELSE CMD(CHGVAR VAR(&CCSID2) VALUE(&CCS
71  CALL PGM(QTQCVRT) PARM(&CCSID1 &ST1 &NA
72  &CCSID2 &ST2 &GCCASN &L2 &LPARNAME &L3
73  &L4 &FB)
74  IF COND(&LPARNAME *EQ ' ') THEN(CHGVAR
75  VAR(&LPARNAME) VALUE('*NONE'))
76  ELSE CMD(CHGVAR VAR(&LPARNAME) +
77  VALUE(%SST(&LPARNAME 1 50)))
78  CHGVAR VAR(&PRTXT) VALUE(&TYPE |< '-'
```

Expliquer à quoi servent les 2 programmes STRWSEM.CLP et WSEM.CLP ouverts dans l'espace de travail. Stocker l'explication dans le fichier EXPLAIN_WSEM.MD

Advanced ▾

Explication des Programmes STRWSEM.CLP et WSEM.CLP

Vue d'ensemble

Ces deux programmes CL forment un système de surveillance et d'alerte pour les messages système IBM i/AS400. Ils travaillent ensemble pour :

- STRWSEM.CLP** : Démarrer la surveillance des messages système
- WSEM.CLP** : Traiter les messages détectés et envoyer des alertes par email

Programme 1 : STRWSEM.CLP (Démarrage de la surveillance)

Objectif

Programme d'initialisation qui configure et démarre un observateur de messages système (Watch).

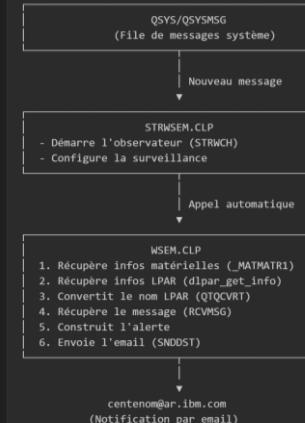
Flux de traitement

1. Surveillance globale des erreurs (ligne 3)

MONMSG surveille tous les types d'erreurs :

- CPC0000** : Messages de compléition
- CPD0000** : Messages de diagnostic

Architecture du système



Documentation de programmes



WSEM.CLP X

NEWCOD > QCLSRC > WSEM.CLP > [] &EVTDATA

```
1 PGM PARM(&WCHOPTION &SESSIONID &ERROR &EVTDATA)
2 /* */
3 DCL VAR(&WCHOPTION) TYPE(*CHAR) LEN(10)
4 DCL VAR(&SESSIONID) TYPE(*CHAR) LEN(10)
5 DCL VAR(&ERROR) TYPE(*CHAR) LEN(10)
6 DCL VAR(&EVTDATA) TYPE(*CHAR) LEN(1024)
```

Mettre des commentaires dans le programme WSEM.CLP ouvert
le sauvegarder en WSEM2.CLP

Task Completed



Le fichier WSEM2.CLP a été créé avec succès avec des commentaires détaillés en français.

Les commentaires ajoutés incluent :

- En-tête du programme avec description et paramètres
- Documentation de chaque section (déclarations, variables, traitement)
- Explication de chaque variable avec son rôle
- Commentaires sur les étapes du traitement principal
- Description des appels API et conversions de jeu de caractères
- Clarification du format du message d'alerte

Le programme commente maintenant clairement sa fonction : récupérer les informations système (matériel et partition logique), formater un message d'alerte avec ces informations et l'envoyer par email.

NEWCOD > QCLSRC > WSEM2.CLP > ...

```
1 PGM PARM(&WCHOPTION &SESSIONID &ERROR &EVTDATA)
2
3 /* ===== */
4 /* Programme : WSEM2.CLP
5 /* Description : Programme de gestion et d'envoi d'alertes par email
6 /* pour les messages système IBM i
7 /* Paramètres :
8 /* &WCHOPTION - Option de traitement (10 caractères)
9 /* &SESSIONID - Identifiant de session (10 caractères)
10 /* &ERROR - Code d'erreur (10 caractères)
11 /* &EVTDATA - Données d'événement (1024 caractères)
12 /* ===== */
13
14 /* -----
15 /* SECTION 1 : Déclaration des variables principales
16 /*
17 DCL VAR(&WCHOPTION) TYPE(*CHAR) LEN(10) /* Option de traitement */
18 DCL VAR(&SESSIONID) TYPE(*CHAR) LEN(10) /* ID de session */
19 DCL VAR(&ERROR) TYPE(*CHAR) LEN(10) /* Code d'erreur */
20 DCL VAR(&EVTDATA) TYPE(*CHAR) LEN(1024) /* Données d'événement */
21 DCL VAR(&MSG) TYPE(*CHAR) LEN(173) /* Texte du message */
22 DCL VAR(&ALERT) TYPE(*CHAR) LEN(255) /* Message d'alerte formaté */
23
24 /*
25 /* SECTION 2 : Variables définies pour extraire les données d'événement */
26 /* Ces variables pointent vers des positions spécifiques dans &EVTDATA */
27 /*
28 DCL VAR(&MSGID) TYPE(*CHAR) STG(*DEFINED) LEN(7) +
29 | DEFVAR(&EVTDATA 5) /* ID du message (position 5) */
30 DCL VAR(&MSGQ) TYPE(*CHAR) STG(*DEFINED) LEN(10) +
31 | DEFVAR(&EVTDATA 13) /* Nom de la file de messages (position 13) */
32 DCL VAR(&MSGQLIB) TYPE(*CHAR) STG(*DEFINED) +
33 | LEN(10) DEFVAR(&EVTDATA 23) /* Bibliothèque de la file de messages (position 23) */
34 DCL VAR(&MSGKEY) TYPE(*CHAR) STG(*DEFINED) +
35 | LEN(4) DEFVAR(&EVTDATA 387) /* Clé du message (position 387) */
36
37 /*
38 /* SECTION 3 : Variables pour les attributs matériels du système
39 /*
40
```

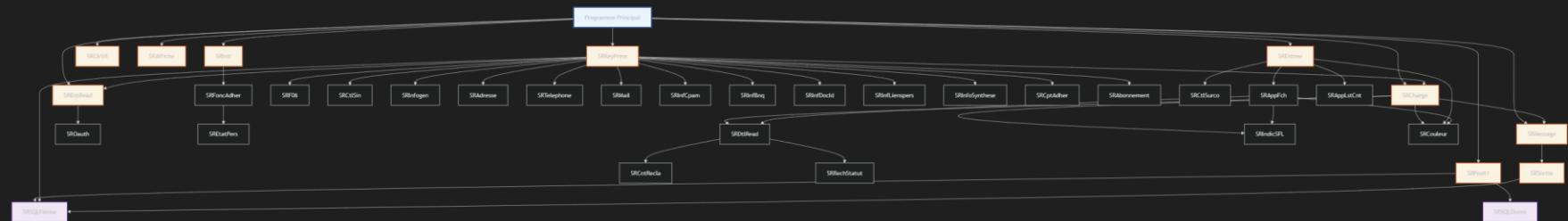
Diagramme des appels *internes* d'un programme



A partir du programme CLI01.SQLRPGLE ouvert dans l'espace de travail, générer un diagramme représentant uniquement les appels internes aux sous-routines ou aux sous-procédures.

Diagramme des appels internes aux sous-routines - CLI01.SQLRPGLE

Ce diagramme représente uniquement les appels internes aux sous-routines (EXSR) du programme CLI01.SQLRPGLE.



Liste des sous-routines identifiées

Sous-routines principales (appelées depuis le programme principal)

1. **SRInit** (ligne 1001) - Initialisation du programme
 2. **SREntRead** (ligne 1080) - Lecture des données d'entête
 3. **SRClrSfl** (ligne 1266) - Mise à blanc du sous-fichier
 4. **SRPosit1** (ligne 1282) - Premier positionnement du fichier

Diagramme des programmes *appelés* par un programme



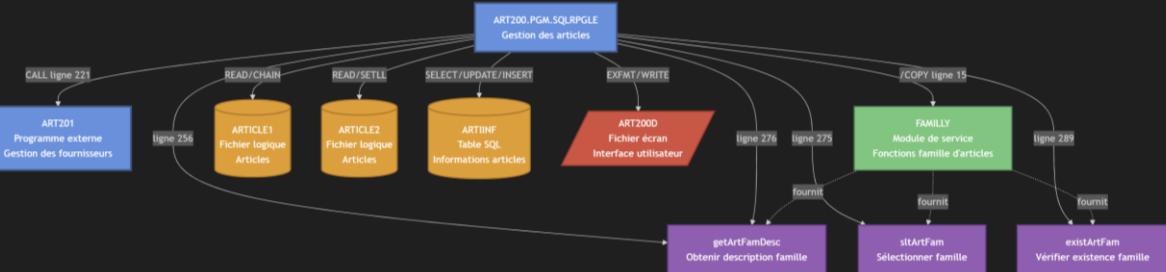
Afficher un diagramme représentant graphiquement tous les programmes qui sont appelés par le programme ART200.PGM.RPGLE ouvert dans l'espace de travail. Stocker le résultat dans un fichier ART200_PGM_CALLED.MD

Diagramme des appels du programme ART200.PGM.SQLRPGLE

Vue d'ensemble

Ce document présente graphiquement tous les programmes et modules appelés par le programme `ART200.PGM.SQLRPGLE`.

Diagramme des appels



Détails des appels

1. Programme externe

- **ART201** (ligne 17-19, 221)
 - Déclaré comme programme externe via `extpgm('ART201')`
 - Appelé avec le paramètre `ar_id` (identifiant article)
 - Fonction : Gestion des fournisseurs d'un article

Quels sont les programmes qui appellent une procédure ?



```
▽ SAMCO3
  ▽ QRPGLESRC
    □ ART200.PGM.SQLRPGLE
    □ ART201.PGM.RPGL
    □ ART202.PGM.RPGL
    □ ART250.PGM.SQLRPGLE
    □ ART300.RPGL
    □ ART301.SQLRPGLE
    □ ART302.SQLRPGLE
    □ COU300.RPGL
    □ COU301.RPGL
    □ CUS200.PGM.SQLRPGLE
    □ CUS250.PGM.RPGL
    □ CUS300.RPGL
    □ CUS301.SQLRPGLE
    □ DAT001.PGM.RPGL
    □ DAT002.PGM.RPGL
    □ FAM300.RPGL
    □ FAM301.RPGL
    □ LOG100.PGM.RPGL
    □ LOG300.RPGL
    □ ORD100.PGM.RPGL
    □ ORD101.PGM.RPGL
    □ ORD200.PGM.SQLRPGLE
    □ ORD201.PGM.SQLRPGLE
    □ ORD202.PGM.RPGL
    □ ORD500.PGM.RPGL
    □ ORD700.PGM.RPGL
    □ ORD900.PGM.RPGL
    □ ORD901.PGM.SQLRPGLE
```

Quels sont les programme qui appellent la procédure GetVATRate dans SAMCO3/QRPGLESRC ?

Task Completed

Programmes appelant la procédure GetVATRate dans SAMCO3/QRPGLESRC

La procédure `GetVATRate` est définie dans le module de service `VAT300.RPGL` et est appelée par 2 programmes :

1. `ORD100.PGM.RPGL` - ligne 269

```
vatRate = GetVatRate(GetArtVatCode(odorid));
```

2. `ORD101.PGM.RPGL` - ligne 226

