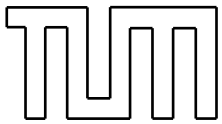
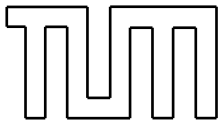


FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master Praktikum: Enterprise Software Engineering am Beispiel von SAP

Project Documentation for the Application Extension Methodology Project with SAP

Kadir Kubilay Erdem
Jiangang Huang
Yunus Emre Konak
Lea Karin Siekmann
Achilleas Tsakpinis



FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master Praktikum: Enterprise Software Engineering am Beispiel von SAP

Projektdokumentation für das Application Extension Methodology Projekt mit SAP

Project Documentation for the Application Extension Methodology Project with SAP

Author:	Group SAP
Supervisor:	Prof. Dr. Helmut Krcmar
Advisor:	Philipp Landler
Submission Date:	05.02.2025

I assure the single handed composition of this project documentation is only supported by declared resources

Garching b. München, den 05.02.2025	Kadir Kubilay Erdem
Ort, Datum	Unterschrift
Garching b. München, den 05.02.2025	Jiangang Huang
Ort, Datum	Unterschrift
Garching b. München, den 05.02.2025	Yunus Emre Konak
Ort, Datum	Unterschrift
Garching b. München, den 05.02.2025	Lea Karin Siekmann
Ort, Datum	Unterschrift
Garching b. München, den 05.02.2025	Achilleas Tsakpinis
Ort, Datum	Unterschrift

Abstract

As organizations grow and evolve, managing SAP system extensions effectively has become a critical focus. Companies seek to strike a balance between flexibility and maintainability, especially given SAP's extensive customization capabilities. However, traditional approaches—such as embedding extensions directly into the core system—often result in high maintenance costs, complex upgrades, and scalability challenges. To address these issues, SAP introduced the Clean Core methodology, which emphasizes separating core functionalities from custom extensions. This approach enhances system stability and long-term compatibility. Despite its benefits, many businesses have found it difficult to implement due to the lack of structured tools for evaluating and managing extensions.

Our application, developed in collaboration with SAP, provides a streamlined solution to this challenge. Unlike traditional spreadsheet-based methods, which can be inconsistent and difficult to manage collaboratively, our tool offers a centralized platform with built-in validation, structured data management, and clear graphical representations of extension strategies. By standardizing the evaluation process, our solution helps organizations make informed decisions and maintain greater control over their SAP extensions.

With a structured and intuitive approach, our tool enables businesses to reduce complexity, improve decision-making, and ensure their extensions remain scalable and maintainable. By aligning with SAP's extensibility framework, companies can confidently future-proof their systems while adhering to Figure 1 practices. This solution empowers organizations to navigate SAP extension management with clarity and efficiency, ultimately optimizing their system's performance and sustainability

Table of Contents

List of Figures

List of Abbreviations

ERP	Enterprise Resource Planning
UI	User Interface
UX	User Experience
API	Application Programming Interface
CDS	Core Data Services
RAP	RESTful Application Programming
JSON	JavaScript Object Notation
UI5	SAP UI5 Framework
CSV	Comma-Separated Values
RFC	Remote Function Call
SQL	Structured Query Language
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
CRUD	Create, Read, Update, Delete
JWT	JSON Web Token
IDE	Integrated Development Environment
CD	Continuous Deployment
MVC	Model-View-Controller
ETL	Extract, Transform, Load
BTP	Business Technology Platform

1. Motivation and Situation

In today's fast-paced business world, companies rely on flexible IT systems to keep up with changing demands. SAP's extensibility options make it possible to tailor ERP systems to unique business needs. However, as organizations add custom extensions, they often run into challenges such as maintaining system stability, ensuring smooth updates, and avoiding long-term technical debt.

A common issue is that many SAP customers have historically embedded extensions directly into the core system. While this approach offers short-term convenience, it leads to higher maintenance costs, increased complexity, and potential conflicts with future SAP updates. A tightly coupled system makes upgrades more difficult and can restrict access to new SAP innovations, such as cloud services and AI-powered tools.

To tackle these challenges, SAP introduced the Clean Core paradigm, which encourages a clear separation between core functionalities and custom extensions. This approach reduces complexity, improves system maintainability, and ensures long-term scalability without limiting flexibility. To help companies implement this strategy, SAP also developed the Application Extension Methodology—a structured framework for evaluating extension strategies that align with business goals and available technologies. However, despite this guidance, many organizations struggled with putting the methodology into practice due to limitations in the available tools.