```
vatRate = GetVatRate(GetArtVatCode(odorid));
```

Informations complémentaires :

- Le prototype de la procédure est défini dans `OPROTOSRC/VAT_RPGL.FINC`

Diagrammes base de données



SAMCO3
QDDSSRC
ADDRESS.PF
ART200D.DSPF
ART201D.DSPF
ART202D.DSPF
ART250D.DSPF
ART301D.DSPF
ARTICLE.PF
ARTICLE1.LF
ARTICLE2.LF
ARTIPRO1.LF
ARTIPRO2.LF
ARTIPROV.PF
COU200D.DSPF
COU301D.DSPF
COUNTR1.LF
COUNTRY.PF
CUS200D.DSPF
CUS250D.DSPF
CUS301D.DSPF
CUSTADRE.PF
CUSTOME1.LF
CUSTOME2.LF
CUSTOMER.PF
DETORD.PF
DETORD1.LF
FAM301D.DSPF
FAMILL1.LF
FAMILL1.PF

Dans le dossier QDDSSRC du projet SAMCO3 de l'espace de travail, analyser les fichiers d'extension PF et LF et créer un diagramme des relations entre ces différents fichiers. Stocker le résultat dans le fichier SAMCO3_DB2REL.MD

Advanced ^



All tasks completed!

Todo List Updated

- Identifier tous les fichiers PF et LF
- Lire et analyser chaque fichier PF pour comprendre leur structure
- Lire et analyser chaque fichier LF pour identifier les relations
- Créer le diagramme des relations
- Générer le fichier SAMCO3_DB2REL.MD avec le diagramme

Task Completed

Analyse terminée et documentation créée avec succès.

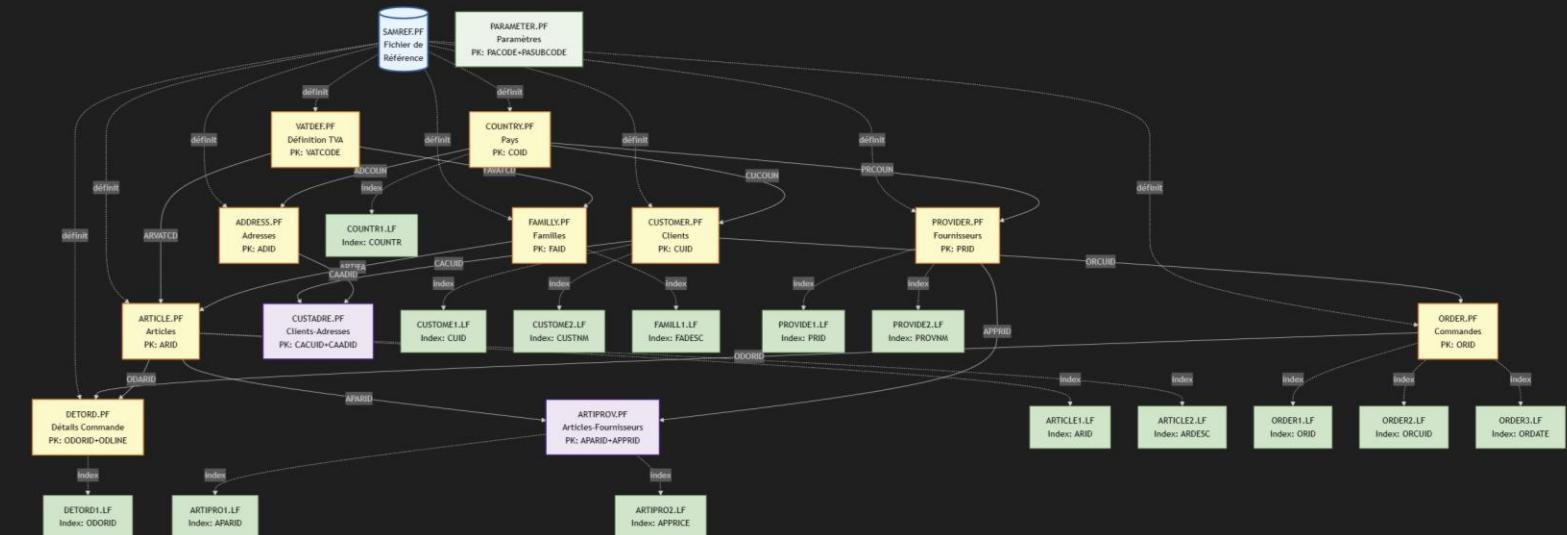
Le fichier SAMCO3_DB2REL.MD a été généré dans le répertoire AGROEX avec :

- 13 fichiers physiques (PF) analysés
- 14 fichiers logiques (LF) analysés
- Diagramme complet des relations au format Mermaid
- Documentation détaillée de chaque table avec :
 - Structure des champs
 - Clés primaires et étrangères
 - Relations entre tables
 - Index disponibles
- Résumé des relations (9 One-to-Many, 2 Many-to-Many)
- Tables de liaison identifiées (ARTIPROV, CUSTADRE)
- Statistiques complètes de la base de données

Diagrammes base de données



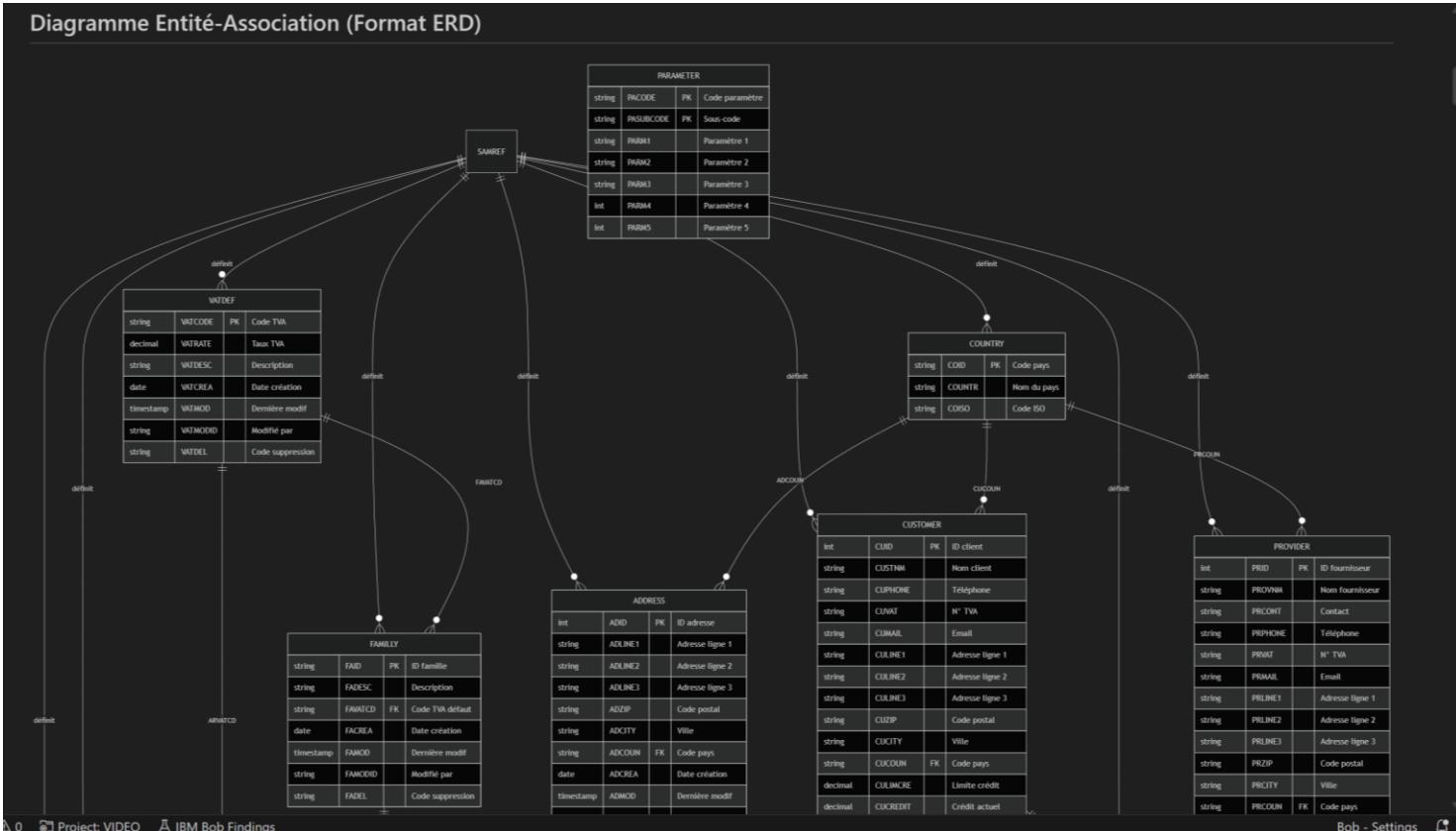
Diagramme Graphique des Relations



Légende du diagramme :

- Bleu clair : Fichier de référence (SAMREF.PF)
- Jaune : Fichiers physiques principaux (PF)
- Violet : Tables de liaison (PF)
- Vert clair : Table de configuration (PF)
- Vert foncé : Fichiers logiques (LF) - Index
- Flèches pleines : Relations de clés étrangères (FK)
- Flèches pointillées : Relations de définition ou d'indexation

Diagrammes base de données



Architecture d'applications complètes



- ✓ SAMCO3
- > QBNDSRC
- > QCBLSRC
- > QCLSRC
- > QCMDSRC
- > QDDSSRC
- > QDTASRC
- > QILESRC
- > QILESRVSR
- > QMSGFSRC
- > QPNLSRC
- > QPROTOSRC
- > QRPGLESRC
- > QRPGSRC
- > QSQLSRC
- > QSRVSR
- > QTRGSRC

A partir du dossier SAMCO3 dans l'espace de travail, fournir, en français, une explication détaillée du code, en expliquant :

- ce que fait l'application
- la conception de la base de données
- la conception de l'interface utilisateur
- les liens entre les différents composants de l'application

et fournir des diagrammes architecturaux détaillant tout cela.

Enregistrer ces informations en français dans un fichier intitulé "architecture_SAMCO3.md".
Tous les diagrammes doivent être au format Mermaid et intégrés dans le fichier markdown.

🛠 Advanced ^



Bob - Settings

Todo List Updated

- Lire les fichiers de base de données (QDDSSRC/*.PF et QSQLSRC/*.sql)
- Lire les fichiers logiques (QDDSSRC/*.LF)
- Lire les fichiers d'écran (QDDSSRC/*.DSPF)
- Lire les programmes principaux (QRPGLESRC/*.RPGLE)
- Lire les modules de service (QILESRVSR/*.ILESVPGM)
- Lire les prototypes (QPROTOSRC/*.RPGLEINC)
- Analyser l'architecture globale
- Créer le fichier architecture_SAMCO3.md avec diagrammes Mermaid

Architecture d'applications complètes



Preview architecture_SAMCO3.md X

...

Architecture de l'Application SAMCO3

Vue d'ensemble

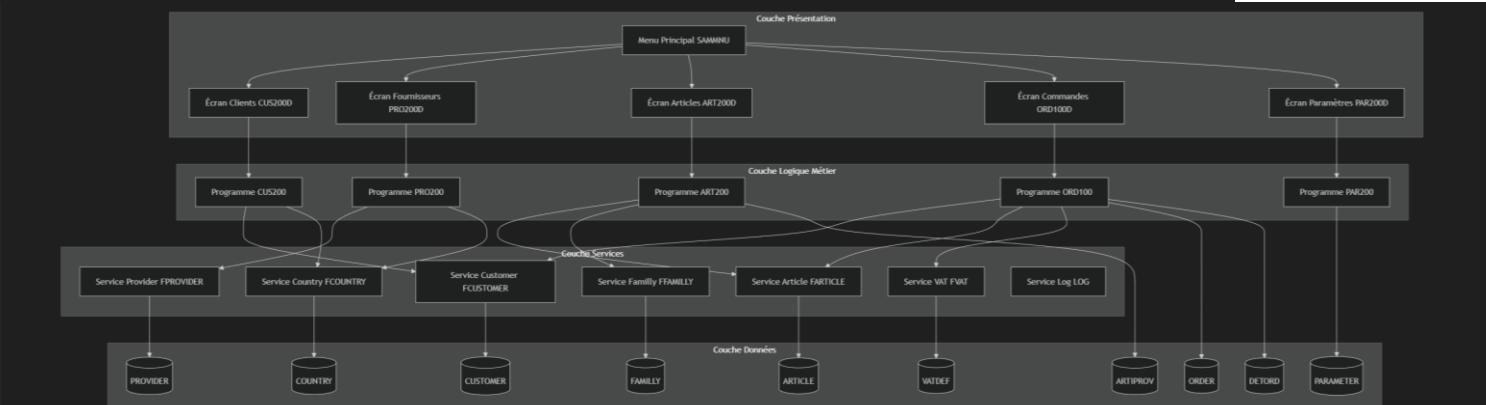
SAMCO3 est une application de gestion commerciale développée en RPG/ILE pour IBM i (AS/400). Elle permet de gérer les clients, les fournisseurs, les articles, et les commandes dans un environnement de vente et d'approvisionnement.

1. Objectif de l'Application

L'application SAMCO3 est conçue pour :

- **Gérer les clients** : création, modification, consultation des informations clients
- **Gérer les fournisseurs** : gestion des fournisseurs et des articles qu'ils proposent
- **Gérer les articles** : catalogue d'articles
- **Gérer les commandes** : création et suivi
- **Gérer les paramètres** : configuration des paramètres

2. Architecture Globale

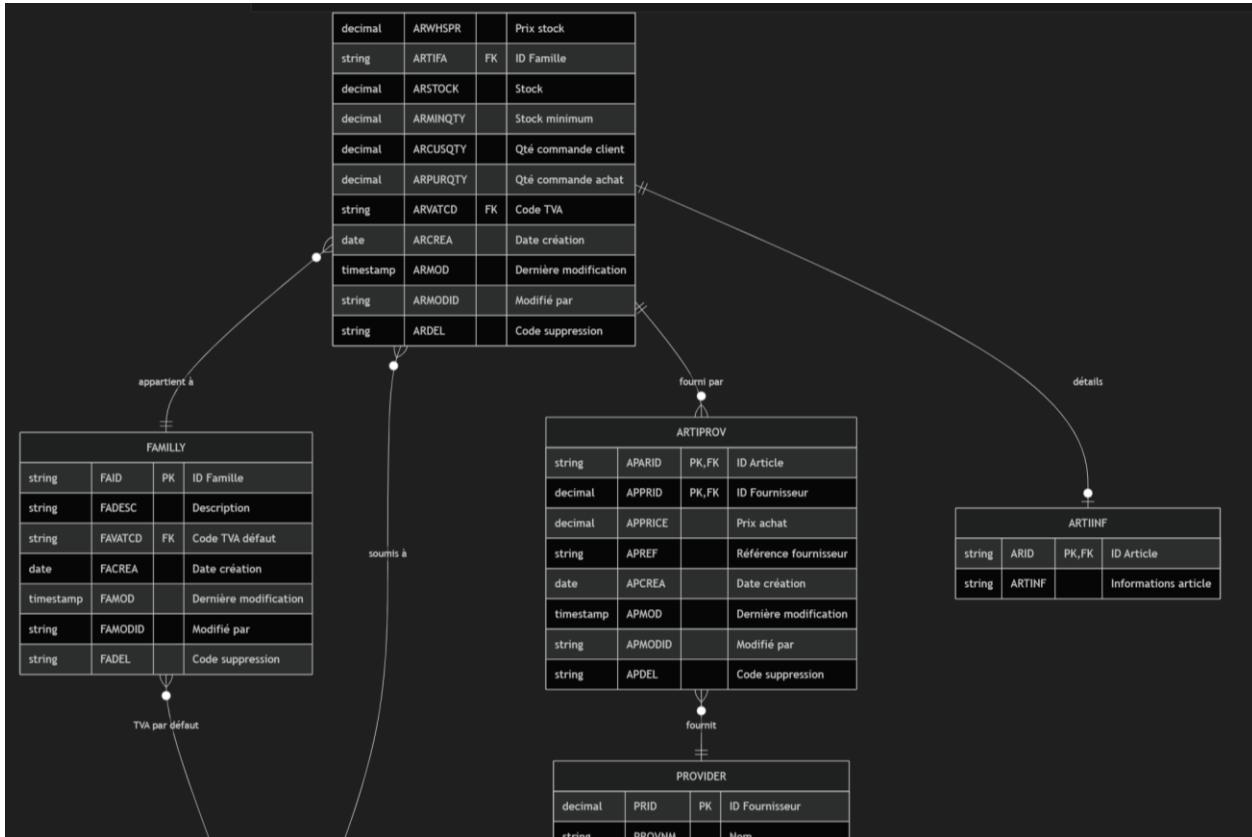


Architecture d'applications complètes



3. Conception de la Base de Données

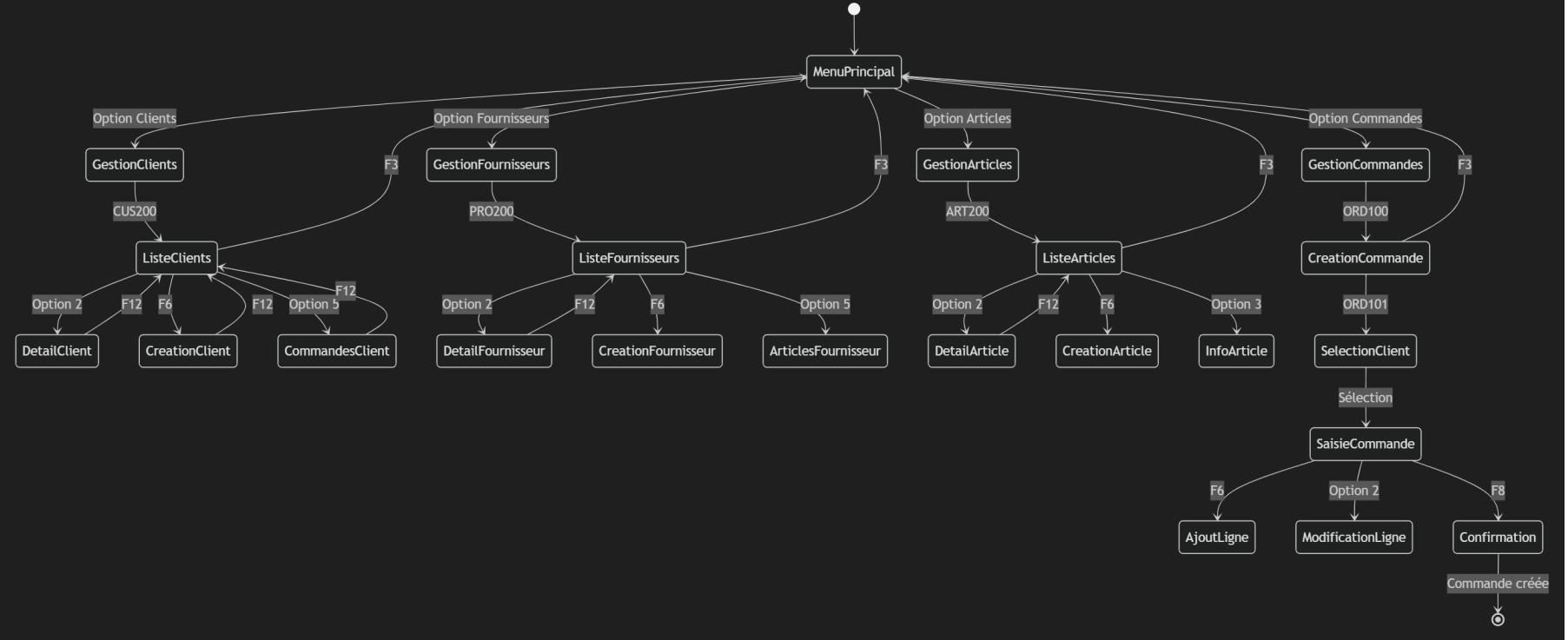
3.1 Modèle Entité-Relation



Architecture d'applications complètes



4.3 Flux de Navigation

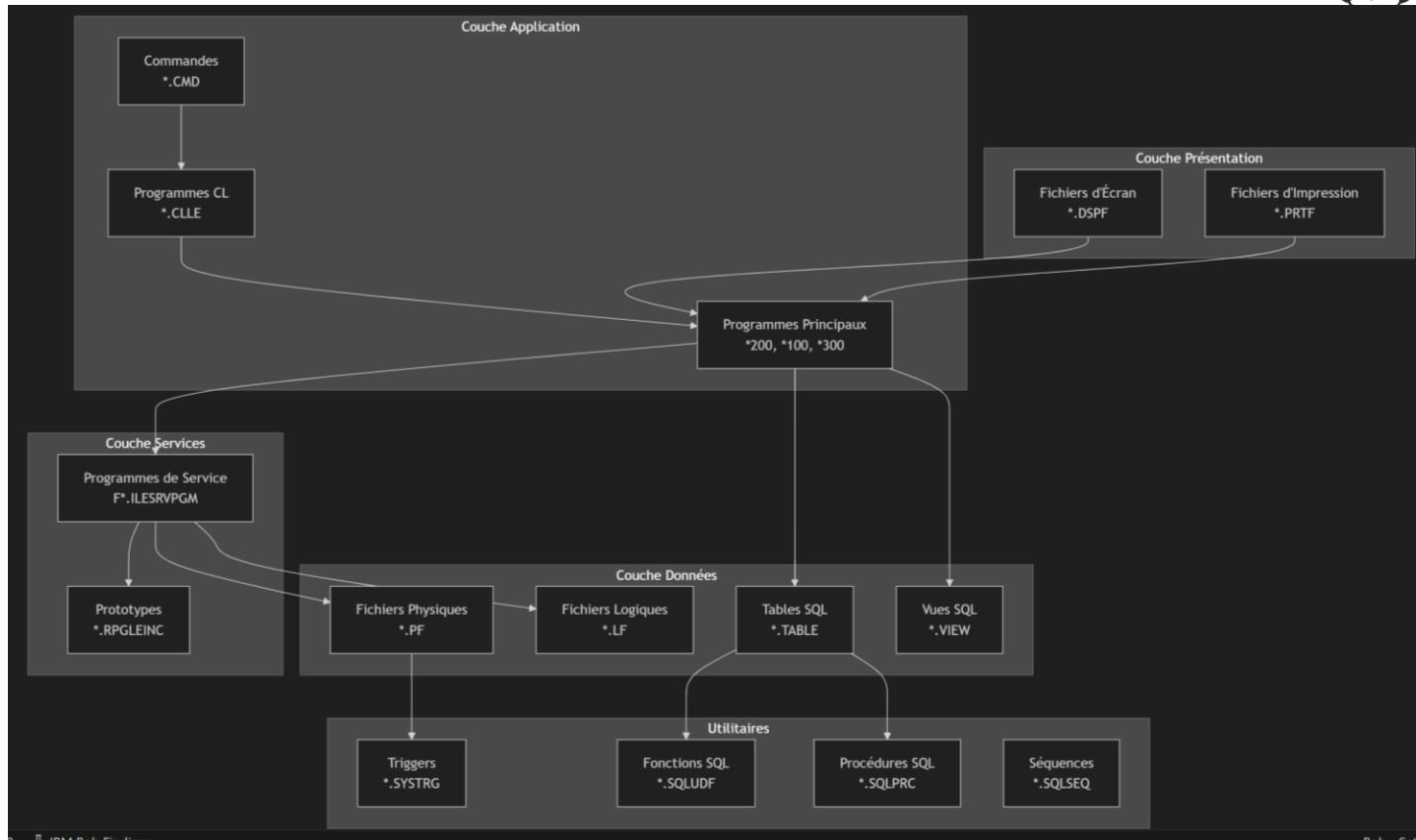


Architecture d'applications complètes



5. Architecture Logicielle

5.1 Architecture en Couches



Architecture d'applications complètes



5.3 Fonctions des Services

FCUSTOMER (Service Client)

- `GetCusName()` : Récupérer le nom du client
- `GetCusPhone()` : Récupérer le téléphone
- `GetCusVat()` : Récupérer le numéro de TVA
- `GetCusMail()` : Récupérer l'email
- `GetCusAdrLine1/2/3()` : Récupérer les lignes d'adresse
- `GetCusZip()` : Récupérer le code postal
- `GetCusCity()` : Récupérer la ville
- `GetCusCountry()` : Récupérer le code pays
- `GetCusLimCredit()` : Récupérer la limite de crédit
- `GetCusCredit()` : Récupérer le crédit actuel
- `ExistCus()` : Vérifier l'existence d'un client
- `IsCusDeleted()` : Vérifier si le client est supprimé
- `SltCustomer()` : Sélectionner un client
- `CloseCUSTOMER()` : Fermer le fichier

FARTICLE (Service Article)

- `GetArtDesc()` : Récupérer la description
- `GetArtRefSalPrice()` : Récupérer le prix de vente
- `GetArtStockPrice()` : Récupérer le prix de stock
- `GetArtFam()` : Récupérer la famille
- `GetArtStock()` : Récupérer le stock

7. Patterns et Bonnes Pratiques

7.1 Pattern Subfile (Liste Dynamique)

Tous les écrans de liste utilisent le pattern subfile standard :

1. **Préparation** (prp) : Effacer le subfile
2. **Chargement** (lod) : Charger les enregistrements
3. **Affichage** (dsp) : Afficher l'écran
4. **Saisie** (key) : Attendre l'action utilisateur
5. **Validation** (chk) : Valider les options saisies
6. **Action** (act) : Exécuter les actions

7.2 Séparation des Responsabilités

- *Programmes d'écran (200) : Gestion de l'interface utilisateur
- *Programmes de maintenance (300) : Logique métier CRUD
- Programmes de service (F)* : Fonctions réutilisables
- **Programmes utilitaires (*500, 700, 900) : Traitements spécialisés

7.3 Gestion de la Suppression Logique

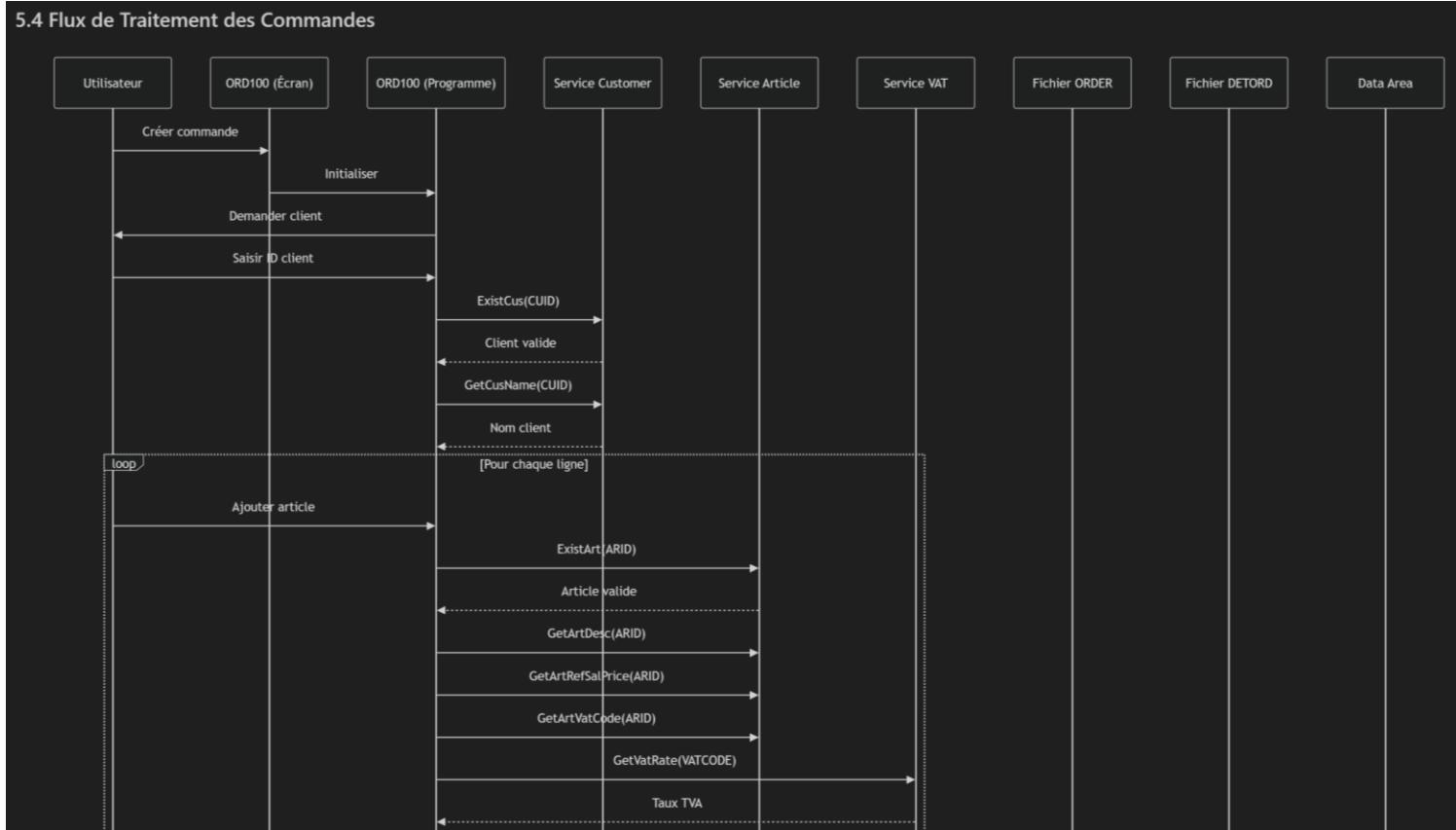
Toutes les tables principales utilisent un champ `*DEL` pour la suppression logique :

- ' ' : Enregistrement actif
- 'X' : Enregistrement supprimé

7.4 Audit Trail

Chaque table contient :

Architecture d'applications complètes

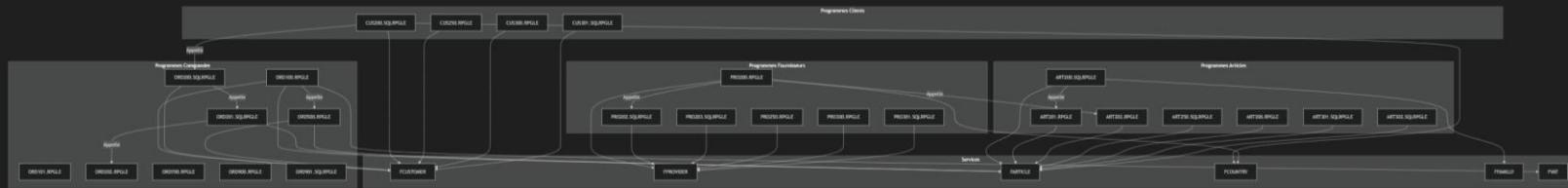


Architecture d'applications complètes

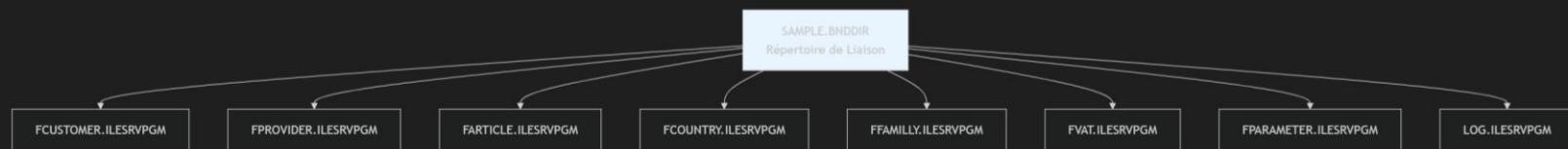


6. Liens entre Composants

6.1 Dépendances des Programmes

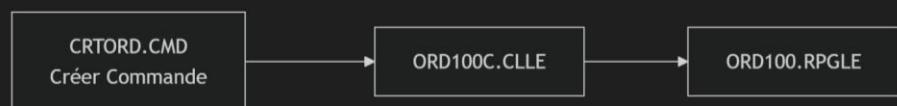


6.2 Binding Directories



Tous les programmes principaux utilisent `BNDDIR('SAMPLE')` pour accéder aux services.

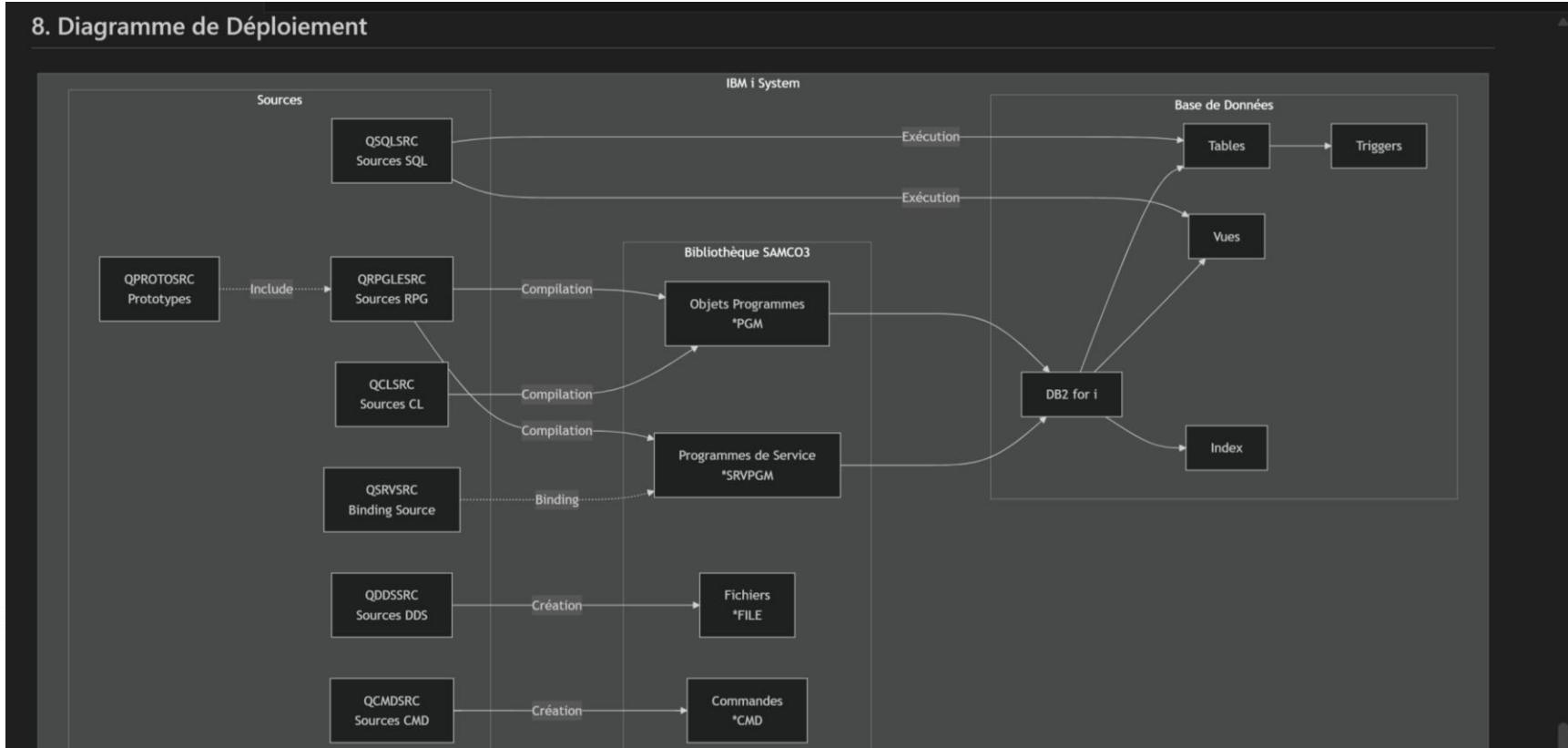
6.3 Commandes Personnalisées



Architecture d'applications complètes



8. Diagramme de Déploiement





2. Amélioration, optimisation de code



Amélioration de code

- ✓ SAMCO1
- > QCLSRC
- > QDDSSRC
- > QDDSSRC2
- > QLBLSRC
- > QRPGLESRC
- ✓ QRPGSRC
 - ≡ hsr200.rpg
 - ≡ hsr210.rpg
 - ≡ hsr215.rpg
 - ≡ hsr217.rpg
 - ≡ hsr220.rpg
 - ≡ hsr230.rpg
 - ≡ hsr341.rpg
 - ≡ hsr342.rpg
 - ≡ hsr701.rpg
 - ≡ hsr711.rpg
 - ≡ hsr721.rpg
 - ≡ hsr735.rpg
 - ≡ hsr761.rpg
- > QSLSRC

Analyser les sources des programmes RPG qui se trouvent dans SAMCO1/QRPGSRC afin de détecter ceux qui travaillent avec des dates qui ont une l'année sur une longueur de 2 (que ces dates soient de type numérique, alphanumériques ou date)

Plan ▾



Excellent ! J'ai trouvé 13 programmes RPG. Je vais maintenant créer un plan d'analyse détaillé pour détecter les dates avec années sur 2 chiffres.

✓= Todo List Created

- Lister tous les programmes RPG dans SAMCO1/QRPGSRC
- Lire et analyser chaque programme RPG pour identifier les patterns de dates sur 2 chiffres
- Identifier les déclarations de variables de dates (numériques, alphanumériques, date)
- Rechercher les patterns spécifiques : YYMMDD, MMDDYY, DDMMYY, YY seul
- Détecter les opérations de manipulation de dates (MOVE, MOVEL, SUBDUR, ADDDUR, etc.)
- Analyser les formats de dates dans les spécifications H et les DATFMT
- Créer un rapport détaillé avec les programmes concernés et les lignes problématiques

Amélioration de code



Programmes Affectés (10/13)

1. `hsr200.rpg` - **CRITIQUE** - Traitement des commandes de vente
2. `hsr210.rpg` - **CRITIQUE** - Traitement des réclamations et cadeaux
3. `hsr215.rpg` - **CRITIQUE** - Ventes d'agence
4. `hsr217.rpg` - **CRITIQUE** - Transactions de réception
5. `hsr220.rpg` - **CRITIQUE** - Impression de factures
6. `hsr230.rpg` - **CRITIQUE** - Impression de bons de livraison
7. `hsr701.rpg` - **CRITIQUE** - Rapports de transactions d'inventaire
8. `hsr711.rpg` - **CRITIQUE** - Rapports de suivi des bons
9. `hsr721.rpg` - **CRITIQUE** - Rapports d'adresses clients
10. `hsr761.rpg` - **MOYEN** - Rapports de détails de commandes

Programmes Non Affectés (3/13)

- `hsr341.rpg` - Fenêtre de recherche F4 (pas de manipulation de dates)
- `hsr342.rpg` - Fenêtre d'allocations (pas de manipulation de dates)
- `hsr735.rpg` - Générateur de rapports de contrôle (pas de manipulation de dates)

B. Logique de Fenêtrage de Siècle

Lignes 414-420 :

C	YY	IFGE 40
C	YY	ANDLE99
C	Z-ADD19	CC
C	ELSE	
C	Z-ADD20	CC

Impact : Fenêtre de siècle 1940-2039. Les dates après 2039 seront mal interprétées.

Amélioration de code



```
≡ COU200BFF.RPGLE ×
SAMCO3 > QRPGLESRC > ≡ COU200BFF.RPGLE > PNL01
1  **FREE
2  //=====
3  // Programme : COU200B
4  // Description : Gestion des pays - Affi
5  // Conversion : RPG Fixed -> RPG Full Fr
6  //=====
7  Ctl-Opt ActGrp(*New);
8
9  // Déclaration des fichiers
10 Dcl-F COU200D WorkStn InfDs(INFDS) SFile
11 Dcl-F COUNTRY Usage(*Update) Keyed;
12
13 // Structure de données pour les informations du fichier
14 Dcl-Ds INFDS;
15 |  LRRN Int(5) Pos(378);
16 End-Ds;
17
18 // Variables globales
19 Dcl-S PANEL Int(10) Inz(1);
20 Dcl-S EtapePanneau1 Char(3) Inz('PRP');
21 Dcl-S EtapePanneau2 Char(3) Inz('PRP');
22 Dcl-S NumEnregSubfile Int(10) Inz(0);
23 Dcl-S NumEnregBase Int(10) Inz(0);
24 Dcl-S ErreurDetectee Ind Inz(*Off);
25
26 //=====
27 // Programme principal
28 //=====
29
30 Dow PANEL <> 0;
31   Select;
32   When PANEL = 1;
33   |  PNL01();
34   When PANEL = 2;
```

A partir du source COU200BFF.RPGLE ouvert dans l'espace de travail, renommer les variables afin de leur donner un nom plus explicite dans le but d'améliorer la lisibilité et la maintenabilité et stocker le résultat dans le fichier COU200CFF.RPGLE

Code ^



```
18 // Variables globales
19 Dcl-S NumPanneau Int(10) Inz(1);
20 Dcl-S EtapePanneau1 Char(3) Inz('PRP');
21 Dcl-S EtapePanneau2 Char(3) Inz('PRP');
22 Dcl-S NumEnregSubfile Int(10) Inz(0);
23 Dcl-S NumEnregBase Int(10) Inz(0);
24 Dcl-S ErreurDetectee Ind Inz(*Off);
25
26 //=====
27 // Programme principal
28 //=====
29
30 Dow NumPanneau <> 0;
31   Select;
32   When NumPanneau = 1;
```

Refactoring terminé avec succès !

Le fichier COU200CFF.RPGLE a été créé avec des noms de variables plus explicites pour le code.

Variables renommées :

Ancien nom	Nouveau nom	Description
INFDS	InfosEcran	Structure d'informations de l'écran
LRRN	NumLigneRelative	Numéro de ligne relative dans le subfile
PANEL	NumPanneau	Numéro du panneau actif
STEP01	EtapePanneau1	Étape du traitement du panneau 1
STEP02	EtapePanneau2	Étape du traitement du panneau 2
RRN01	NumEnregSubfile	Numéro d'enregistrement dans le subfile
RRB01	NumEnregBase	Numéro d'enregistrement de base/position
ERR01	ErreurDetectee	Indicateur d'erreur détectée

Optimisation de code



Optimiser le code du programme VIDEO.RPGL ouvert dans l'espace de travail

Plan ▾

⚡ ⚡ ⚡ ➔

② Avant de créer le plan d'optimisation, j'aimerais clarifier vos priorités. Quels aspects souhaitez-vous privilégier dans l'optimisation ?

Performance (vitesse d'exécution, accès aux données, gestion mémoire)

Maintenabilité (lisibilité du code, structure, documentation, réutilisabilité)

Modernisation (conversion vers SQL embarqué, procédures, modules)

Tous les aspects (optimisation complète et équilibrée)



Optimisation de code



Todo List Created

- Analyser le code source et identifier les opportunités d'optimisation
- Créer des procédures réutilisables pour le formatage des données
- Remplacer les fichiers logiques par des requêtes SQL embarquées
- Optimiser la recherche par titre avec SQL LIKE au lieu de %scan
- Implémenter une pagination efficace pour les sous-fichiers
- Remplacer les indicateurs numériques par des indicateurs nommés
- Ajouter une gestion d'erreurs structurée avec ON-ERROR
- Créer des constantes pour les valeurs magiques
- Refactoriser les sous-routines en procédures
- Ajouter une documentation complète (commentaires et prototypes)
- Optimiser la gestion de la mémoire et des variables
- Créer un module séparé pour les utilitaires de formatage
- Tester les performances avant/après optimisation
- Documenter les changements et créer un guide de migration

Plan d'optimisation complet créé pour VIDEO.RPGLÉ avec 3 documents détaillés :

Documents Livrés

1. PLAN_OPTIMISATION_VIDEO.md (678 lignes)
 - 14 tâches d'optimisation détaillées
 - Exemples de code avant/après
 - Planning sur 10 semaines
 - Stratégie de déploiement
2. ARCHITECTURE_COMPARISON.md (598 lignes)
 - Diagrammes architecture avant/après
 - Comparaison des flux de traitement
 - Métriques de performance détaillées
 - Exemples de code comparés
3. RESUME_EXECUTIF.md (329 lignes)
 - Synthèse pour décideurs
 - ROI et investissement
 - Risques et mitigation
 - Recommandations d'action



Optimisation de code

🎯 Gains Attendus

Performance : 40-80% selon les opérations

Maintenabilité : -40% de code, +200% de documentation

Qualité : Standards RPG ILE modernes, tests unitaires possibles

📊 Plan en 14 Étapes

- Phase 1 : Préparation (2 tâches)
- Phase 2 : Modernisation SQL (2 tâches)
- Phase 3 : Optimisation sous-fichiers (1 tâche)
- Phase 4 : Amélioration lisibilité (3 tâches)
- Phase 5 : Refactorisation (2 tâches)
- Phase 6 : Optimisation mémoire (2 tâches)
- Phase 7 : Validation (2 tâches)

Prêt à passer en mode Code pour l'implémentation.

🚀 Stratégie de Déploiement

Approche Recommandée : **Déploiement Progressif**



🎯 Axes d'Optimisation Identifiés

1. Performance (35% du plan)

- ✗ Lecture séquentielle complète des fichiers
- ✗ Utilisation de `%scan()` sur tous les enregistrements
- ✗ Pas de limitation du nombre d'enregistrements
- ✗ Chargement complet des sous-fichiers en mémoire

2. Maintenabilité (30% du plan)

- ✗ Code monolithique avec sous-routines
- ✗ Logique dupliquée (formatage NULL)
- ✗ Pas de procédures réutilisables
- ✗ Documentation limitée

3. Modernisation (25% du plan)

📋 Plan d'Action Détailé

Phase 1 : Préparation et Analyse (Tâches 1-2)

- Tâche 1 : Analyser le code source

Statut: Complété

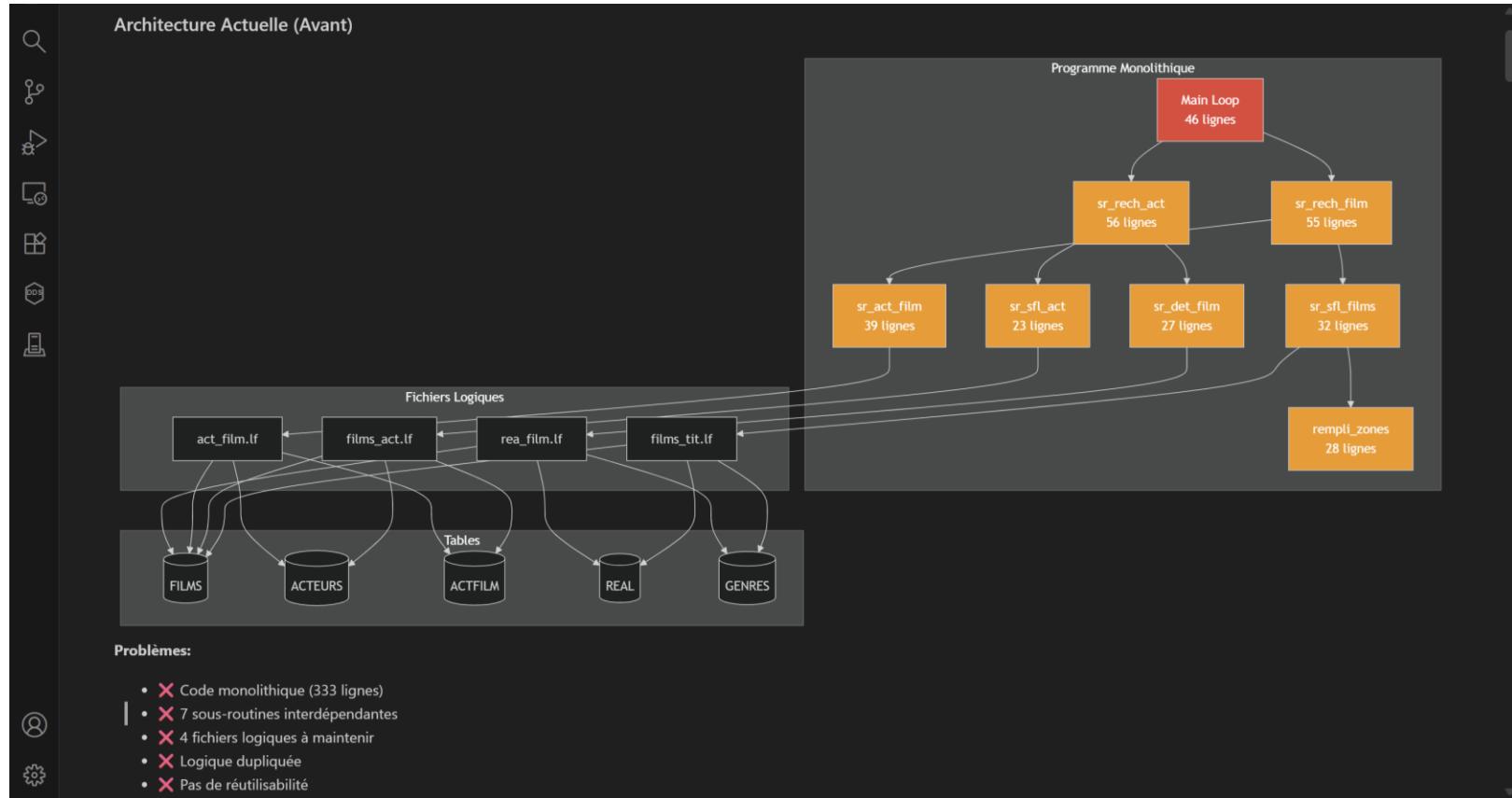
Résultats:

- 333 lignes de code analysées
- 6 sous-routines identifiées
- 4 fichiers logiques utilisés
- 12 indicateurs numériques

- Tâche 2 : Créer des procédures réutilisables pour le formatage

Problème actuel:

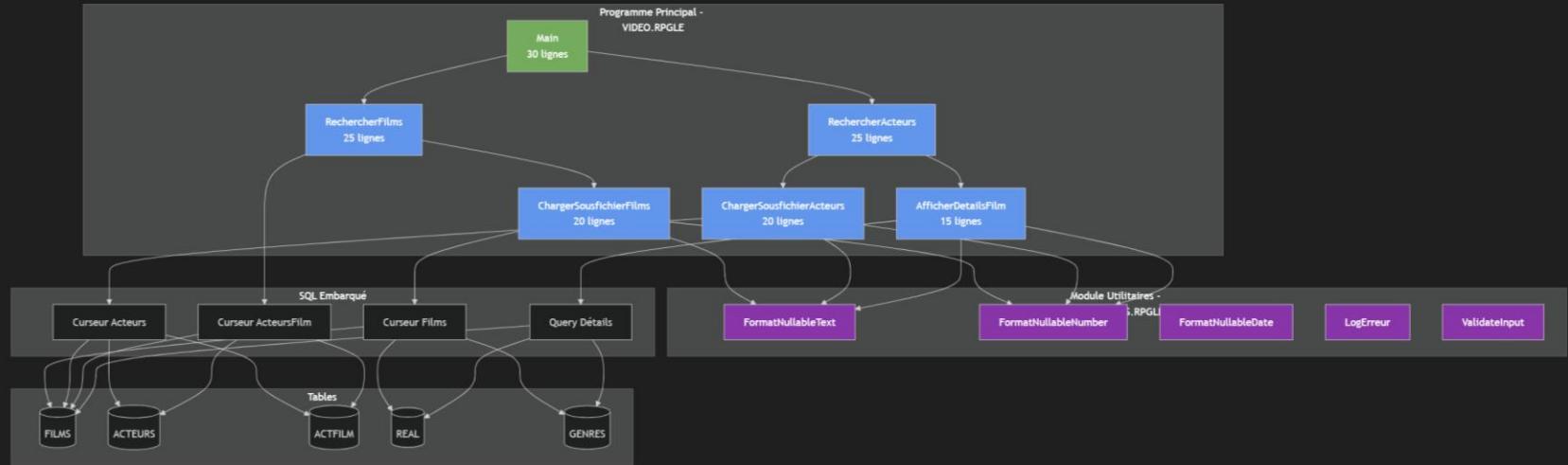
Optimisation de code



Optimisation de code



Architecture Optimisée (Après)



Avantages:

- Code modulaire (~200 lignes)
- 5 procédures indépendantes
- Module utilitaire réutilisable
- SQL embarqué optimisé
- Pas de fichiers logiques à maintenir



3. Restructuration, transformation, conversion de code

Conversion RPG Fixe vers RPG Full Free



Convertir en RPG Full Free la sous-routine CalNbrM (lignes 5079 à 5101) qui s trouve dans le source du programme ADH02.SQLRPGLE ouvert dans l'espace de travail.

1.sqlrpgle adh02.sqlrpgle adh02.sqlrpgle: Original ↔ IBM Bob's Changes (Editable)

ASSUEX > QRPGLESRC > adh02.sqlrpgle > CalNbrM

```
5078 5078 | c*---/ Calcul du nombre de mois dans périodicité
5079  | c  CalNbrM    Begsr
5080  | c  AnneeTarif Ifne  *Zero
5081  | c           Movel  AnneeTarif  waaaamm  6 0
5082  | c           Else
5083  | c           Movel  *YEAR    wAaaaamm
5084  | c           Endif
5085  | c           Move   uMonth   waaaamm
5086  | c*---/
5087  | c           Move   DateIns  waaaaIns
5088  | c           Movel  DateIns  wmmIns
5089  | c           Z-add  1       wwNbm    2 0
5090  | c*---/ Si on se trouve avant la date d'inscription
5091  | c  wAAAAMMIns Ifge  wAAAAMM
5092  | c           Movel  DateIns  wwMmIns  2 0
5093  | c           Else
5094  | c*---/ Sinon, ça n'a plus de sens (de faire apparaître un prorata é
5095  | c           Move   '01'    wwMmIns
5096  | c           Endif
5097  | c*---/ Décidé le 07/07/2010, on met le mois d'effet
5098  | c           Move   Dtfmm   wwMmIns
5099  | c*---/
5100  | c           Exsr   CalwwMmIns
5101  | c           Endsr
5079+ // Sous-routine CalNbrM - Calcul du nombre de mois
5080+ Begsr CalNbrM;
```

466+	d	wwaaamm	s	6s 0
467+	d	wwaaaIns	s	4s 0
468+	d	wwNbmm	s	2s 0
469+	d	wwMmIns	s	2s 0

```
// Sous-routine CalNbrM - Calcul du nombre de mois
Begsr CalNbrM;

// Initialisation de waaaamm avec l'année du tarif ou l'année courante
If AnneeTarif <> *Zero;
  | waaaamm = AnneeTarif * 100; // Movel équivalent
Else;
  | waaaamm = %Dec(%Char(%Date():*ISO0):6:0); // *YEAR équivalent
  | waaaamm = %Int(%Subst(%Char(%Date():*ISO):1:4)) * 100;
Endif;

// Ajout du mois courant
wwaaamm = %Rem(waaaamm:100) * 10 + uMonth; // Move équivalent

// Extraction de l'année et du mois de la date d'inscription
```

Conversion RPG Fixe vers RPG Full Free



RPGSRC > COU200B.RPG

```

FCOU200D CF E          WORKSTN      KINFDS INFDS
F                           RRN01 KSFILe SFL01
*
F COUNTRY UF E          K          DISK
I INFDS DS
*
C     LOOP    TAG
C     PANEL   CASEQ1     PNL01
C     PANEL   CASEQ2     PNL02
C     ENDCS
C     PANEL   CABEQ0     ENDPGM
C     GOTO    LOOP
C     ENDPGM TAG
C     SETON
C     LR
*
C     PNL01 BEGSR
C     STEP01 CASEQ'PRP'  S01PRP
C     STEP01 CASEQ'LOD'  S01LOD
C     STEP01 CASEQ'DSP'  S01DSP
C     STEP01 CASEQ'KEY'  S01KEY
C     STEP01 CASEQ'CHK'  S01CHK
C     STEP01 CASEQ'ACT'  S01ACT
C     ENDCS
C     ENDSR
C     S01PRP BEGSR
C     Z-ADD0   RRN01    40
C     SETON
C     WRITECTL01
C     SETOF
C     MOVE 'LOD'   STEP01
C     ENDSR

```



```

SAMCO3 > QRPGLESRC > COU200BFF.RPGLE > ...
1  **FREE
2  //=====
3  // Programme : COU200B
4  // Description : Gestion des pays - Affichage et modification
5  // Conversion : RPG Fixed -> RPG Full Free
6  //=====
7  Ctl-Opt ActGrp(*New);
8
9  // Déclaration des fichiers
10 Dcl-F COU200D WorkStn InfDs(INFDS) SFile(SFL01:RRN01);
11 Dcl-F COUNTRY Usage(*Update) Keyed;
12
13 // Structure de données pour les informations du fichier écran
14 Dcl-Ds INFDS;
15 | LRRN Int(5) Pos(378);
16 End-Ds;
17
18 // Variables globales
19 Dcl-S PANEL Int(10) Inz(1);
20 Dcl-S STEP01 Char(3) Inz('PRP');
21 Dcl-S STEP02 Char(3) Inz('PRP');
22 Dcl-S RRN01 Int(10) Inz(0);
23 Dcl-S RRB01 Int(10) Inz(0);
24 Dcl-S ERROR1 Ind Inz(*Off);
25
26 //=====
27 // Programme principal
28 //=====
29
30 Dow PANEL <> 0;
31   Select;
32   | When PANEL = 1;
33   | | PNL01();
34   | When PANEL = 2;
35   | | PNL02();
36   EndSl;
37 EndDo;
38
39 *InLR = *On;

```

```

40
41 //=====
42 // PNL01 - Gestion du panneau de liste
43 //=====
44 Dcl-Proc PNL01;
45
46   Select;
47   | When STEP01 = 'PRP';
48   | | S01PRP();
49   | When STEP01 = 'LOD';
50   | | S01LOD();
51   | When STEP01 = 'DSP';
52   | | S01DSP();
53   | When STEP01 = 'KEY';
54   | | S01KEY();
55   | When STEP01 = 'CHK';
56   | | S01CHK();
57   | When STEP01 = 'ACT';
58   | | S01ACT();
59   EndSl;
60
61 End-Proc;
62
63 //=====
64 // S01PRP - Préparation du subfile
65 //=====
66 Dcl-Proc S01PRP;
67
68   RRN01 = 0;
69   *In30 = *On;
70   Write CTL01;
71   *In30 = *Off;
72   STEP01 = 'LOD';
73
74 End-Proc;

```

Conversion DDS → SQL



Convertir en SQL tous les PF et tous les LF du dossier QDDSSRC de SAMCO3 et mettre le résultat dans le dossier QSLSRC de SAMCO3

The screenshot shows a terminal window with the following details:

- Title bar: ORDER.SQL x
- Path: SAMCO3 > QSLSRC > ORDER.SQL
- Toolbar: Includes icons for Advanced, Refresh, Favorites, Print, and Copy/Save.
- Code content (lines 1-30):

```
1  -- =====
2  -- Table: ORDER (Commandes)
3  -- Description: Table des commandes clients
4  -- Fichier source: QDDSSRC/ORDER.PF
5  -- =====
6
7  CREATE TABLE "ORDER" (
8      ORID      DECIMAL(6,0)    NOT NULL,
9      ORYEAR    DECIMAL(4,0)    NOT NULL,
10     ORCUID    DECIMAL(5,0)    NOT NULL,
11     ORDATE    DECIMAL(8,0),
12     ORDATDEL  DECIMAL(8,0),
13     ORDATCLO  DECIMAL(8,0),
14
15     CONSTRAINT PK_ORDER PRIMARY KEY (ORID),
16     CONSTRAINT FK_ORDER_CUSTOMER FOREIGN KEY (ORCUID)
17     |   REFERENCES CUSTOMER(CUID)
18 )
19 RCDfmt FORDE;
20
21 LABEL ON TABLE "ORDER" IS 'Order File';
22
23 LABEL ON COLUMN "ORDER" (
24     ORID      IS 'ORD NUM',
25     ORYEAR    IS 'YEAR',
26     ORCUID    IS 'CUST ID',
27     ORDATE    IS 'ORDER DATE',
28     ORDATDEL  IS 'DELIVERY DATE',
29     ORDATCLO  IS 'CLOSE DATE'
30 );
```

Conversion DDS → SQL



Conversion DDS vers SQL - SAMCO3

Vue d'ensemble

Ce dossier contient la conversion complète des fichiers DDS (Data Description Specifications) du dossier QDDSSRC en scripts SQL standard.

Fichiers générés

Tables SQL (CREATE TABLE)

Fichier SQL	Fichier DDS Source	Description
COUNTRY.SQL	COUNTRY.PF	Table des pays
VATDEF.SQL	VATDEF.PF	Définitions des codes TVA
FAMILLY.SQL	FAMILLY.PF	Familles d'articles
ARTICLE.SQL	ARTICLE.PF	Articles
PROVIDER.SQL	PROVIDER.PF	Fournisseurs
CUSTOMER.SQL	CUSTOMER.PF	Clients
ADDRESS.SQL	ADDRESS.PF	Adresses
CUSTADRE.SQL	CUSTADRE.PF	Liaison Clients-Adresses
ORDER.SQL	ORDER.PF	Commandes
DETORD.SQL	DETORD.PF	Détails des commandes
PARAMETER.SQL	PARAMETER.PF	Paramètres système

- SAMCO3
- ↳ QSQLSRC
 - ART001.SQLPR
 - ARTICLE.SQL
 - ARTINF.TABLE
 - ARTIPROV.SQL
 - ARTLSTDAT.VIEW
 - COUNTRY.SQL
 - CREATE_ALL.SQL
 - CUSSEQ.SQLSEQ
 - CUSTADRE.SQL
 - CUSTOMER.SQL
 - DETORD.SQL
 - FAMILLY.SQL
 - INDEXES.SQL
 - ISOTODATE.SQLUDF
 - ISOTODATE4.SQLUDF
 - ORD701.SQLTRG
 - ORDER.SQL
 - ORDERCUS.VIEW
 - PARAMETER.SQL
 - PROVIDER.SQL
 - README.MD
 - VATDEF.SQL
- ↳ QSRVSR
- ↳ QTRGSRC
 - architecture_SAMCO3.md
 - SAMCO3_DB2REL.MD
- VIDEO
- OUTLINE
- TIMELINE

Conversion accès BD natifs (RLA) → accès SQL



Dans le programme VIDEOFF.RPGLE ouvert dans l'espace de travail, refactorisez le code afin de remplacer les accès natifs RPG aux fichiers base de données par des accès SQL, en suivant les consignes suivantes :

- Pour la gestion des valeurs nulles : il faut utiliser en RPG des variables de type INT(5) et ces variables ne peuvent pas être définies dans des structures de données (elles doivent être définies comme variables isolées)
- Si génération de structures de données (DS) : si possible utiliser des DS externes avec EXTNAMES pour simplifier le code

Sauvegarder le résultat dans le fichier VIDEOFFSQL.RPGLE

Advanced ^



Todo List Updated

- Analyser la structure des tables SQL et les fichiers logiques
- Créer le nouveau fichier VIDEOFFSQL.RPGLE avec les structures de données
- Définir les variables indicatrices NULL (INT(5)) comme variables isolées
- Remplacer les opérations READ/READE/SETLL par des curseurs SQL
- Remplacer les opérations CHAIN par des SELECT SQL
- Adapter la gestion des valeurs NULL avec les variables indicatrices

Conversion accès BD natifs (RLA) → accès SQL



```
dcl-f films_tit disk keyed;
dcl-f films_act disk keyed;
dcl-f act_film disk keyed;
dcl-f rea_film disk keyed;
```

```
dcl-proc sfl_act;
nommin = %trim(nom);
nommax = %trimr(nommin) + max;
setll nommin films_act;
read films_act;
dow nomact <= nommax
  and not %eof;
  rrnact = rrnact + 1;
  debnom = nomact;
  if %nullind(preact) = *on;
    debpre = ...';
  else;
    debpre = preact;
  endif;
  debtitre = titre;
  write actors_sfl;
  read films_act;
enddo;
end-proc;
```

```
dcl-proc sfl_act;
nommin = %trim(nom);
nommax = %trimr(nommin) + max;

// Curseur SQL pour récupérer les acteurs et leurs films
exec sql
DECLARE c_actors CURSOR FOR
SELECT A.CODACT, A.NOMACT, A.PREACT, F.CODFILM, F.TITRE
FROM VIDEO.ACTEURS A
JOIN VIDEO.ACTFILM AF ON A.CODACT = AF.CODACT
JOIN VIDEO.FILMS F ON AF.CODFILM = F.CODFILM
WHERE A.NOMACT >= :nommin AND A.NOMACT <= :nommax
ORDER BY A.NOMACT, F.TITRE;

exec sql
OPEN c_actors;

exec sql
FETCH c_actors INTO :ds_acteurs.codact,
                      :ds_acteurs.nomact,
                      :ds_acteurs.preact :ind_preact,
                      :ds_films.codfilm,
                      :ds_films.titre;

dow sqlcode = 0;
  rrnact = rrnact + 1;
  debnom = ds_acteurs.nomact;
  codfilm = ds_films.codfilm;
```

```
if ind_preact < 0;
  debpre = ...';
else;
  debpre = ds_acteurs.preact;
endif;

debititre = ds_films.titre;
write actors_sfl;

exec sql
  FETCH c_actors INTO :ds_acteurs.codact,
                      :ds_acteurs.nomact,
                      :ds_acteurs.preact :ind_preact,
                      :ds_films.codfilm,
                      :ds_films.titre;

enddo;

exec sql
  CLOSE c_actors;
end-proc;
```

Refactorisation : génération de procédures ILE



```
ASSUR1 > QRPGLESRC >  DCLSINPFF.RPGLE > ...
1  **free
2
3  ctl-opt actgrp(*new) option(*srcstmt:*nodebugio);
4
5  //***** ****
6  // Programme : DCLSINPFF
7  // Description : Déclaration de sinistres - Version
8  // Converti de : decl_sin_p.rpgle
9  //***** ****
10
11 // Déclaration des fichiers
12 dcl-f decl_sin_d workstn sfile(chx_ct_sfl:rrn) sfile
13 dcl-f clients disk(*ext) keyed usage(*input);
14 dcl-f ct_det_ctn disk(*ext) keyed usage(*input);
15 dcl-f ct_det_cln disk(*ext) keyed usage(*input);
16 dcl-f types_sin disk(*ext) keyed usage(*input);
17 dcl-f sinistres disk(*ext) keyed usage(*update:*delete:*ou
18
19 // Déclaration des prototypes de programmes externes
20 dcl-pr qcめdexc extpgm('QCМDEXC');
21   cmd char(200) const;
22   cmdlen packed(15:5) const;
23 end-pr;
24
25 dcl-pr enq_h_sin extpgm('ENQHSINFF');
26   *n like(cli_num);
27   *n char(1);
28 end-pr;
29
```

Refactoriser le code du programme

C:\pbourgeois\AI\BOB\PROJETS\ASSUR1\QRPGLESRC\DCLSINPFF.RPGL afin de remplacer l'appel du programme externe ENQHSINFF par l'appel d'une procédure ILE externe stockée dans un programme de service appelé. Pour le programme de service : créer le source en langage de liage (source BND) dans un répertoire appelé QSRVSRC et dans le programme principal faire appel à un répertoire de liage appelé UTIL_BND. Le source du nouveau programme principal s'appellera DCLSINP1 et le source du programme de service qui contiendra la procédure ENQHSINFF s'appellera UTIL_SRV

Advanced ^



✓ Todo List Created

- Analyser les fichiers sources existants
 - Créer le répertoire QSRVSRCC pour les sources de programmes de service
 - Créer la source du programme de service UTIL_SRV.RPGL avec la procédure ENQHSINFF
 - Créer la source de liage UTIL_BND.BND dans QSRVSRCC
 - Créer le nouveau programme principal DCLSINP1.RPGL avec l'appel à la procédure
 - Vérifier la cohérence de tous les fichiers créés

Refactorisation : génération de procédures ILE



ASSUR1 > QSRVSRC > UTIL_BND.BND

```
1  /*=====
2  /* Source de liage : UTIL_BND
3  /* Description : Définit les procédures exportées du programme de service
4  /*           UTIL_SRV
5  /*=====
6
7  STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('UTIL_SRV_V1')
8
9  /*
10 /* Procédure : ENQHSINFF
11 /* Description : Enquête historique sinistres
12 /*
13 EXPORT SYMBOL('ENQHSINFF')
14
15 ENDPGMEXP
```

ASSUR1 > QSRVSRC > UTIL_SRV.RPGLE > ...

```
1  **free
2
3  //*****
4  // Programme de service : UTIL_SRV
5  // Description : Programme de service contenant des procédures utilitaires
6  // Procédures exportées :
7  //   - ENQHSINFF : Enquête historique sinistres
8  //*****
9
10 ctl-opt nomain;
11
12 // Déclaration des fichiers
13 dcl-f sin_cli disk(*ext) keyed usage(*input);
14
15 //*****
16 // Procédure : ENQHSINFF
17 // Description : Enquête historique sinistres - Vérifie si un client a des
18 //   sinistres avec remboursements dans son historique
19 // Paramètres :
20 //   num_cli : Numéro du client (entrée)
21 //   enquête : Résultat de l'enquête (sortie)
22 //     '0' = Client a des sinistres avec remboursements
23 //     'N' = Client n'a pas de sinistres avec remboursements
24 //     '?' = Client non trouvé
25 //*****
26 dcl-proc ENQHSINFF export;
27   dcl-pi *n;
28     num_cli like(cli_num);
29     enquête char(1);
30   end-pi;
31
32 // Déclaration des variables locales
33 dcl-s total_rep packed(8:0);
34
```

ASSUR1 > QRPGLESRC > DCLSINP1.RPGLE > ...

```
1  **free
2
3  ctl-opt actgrp(*caller) option(*srcstmt:*nodebugio) bnddir('UTIL_BND');
4
5  //*****
6  // Programme : DCLSINP1
7  // Description : Déclaration de sinistres - Version ILE avec programme de service
8  // Refactorisé de : DCLSINPFF.RPGLE
9  // Changements : Remplacement de l'appel au programme externe ENQHSINFF par
10 //           l'appel à la procédure ENQHSINFF du programme de service UTIL_SRV
11 //*****
```

Refactorisation : extraction de logique métier



Dans le code du programme DCLSINPFF2.RPGLE ouvert dans l'espace de travail détecter les règles métier complexes qui pourraient être externalisées. Il ne s'agit pas d'externaliser toutes les procédures et les sous-routines du programme, ni les procédures qui ne contiennent que quelques lignes, seulement celles qui contiennent de la logique métier un peu complexe qui pourrait être réutilisée dans d'autres programmes. Stocker le résultat dans le fichier DCLSINPFF2_BUSINESS_RULES.MD

Advanced ^



```
sin_declare = '0';

// Vérification de la nécessité de lancer une enquête sur le client
reset enquete;
reset nb_sin;
reset total_rep;
setll cli_num sin_cli;
if %equal;
  reade cli_num sin_cli;
  dow not %eof(sin_cli);
  if sin_num <> 0;
    nb_sin = nb_sin + 1;
    if accord = '0';
      total_rep = total_rep + mnt_rep;
    endif;
  endif;
  reade cli_num sin_cli;
enddo;
if (nb_sin > 8) and (total_rep > 20000);
  enquete = '0';
else;
  enquete = 'N';
endif.
```

Tâche terminée

Analyse terminée. J'ai identifié et documenté les règles métier complexes du programme DCLSINPFF2.RPGLE dans le fichier [DCLSINPFF2_BUSINESS_RULES.MD](#).

Règles métier identifiées pour externalisation:

1. Règle de Déclenchement d'Enquête Client (Priorité HAUTE)

- Logique complexe analysant l'historique des sinistres
- Critères: >8 sinistres ET >20 000€ de remboursements
- Fort potentiel de réutilisation dans d'autres programmes

2. Génération du Numéro de Sinistre (Priorité MOYENNE)

- Gestion sécurisée de la séquence avec verrouillage
- Garantit la cohérence entre programmes

Le document inclut:

- Description détaillée de chaque règle avec références au code source
- Propositions de procédures exportables pour le service program UTIL_SRV
- Analyse du potentiel de réutilisation
- Plan d'action en 3 phases pour l'externalisation
- Bénéfices attendus (maintenabilité, réutilisabilité, testabilité)

Refactorisation : extraction de logique métier



Règles Métier Identifiées

1. ★ Logique de Déclenchement d'Enquête Client (RECOMMANDÉ)

Localisation : Lignes 227-250 dans la procédure `decl_sin()`

Description :

Cette règle détermine automatiquement si une enquête doit être lancée sur un client lors de la déclaration d'un nouveau sinistre. Elle analyse l'historique des sinistres du client pour évaluer le risque.

Logique métier :

```
// Vérification de la nécessité de lancer une enquête sur le client
reset enquete;
reset nb_sin;
reset total_rep;
setll cli_num sin_cli;
if %equal;
  reade cli_num sin_cli;
  dow not %eof(sin_cli);
    if sin_num <> 0;
      nb_sin = nb_sin + 1;
      if accord = '0';
        total_rep = total_rep + mnt_rep;
      endif;
    endif;
    reade cli_num sin_cli;
  enddo;
if (nb_sin > 8) and (total_rep > 20000);
  enquete = '0';
else;
  enquete = 'N';
endif;
else;
  enquete = '?';
endif;
```

Critères d'évaluation :

- Nombre de sinistres du client > 8
- Montant total des réparations accordées > 20 000 €
- Les deux conditions doivent être remplies simultanément

Résultat :

- '0' : Enquête nécessaire (client à risque)
- 'N' : Pas d'enquête nécessaire
- '?' : Client sans historique

Complexité : ★ ★ ★ ★ (Élevée)

- Parcours de fichier avec accumulation
- Logique conditionnelle multiple
- Calculs sur l'historique client

Réutilisabilité : ★ ★ ★ ★ ★ (Très élevée)

- Applicable à d'autres programmes d'analyse de risque
- Pourrait être utilisée dans des rapports de gestion
- Utile pour la détection de fraude
- Paramètres configurables (seuils)

Recommandation d'externalisation : FORTEMENT RECOMMANDÉ

Proposition de module :

```
// Module : SINRISK (Gestion du risque sinistre)
// Procédure : evaluerRisqueClient()

dcl-proc evaluerRisqueClient export;
  dcl-pi *n char(1);
  pCliNum like(cli_num) const;
  pSeuilNbSin packed(2:0) const options(*nopass);
  pSeuilMontant packed(8:0) const options(*nopass);
end-pi;

// Logique d'évaluation du risque
// Retourne '0', 'N' ou '?'
end-proc;
```

Avantages de l'externalisation :

- Réutilisable dans d'autres programmes (gestion des sinistres, rapports, etc.)
- Facilite les tests unitaires
- Permet de modifier les seuils sans recompiler le programme principal
- Centralise la logique de détection de risque
- Améliore la maintenabilité
- Facilite l'évolution des règles métier (ajout de nouveaux critères)

Conversion de l'interface utilisateur 5250



ASSUR1 > QDDSSRC > decl_sin_d.dsfp

1 Dans le dossier ASSUR1/MODER_UI céer des composants de type REACT (en utilisant IBM Carbon Design System)
2 pour remplacer les écrans 5250 définis dans le fichier DECL_SIN_D.DSPF du dossier QDDSSRC
3
4
5
6



Advanced ^



```
12 A 1 26 'Gestion simplifiée de sinistre'  
13 A COLOR(WHT)  
14 A 1 70DATE(*JOB *YY)  
15 A EDTCDE(Y)  
16 A 7 5 'Choisissez une option, puis appuye-  
17 A z sur ENTREE'  
18 A COLOR(BLU)  
19 A 9 9'1. Déclaration de sinistre'  
20 A 10 9'2. Mise à jour de sinistre'  
21 A 11 9'3. Coordonnées experts'  
22 A 12 9'4. Coordonnées conseillers'  
23 A 13 9'5. Messages'  
24 A 14 9'6. Spoules'  
25 A 20 4'Option : '  
26 A OPTION 1D 0I 20 14  
27 A 24 2'F3=Exit'  
28 A COLOR(BLU)
```

Assurances IBM_PB Gestion simplifiée de sinistre
Application 1 - Menu principal 27/01/2028

Vous êtes connecté sous un profil de type CALL CENTER

Choisissez une option, puis appuyez sur ENTREE

1. Déclaration de sinistre
2. Mise à jour de sinistre
3. Coordonnées experts
4. Coordonnées conseillers
5. Messages
6. Spoules

Option : _

F3=Exit F12=Annuler

Assurances IBM_PB Déclaration de sinistre 27/01/2028

Choisissez un critère de recherche, puis appuyez sur ENTREE.

Numéro de contrat
OU
Numéro de client.
OU
Début du nom du client.

F12=Annuler

Conversion de l'interface utilisateur 5250



Tâche terminée

Projet React créé avec succès dans ASSUR1/MODER_UI

Composants créés

6 composants React remplaçant les écrans 5250 du fichier DECL_SIN_D.DSPF :

1. **MenuPrincipal** - Menu principal avec 6 options
2. **RechercheDeclaration** - Recherche multi-critères (contrat/client/nom)
3. **ChoixContrat** - Liste interactive des contrats client
4. **DetailContrat** - Affichage détaillé d'un contrat
5. **DeclarationSinistre** - Formulaire de déclaration de sinistre
6. **ChoixTypeSinistre** - Modal de sélection du type de sinistre

Technologies

- React 18 + TypeScript
- IBM Carbon Design System (thème dark g100)
- Vite (build tool moderne)
- Architecture modulaire avec types TypeScript

Interface React Moderne - Gestion de Sinistres

Description

Cette application React moderne remplace les écrans 5250 définis dans le fichier DECL_SIN_D.DSPF du moderne, accessible et cohérente.

Écrans Implémentés

L'application reproduit fidèlement les 6 écrans 5250 suivants :

1. Menu Principal (MENU)

- **Composant:** [MenuPrincipal.tsx](#)
- **Description:** Menu principal avec 6 options de navigation
- **Fonctionnalités:**
 - Affichage du profil utilisateur (CALL CENTER)
 - Navigation vers les différentes sections
 - Options F3 (Exit) et F12 (Annuler)

- ✓ ASSUR1
- > DOCS
- ✓ MODER_UI
 - ✓ src
 - ✗ ChoixContrat.tsx
 - ✗ ChoixTypeSinistre.tsx
 - ✗ DeclarationSinistre.tsx
 - ✗ DetailContrat.tsx
 - ✗ MenuPrincipal.tsx
 - ✗ RechercheDeclaration.tsx
 - ✓ types
 - TS index.ts
 - ✗ App.tsx
 - ✗ index.scss
 - ✗ main.tsx
 - ✗ .gitignore
 - ✗ index.html
 - ✗ package.json
 - ✗ README.md
 - ✗ tsconfig.json
 - ✗ tsconfig.node.json
 - ✗ vite.config.ts

Conversion de l'interface utilisateur 5250



Assurances IBM_PB Gestion simplifiée de sinistre
Application 1 - Menu principal

Vous êtes connecté sous un profil de type CALL CENTER

Choisissez une option, puis appuyez sur ENTREE

1. Déclaration de sinistre
2. Mise à jour de sinistre
3. Coordonnées experts
4. Coordonnées conseillers
5. Messages
6. Spouses

Option : 1

F3=Exit F12=Annuler

Assurances IBM_PB Déclaration de sinistre 29/01/2026
Choix d'un contrat client

Numéro de client : 20
Nom : PARENT
Prénom : Alice
Adresse : 3, place Marceau - 87000 LIMOGES

Liste des contrats

Choisissez un contrat, puis appuyez sur ENTREE

Opt N° contrat	Famille	Titre
X 1010	AUTO	Auto Formule Confort
- 1020	HABITATION	Habitation multirisques Formule Standard

Déclaration du sinistre

Date de déclaration . . . 29.01.2026

Type de sinistre . . . F4=Liste

Date du sinistre . . . 29.01.2026

Description.

F12=Annuler ENTREE=Déclarer le sinistre

Conversion de l'interface utilisateur 5250



Assurances IBM_PB

Gestion simplifiée de sinistre

Application 1 - Menu principal

Profil utilisateur Vous êtes connecté sous un profil de type CALL CENTER

Choisissez une option, puis appuyez sur ENTREE

1. Déclaration de sinistre
2. Mise à jour de sinistre
3. Coordonnées experts
4. Coordonnées conseillers
5. Messages
6. Spouses

F3=Exit F12=Annuler

Assurances IBM_PB

Déclaration de sinistre

Choix d'un contrat client

Numéro de client : 20
Nom : PARENT
Prénom : Alice
Adresse : 3, place Marceau - 87000 LIMOGES

Liste des contrats

Choisissez un contrat, puis appuyez sur ENTREE

N° contrat	Famille	Titre
1010	AUTO	Auto Formule Confort
1020	HABITATION	Habitation multirisques Formule Standard

Valider F12=Annuler

Déclaration du sinistre

Date de déclaration
12/12/2025

Type de sinistre
Incendie

Date du sinistre
01/12/2025

Description
Description

Date de déclaration
/ /mm/aaaa

January 2026

S	M	T	W	Th	F	S
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

ENTREE=Déclarer le sinistre F12=Annuler



C. Développement de programmes et d'applications IBM i avec IBM Bob

Génération de code : Bob peut générer...



- **Du code applicatif**
 - Procédures métier, programmes, applications entières
 - En RPG, COBOL, CL, DDS, SQL, C, C++, Java, PHP, Python, TypeScript...
- **Des composants base de données**
 - Tables, vues, index, procédures, fonctions, triggers...
- **Une interface utilisateur**
 - Ecrans 5250
 - UI modern (react, angular...)
- **De la documentation**
 - Architecture, API, readme...
 - Diagrammes de flux, ERD...
- **Des composants DevOps**
 - Scripts de build
 - Pipelines CI/CD...
- etc. etc. etc.

Génération de procédures



UTIL_SRV.RPGLE •

ASSUR1 > QSRVSRV > UTIL_SRV.RPGLE > ...

Literate Coding Mode is enabled ⓘ

```
26 dcl-proc ENQHSINFF export;
44     reade num_cli sin_cli;
45     total_rep = total_rep + mnt_rep;
46     enddo;
47
48     if total_rep > 0;
49     | enqueue = '0';
50     endif;
51     else;
52     | enqueue = '?';
53     endif;
54
55 end-proc;
56
57 Créer une procédure qui fera le calcul de la prime de renouvellement.
58 Entrées : Ancienneté (années), Nb Sinistres, Prime N-1, Prime Technique N.
59 Logique :
60     Calculer un coefficient d'ajustement sur la Prime Technique :
61     Si Nb Sinistres > 0 : Malus de +15%.
62     Sinon : Bonus fidélité (0% si <3 ans, -5% si 3-5 ans, -10% si >5 ans).
63     Calculer la "Prime Ajustée" = Prime Technique * Coefficient.
64     Appliquer le "Tunnel d'évolution" (Bornes basées sur la Prime N-1) :
65     Plafond Max = Prime N-1 + 20%.
66     Plancher Min = Prime N-1 - 10%.
67     Si Prime Ajustée > Plafond, alors Prime Finale = Plafond.
68     Si Prime Ajustée < Plancher, alors Prime Finale = Plancher.
69     Sinon Prime Finale = Prime Ajustée.
70 Retourne la prime finale.
```

Generate Ctrl+G

Créer une procédure qui fera le calcul de la prime de renouvellement.
Entrées : Ancienneté (années), Nb Sinistres, Prime N-1, Prime Technique N.

Logique :

- Calculer un coefficient d'ajustement sur la Prime Technique :
- Si Nb Sinistres > 0 : Malus de +15%.
- Sinon : Bonus fidélité (0% si <3 ans, -5% si 3-5 ans, -10% si >5 ans).
- Calculer la "Prime Ajustée" = Prime Technique * Coefficient.
- Appliquer le "Tunnel d'évolution" (Bornes basées sur la Prime N-1) :
- Plafond Max = Prime N-1 + 20%.
- Plancher Min = Prime N-1 - 10%.
- Si Prime Ajustée > Plafond, alors Prime Finale = Plafond.
- Si Prime Ajustée < Plancher, alors Prime Finale = Plancher.
- Sinon Prime Finale = Prime Ajustée.

Retourne la prime finale.

```
58 //*****
59 // Procédure : CALCPRIMREN
60 // Description : Calcul de la prime de renouvellement avec application du
61 // coefficient d'ajustement et du tunnel d'évolution
62 // Paramètres :
63 // anciennete : Ancienneté du client en années (entrée)
64 // nb_sinistres : Nombre de sinistres (entrée)
65 // prime_n1 : Prime de l'année N-1 (entrée)
66 // prime_tech_n : Prime technique de l'année N (entrée)
67 // prime_finale : Prime de renouvellement calculée (sortie)
68 // Retour :
69 // prime_finale : Prime finale après application des règles
70 //*****
71 dcl-proc CALCPRIMREN export;
72     dcl-pi *n packed(9:2);
73     anciennete packed(3:0) const;
74     nb_sinistres packed(3:0) const;
75     prime_n1 packed(9:2) const;
76     prime_tech_n packed(9:2) const;
77
78 end-pi;
```

↑ 1/2 ↓ Undo All Ctrl+Shift⌫ Accept All Ctrl+G

Génération de *procédures*

```
75  
79  // Déclaration des variables locales  
80  dcl-s coefficient packed(5:4);  
81  dcl-s prime_ajustee packed(9:2);  
82  dcl-s plafond_max packed(9:2);  
83  dcl-s plancher_min packed(9:2);  
84  dcl-s prime_finale packed(9:2);  
85  
86  // Initialisation  
87  coefficient = 1.0000;  
88  
89  // Calcul du coefficient d'ajustement  
90  if nb_sinistres > 0;  
91  | // Malus de +15%  
92  | coefficient = 1.1500;  
93  else;  
94  | // Bonus fidélité selon l'ancienneté  
95  | if anciennete < 3;  
96  | | // Pas de bonus  
97  | | coefficient = 1.0000;  
98  | elseif anciennete >= 3 and anciennete <= 5;  
99  | | // Bonus de -5%  
100 | | coefficient = 0.9500;
```

```
101  | else;  
102  | | // Bonus de -10%  
103  | | coefficient = 0.9000;  
104  | endif;  
105  endif;  
106  
107  // Calcul de la prime ajustée  
108  prime_ajustee = prime_tech_n * coefficient;  
109  
110  // Calcul des bornes du tunnel d'évolution  
111  plafond_max = prime_n1 * 1.20; // +20%  
112  plancher_min = prime_n1 * 0.90; // -10%  
113  
114  // Application du tunnel d'évolution  
115  if prime_ajustee > plafond_max;  
116  | prime_finale = plafond_max;  
117  elseif prime_ajustee < plancher_min;  
118  | prime_finale = plancher_min;  
119  else;  
120  | prime_finale = prime_ajustee;  
121  endif;  
122  
123  return prime_finale;  
124  
125 end-proc;
```



Génération de *programmes*



```
☰ e2.rpgle  X
BAT25 > QRPGLESRC > ☰ e2.rpgle
1  **free
2  // Objectif : afficher la liste des vols possibles depuis un aéroport de départ
3
4  // Contexte :
5  // - La table VOLS contient la liste des vols (2 colonnes : aéroport de départ (DEPART), aéroport d'arrivée (ARRIVEE))
6  // - L'index VOLSBYDEP sur la table des VOLS a une clé sur la colonne DEPART en ascendant
7  // - La description et les données de la table VOLS ainsi que la description de l'index VOLSBYDEP
8  //   disponibles dans le fichier QSQLSRC/VOLS.SQL
9
10 // Résultat attendu (exemple pour un départ de la ville de New-York )
11 //
12 // Vols au départ de New-York
13 // New York > Paris
14 // New York > Paris > Madrid
15 // New York > Paris > Caire
16 // New York > Paris > Rome
17 // New York > Londres
18 // New York > Londres > Athènes
19 // New York > Londres > Athènes > Nicosie
20 // New York > Los Angeles
21 // New York > Los Angeles > Tokyo
22 // New York > Los Angeles > Tokyo > Hawaii
23
24
25 // Consignes :
26 // 1. Utiliser uniquement des instructions RPG natives (pas de SQL, pas d'appels d'API...)
27 // 2. Les vols pour lesquels un cycle serait détecté ne doivent pas être affichés
28 // 3. Le programme doit s'appeler E2_BOB.RPGL
```

Dans le dossier BAT25/QRPGLESRC Ecrire un programme en RPG Full Free selon les consignes qui sont données dans le source E2.RPGL

🛠 Advanced ⌂



Génération de *programmes*



Todo List Created

- Analyser les consignes et la structure de la table VOLS
- Créer le programme E2_BOB.RPGL en RPG Full Free
- Implémenter l'algorithme de parcours récursif des vols
- Gérer la détection des cycles pour éviter les boucles infinies
- Tester le programme avec l'exemple fourni (New York)

Todo List Updated

- Analyser les consignes et la structure de la table VOLS
- Créer le programme E2_BOB.RPGL en RPG Full Free
- Implémenter l'algorithme de parcours récursif des vols
- Gérer la détection des cycles pour éviter les boucles infinies
- Vérifier et documenter le programme

Génération de *programmes*

Tâche terminée

Programme E2_BOB.RPGLE créé avec succès dans le dossier BAT25/QRPGLESRC.

Caractéristiques du programme :

1. **RPG Full Free** : Utilise la syntaxe moderne ****free** avec uniquement des instructions RPG natives (pas de SQL)
2. **Algorithme récursif** : Implémente un parcours en profondeur (DFS) pour explorer tous les chemins possibles
3. **Détection de cycles** : Vérifie à chaque étape si la destination fait déjà partie du chemin actuel pour éviter les boucles infinies
4. **Backtracking** : Retire les villes du chemin après exploration pour permettre d'autres branches

Fonctionnement :

- Lit le fichier VOLS via l'index VOLSBYDEP (clé sur DEPART)
- Affiche tous les chemins possibles depuis "New York"
- Format de sortie : New York > Paris > Madrid
- Ignore automatiquement les vols créant des cycles (ex: Caire > Paris alors que Paris est déjà dans le chemin)

Le programme respecte toutes les consignes et produira le résultat attendu pour l'exemple fourni.

BAT25 > QRPGLESRC > E2_BOB.RPGLE > ...

```
1  **free
2  // Programme : E2_BOB
3  // Objectif : Afficher la liste des vols possibles depuis un aéroport de départ
4  // Auteur : IBM Bob
5  // Date : 2026-01-27
6
7  ctl-opt dftactgrp(*no) actgrp(*new);
8  |
9  // Déclaration de l'index VOLSBYDEP (fichier logique sur VOLS)
10 // L'index VOLSBYDEP a une clé sur la colonne DEPART
11 dcl-f VOLSBYDEP keyed usage(*input);
12
13 // Structure pour stocker un vol
14 dcl-ds Vol_t qualified template;
15   depart varchar(20);
16   arrivee varchar(20);
17 end-ds;
18
19 // Structure pour le chemin en cours
20 dcl-ds Chemin_t qualified template;
21   villes char(20) dim(50);
22   nbVilles int(10);
23 end-ds;
24
25 // Variables globales
26 dcl-s villeDepart varchar(20);
27 dcl-ds cheminCourant qualified;
28   villes char(20) dim(50);
29   nbVilles int(10);
30 end-ds;
31 dcl-s i int(10);
32 dcl-s j int(10);
33 dcl-s trouve ind;
34
35 // Programme principal
36 villeDepart = 'New York';
37
38 snd-msg *info 'Vols au départ de ' + %trim(villeDepart) %target(*pgmbdy);
39
```

Génération d'applications complètes



Créer une application simple pour IBM i pour gérer une base de livres.

Les livres seront stockés dans la table SQL LIVRES qui contiendra plusieurs colonnes (unique ID, titre, auteur...) et contiendra quelques exemples de livres connus.

La logique RPG qui permettra de connaître le détail d'un livre, de récupérer la liste des livres selon un critère de recherche, de mettre à jour ou de supprimer un livre

sera définie dans des procédures RPGLE d'un programme de service et les accès à la base se feront en SQL.

Ce programme de service sera ajouté à un répertoire de liage.

Créer quelques procédures de test unitaires pour tester les procédures du programme de service, en utilisant le framework RPGUnit.

L'interface utilisateur sera composée de formats 5250 dans un DSPF.

Le programme qui pilotera tout cela sera en RPGLE et proposera un menu qui permettra d'appeler les différentes fonctions (détail d'un livre, liste des livres avec des possibilités de tri, ajout / mise à jour / suppression d'un livre) et une option pour générer des rapports simples.

Tous les sources de cette application seront stockés dans le répertoire LIVRES sous Projets Bob.

Améliorer la requête avec un contexte supplémentaire

Advanced ^



Bob va améliorer / compléter le prompt !

Génération d'applications complètes



Développer une application complète de gestion de bibliothèque pour IBM i en utilisant les technologies natives de la plateforme.

L'application doit inclure les composants suivants :

Base de données SQL : Créer une table LIVRES avec les colonnes ID (clé primaire auto-incrémentée), TITRE (VARCHAR 100), AUTEUR (VARCHAR 100), ISBN (VARCHAR 20), ANNEE_PUBLICATION (INTEGER), EDITEUR (VARCHAR 100), GENRE (VARCHAR 50), NOMBRE_PAGES (INTEGER), DATE_AJOUT (TIMESTAMP), STATUT (CHAR 1 pour disponible/emprunté). Peupler la table avec au moins 15 livres célèbres couvrant différents genres littéraires.

Programme de service RPGLE : Développer un module LIVRESSRV contenant des procédures exportées avec gestion complète des erreurs et codes retour. Inclure les procédures suivantes utilisant SQL embarqué : ObtenirLivre (par ID), RechercherLivres (avec paramètres optionnels pour titre, auteur, genre), AjouterLivre, ModifierLivre, SupprimerLivre, ObtenirStatistiques (nombre total, par genre, par auteur). Chaque procédure doit retourner des structures de données appropriées et gérer les exceptions SQL. Créer le binding directory LIVRESBD et y ajouter le programme de service.

Tests unitaires RPGUnit : Créer le module de test TESTLIVRES avec des cas de test couvrant tous les scénarios : test d'ajout avec données valides et invalides, test de recherche avec différents critères, test de modification incluant les cas limites, test de suppression avec vérification d'intégrité, test de récupération de détails pour livres existants et inexistant, tests de performance pour les recherches sur grand volume.

Interface utilisateur 5250 : Concevoir le fichier d'affichage LIVRESDF avec les formats suivants : MENU (écran principal avec options numérotées et touches fonction), DETAIL (affichage complet des informations d'un livre avec navigation), LISTE (sous-fichier paginé avec en-têtes de colonnes, options de tri par titre/auteur/année, indicateurs de sélection), RECHERCHE (écran de saisie des critères avec validation), EDITION (écran de saisie/modification avec contrôles de champs obligatoires et formats), CONFIRMATION (pour suppressions et actions critiques), RAPPORT (sélection des paramètres de rapport).

Génération d'applications complètes



Programme principal RPGLE : Développer LIVRESPGM qui implémente une boucle de traitement des commandes utilisateur avec gestion d'état. Implémenter la navigation entre écrans avec historique, la validation des entrées utilisateur côté serveur, la gestion des messages d'erreur et de confirmation dans la zone de message, le traitement des touches fonction (F3=Quitter, F5=Actualiser, F6=Ajouter, F12=Retour), la gestion de session pour mémoriser les derniers critères de recherche, le tri dynamique des listes sans rechargement complet.

Module de génération de rapports : Créer des procédures pour générer des rapports imprimables ou exportables : liste complète des livres avec formatage, statistiques par genre et auteur avec graphiques ASCII, livres ajoutés dans une période donnée, rapport d'inventaire avec totaux. Les rapports doivent être générés en fichiers spool avec en-têtes et pieds de page appropriés.

Structure du projet : Organiser tous les sources dans le répertoire LIVRES sous Projets Bob avec la structure suivante : QDDSSRC pour les fichiers physiques et d'affichage, QRPGLESRC pour les programmes et modules RPG, QSQLSRC pour les scripts SQL de création et d'initialisation, QCMDSRC pour les commandes CL de compilation, QBNDSRC pour les binding directories, QTXTSRC pour la documentation. Inclure un fichier README avec instructions de compilation et déploiement, un script de build complet automatisant la création de tous les objets dans l'ordre correct, et des commentaires détaillés dans tous les sources.

Exigences techniques supplémentaires : Utiliser les spécifications RPG modernes (free-format complet), implémenter la gestion des erreurs avec blocs monitor/on-error, utiliser des indicateurs nommés plutôt que numérotés, appliquer les conventions de nommage cohérentes, optimiser les requêtes SQL avec indexes appropriés, gérer correctement la fermeture des curseurs et libération des ressources, implémenter le logging des opérations critiques, assurer la compatibilité avec IBM i 7.3 et versions ultérieures.

Génération d'applications complètes



Je vais développer une application complète de gestion de bibliothèque pour IBM i. C'est un projet complexe qui nécessite une planification détaillée.

✓— Todo List Created

- Créer la structure de répertoires du projet LIVRES
- Créer la base de données SQL (table LIVRES avec contraintes)
- Peupler la table avec 15 livres célèbres
- Développer le programme de service LIVRESSRV avec procédures SQL
- Créer le binding directory LIVRESBD
- Développer les tests unitaires RPGUnit (TESTLIVRES)
- Concevoir le fichier d'affichage LIVRESDF avec tous les formats
- Développer le programme principal LIVRESPGM
- Créer le module de génération de rapports
- Créer les scripts de compilation CL
- Créer le script de build automatisé
- Rédiger la documentation README

Génération d'applications complètes



Tâche terminée

Application complète de gestion de bibliothèque pour IBM i développée avec succès. Le projet inclut :

Composants créés :

1. Base de données SQL (`LIVRES.TABLE`)

- Table LIVRES avec clé primaire auto-incrémentée
- Contraintes de validation (statut, année, pages)
- Index optimisés pour les recherches
- 15 livres célèbres pré-chargés (`LIVRES_DATA.SQL`)

2. Programme de service (`LIVRESSRV.RPGLE`)

- 6 procédures exportées avec SQL embarqué
- Gestion complète des erreurs avec monitor/on-error
- Structures de données typées
- Binding directory (`LIVRESBD.BND`)

3. Tests unitaires RPGUnit (`TESTLIVRES.RPGLE`)

- 14 cas de test couvrant tous les scénarios
- Tests de validation, performance et cas limites

4. Interface utilisateur 5250 (`LIVRESDF.DSPF`)

- 8 formats d'écran (menu, détail, liste, recherche, édition, confirmation, rapport, statistiques)
- Sous-fichier paginé avec tri dynamique
- Validation des champs avec indicateurs d'erreur

5. Programme principal (`LIVRESPGM.RPGLE`)

- Navigation complète entre écrans
- Gestion d'état et mémorisation des critères
- Touches fonction (F3, F5, F6, F12)
- 783 lignes de code RPG moderne

6. Module de rapports (`LIVRESRPT.RPGLE`)

- 4 types de rapports avec graphiques ASCII
- Statistiques par genre et auteur
- Rapports d'inventaire et par période

7. Scripts de compilation CL

- `CRTLIVRES.CLLE` - Service
- `CRTLIVPGM.CLLE` - Programme principal
- `CRTLIVRPT.CLLE` - Rapports
- `BUILD.CLLE` - Build automatisé complet

Génération d'applications complètes



8. Documentation (README.md)

- Instructions d'installation et utilisation
- Architecture détaillée
- Guide de dépannage
- 502 lignes de documentation complète

Caractéristiques techniques :

- RPG Free Format complet (**FREE)
- SQL embarqué avec gestion d'erreurs
- Indicateurs nommés
- Conventions de nommage cohérentes
- Compatible IBM i 7.3+
- Code entièrement commenté en français

Pour déployer : CALL PGM(LIVRES/BUILD)

Pour lancer : CALL PGM(LIVRES/LIVRESPGM)

Génération d'applications complètes



README.md Preview README.md X



Application de Gestion de Bibliothèque pour IBM i

Description

Application complète de gestion de bibliothèque développée en RPG Free Format pour IBM i, utilisant les technologies natives de la plateforme (SQL embauché). Cette application permet de gérer un catalogue de livres avec des fonctionnalités complètes de recherche, ajout, modification, suppression et génération d'rapports.

Fonctionnalités

Gestion des livres

- Recherche avancée** : Recherche par titre, auteur, genre, statut, année de publication
- Ajout de livres** : Saisie complète avec validation des données
- Modification** : Mise à jour des informations d'un livre existant
- Suppression** : Suppression avec confirmation
- Affichage détaillé** : Consultation des informations complètes d'un livre

Interface utilisateur

- Menu principal** : Navigation intuitive entre les différentes fonctions
- Écrans de saisie** : Validation en temps réel avec messages d'erreur
- Liste paginée** : Affichage des résultats avec sous-fichier
- Tri dynamique** : Tri par titre, auteur, genre ou date
- Touches fonction** : F3=Quitter, F5=Actualiser, F6=Ajouter, F12=Retour

Menu principal

GESTION DE BIBLIOTHEQUE

Menu Principal

-
1. Rechercher des livres
 2. Ajouter un livre
 3. Modifier un livre
 4. Supprimer un livre
 5. Afficher les statistiques
 6. Générer un rapport
 7. Liste complète des livres
 0. Quitter

Sélection : _

F3=Quitter F5=Actualiser

Navigation

- F3** : Quitter l'application
- F5** : Actualiser l'écran courant
- F6** : Ajouter un nouveau livre (dans les listes)
- F12** : Retour à l'écran précédent
- Entrée** : Valider la saisie

Génération d'applications complètes



```
28/01/26          GESTION DE BIBLIOTHEQUE          18:09:44
Menu Principal
1. Rechercher des livres
2. Ajouter un livre
3. Modifier un livre
4. Supprimer un livre
5. Afficher les statistiques
6. Générer un rapport
7. Liste complète des livres
0. Quitter

Sélection : _

F3=Quitter  F5=Actualiser
Bienvenue dans le système de gestion de bibliothèque
```

```
28/01/26          STATISTIQUES          18:21:32
Statistiques de la bibliothèque
Nombre total de livres : 15
Livres disponibles : 12
Livres empruntés : 3

Répartition par genre :
Biographie 1
Conte philosophique 1
Fantasy 2
Policier 1
Roman classique 1
Roman contemporain 2
Roman d'aventure 1
Roman philosophique 1
Roman réaliste 1
Science-fiction 3

F3=Quitter  F12=Retour
Statistiques de la bibliothèque
```

IBM i

continuous innovation
continuous integration



D. Pour aller plus loin

Bilan des tests effectués



Fonctionnalité	Avis	Remarques
Explication de code	****	
Rétro-documentation fonctionnelle	***	Dépend de la qualité du code (nom des variables, commentaires)
Analyse de l'architecture des applications	****	Les + : <ul style="list-style-type: none">• Découpage en couches• Diagrammes des flux
Génération de documentation	****	

Bilan des tests effectués



Fonctionnalité	Avis	Remarques
Amélioration / optimisation de code	***	A retester avec des règles personnalisées A retester après la GA (fonctions spécifiques IBM i)
Conversion RPG fixe → RPG Full Free	**	Idem
Conversion DDS → SQL	****	Idem
Conversion RLA → SQL	***	Idem
Refactorisation en procédures	***	Idem
Extraction de logique métier	***	Idem
Conversion des écrans 5250	***	Idem

Bilan des tests effectués



Fonctionnalité	Avis	Remarques
Génération de code / procédures	****	A retester avec des règles personnalisées A retester après la GA (fonctions spécifiques IBM i)
Génération de programmes	***	Idem
Génération d'applications	**	Idem

Bilan des tests effectués



- Bob ne remplace pas les développeurs, ni les architectes applicatifs
- Bob ne va pas magiquement moderniser des décades de code
- Bob accélère la compréhension du code, elle ne la remplace pas
- Bob permet d'avoir une vision claire des applications
- Bob permet d'avoir une documentation structurée des applications
- Vous êtes responsables du code
- Vous devez approuver les modifications
- Vous devez comprendre les règles métier
- Vous êtes le garant de l'exactitude de ce qui est généré

IBM Bob – Tarification



- Pas encore connue
- A l'usage
- Packages spécifiques aux plateformes d'entreprise
 - Package spécifique IBM i

IBM Bob – Comment le tester ?



- Il faut s'inscrire à la Technology Preview
 - <https://www.ibm.com/products/bob#Form>

Don't miss out! Sign up for the tech preview now.

Early adopters are already shipping faster with Bob. Save time, skip the grunt work, and focus on what matters.

Business Contact Information

First Name

Last Name

Work Email

Country

IBM Bob – Comment IBM peut vous aider ?



- **Pour un POC**
 - Benoit Marolleau
 - *AI Engineer / Senior Solution Architect* - IBM Client Engineering Europe
 - benoit.marolleau@fr.ibm.com
 - Ludovic Gasc
 - *Open Source and AI Solutions Architect on IBM Power* – IBM Benelux
 - ludovic.gasc@be.ibm.com
- **Pour un POC, une formation ou une prestation de mise en œuvre**
 - Philippe Bourgeois
 - *Consultant IBM i* - IBM Expert Labs
 - pbourgeois@fr.ibm.com

IBM Bob – Comment se préparer ?



- Pour vous **former** aux technologies IBM i :
 - **Workshops** de 4 heures, sur site ou à distance
 - IBM Bob (*x workshops*) – *A venir*
 - VS Code for i (*3 workshops*)
 - RPG Full Free et nouveautés RPG (*1 workshop*)
 - Services Web (*3 workshops*)
 - Nouveautés SQL (*1 workshop*)
 - Nouveautés IBM i 7.5 / 7.6 (*4 workshops*)
 - SQL avancé (*3 workshops*)
 - RDi (*2 workshops*)
 - <https://ibm.box.com/s/yupdrjzmq5tudl7yfje9p8kqu6g98tpr>
- **IBM i Skills Connect**
 - Une offre pour que vos équipes IBM i soient informées, formées et accompagnées aux dernières technologies IBM i
 - <https://ibm.box.com/s/yw23djm9wecib988xh39qfwy7886kxy>
- **IBM i Modernization Connect**
 - Une offre d'assistance à la modernisation de vos applications IBM i
 - <https://ibm.box.com/s/y8ctwljvh3m0qszk1hh2szas3cv9xg0d>
- **Philippe Bourgeois**
 - *Consultant IBM i - IBM Expert Labs*
 - pbourgeois@fr.ibm.com

IBM Bob – Documentation



- Sites Web
 - <https://www.ibm.com/products/bob>
- Documentation
 - <https://pages.github.ibm.com/code-assistant/bob-docs/>
 - https://learn.common.org/products/deep-dive-into-project-bob-for-rpg-and-more#tab-product_tab_overview
- Vidéos
 - <https://www.youtube.com/channel/UC-dkbPjzN2bh-k-V4rZQppQ>

Prochain webinar COMMON France



■ IA agentique et IBM i

- 20 février 2026
- COMMON France / IBM
- Pour les adhérents au club COMMON France
- <https://common-france.assoconnect.com/collect/description/661242-d-ia-agentique-et-ibm-i>



Congrès COMMON Europe à Lyon



- SAVE THE DATE

**COMMON EUROPE
CONGRESS 2026**

14 - 17 June
Lyon, France

The largest conference in Europe
for solutions around IBM Power (IBM i, AIX, Linux) & IBM Storage

common
EUROPE

www.common-europe.org

common
FRANCE



MERCI