Until now, companies following the Application Extension Methodology primarily relied on Excel spreadsheets provided by SAP. While this approach offered a basic structure for evaluating extensions, it also introduced several inefficiencies. Without automated validation, inconsistencies and errors could easily occur. Collaboration was another challenge, as spreadsheets had to be manually updated and shared among different stakeholders, often leading to version control issues. Additionally, the lack of visual representation made it difficult to compare and analyze extension strategies effectively.

Because of these challenges, applying the Clean Core methodology was often cumbersome and time-consuming. To make this process more efficient, we collaborated with SAP to develop a dedicated application that enhances the Application Extension Methodology, particularly in defining and evaluating extension possibilities. This new tool moves beyond Excel by offering a centralized and interactive platform, allowing organizations to manage and share extension strategies in a more structured way. By incorporating validation mechanisms, it helps maintain consistency and accuracy in extension assessments. Furthermore, the application introduces graphical representations, making it easier to visualize and compare different extension strategies.

With this solution, we aim to simplify and streamline the evaluation process, ensuring that businesses can make well-informed decisions about their SAP extensions. By adopting this new approach, organizations can take full advantage of SAP's extensibility options while maintaining a structured, scalable, and future-ready ERP system.

2. Requirements Analysis

During the Requirements Engineering phase of our project, we identified a total of 26 user stories in collaboration with our project partner, which were subsequently grouped into six epics. These user stories were designed to address the needs and workflows of the intended users of the application while ensuring a smooth transition from current practices.

To ensure a user-centered approach, we developed two personas, which are referenced throughout the user stories. These personas represent the primary users of the application, all of whom are associated with an organization utilizing SAP technology and requiring AEM strategy management. The personas are as follows:

> *Power User*: Responsible for defining strategies and managing key configurations within the system. This persona includes roles such as IT Managers and Solution Architects.
>
> *View User*: Primarily relies on the application for accessing crucial information without the ability to edit any of it. This persona includes roles such as Consultants and Developers.

The following section provides a detailed overview of the identified epics and user stories.

a) Epics

1. Managing Multiple Projects
2. External Data Management
3. User Management
4. Configure Project Data
5. Change History
6. Visual Overview

b) User Stories

1.1. As a power user of this app, I want to save the data for multiple projects within one application, so that all my data is centralized.

1.2. As a power user of this app, I want to create a new project by duplicating an existing one from the app, so that I can save time in set-up.

1.3. As a view user of this app, I want to navigate between multiple projects datasets, so that I can efficiently manage all projects in one place.

1.4. As a power user of this app, I want to create a new version for a project, so that major updates are more realizable for stakeholders.

2.1. As a power user of this app, I want to create a new project when I am importing the Excel, so that right project creation can be done easily.

2.2. As a power user of this app, I want to create a draft project even if my Excel has faulty data, so that I can see the inconsistency from the app.

2.3. As a power user of this app, I want to update existing projects when I am importing an Excel, so that external changes are reflected.

2.4. As a view user of this app, I want to export projects to Excel in a desired format, so that I can easily share them with our partners or stakeholders.

2.5. As a view user of this app, I want to export single extension task graphical charts, so that visuals can be shared with stakeholders.

3.1. As a power user of this app, I want to ensure that viewers have read-only access, so that unauthorized modifications are prevented.

3.2. As a power user of this app, I want to have all the functionalities of a normal user, so that I do not have to switch accounts.

4.1. As a power user of this app, I want to edit my projects directly in the app, so that I do not need to deal with Excel for updates.

4.2. As a power user of this app, I want to update the task hierarchy within my projects from the app, so that I can adapt the project to changing needs.

4.3. As a power user of this app, I want to define a parent-child relationship between building blocks, so that this relationship can be visually represented.

4.4. As a power user of this app, I want to assign a traffic light color to sequences representing the cleanness level of my projects, so that I can understand faster.

4.5. As a power user, I want to save my project as a draft, so that I can work on updates while keeping an original version fully operational for ongoing activities.

4.6. As a view user of this app, I want to view a detailed page for a Building Block, so that all related information is displayed clearly.

4.7. As a power user of this app, I want to customize the project with my own fields, so that I can also process my custom information on the app.

4.8. As a power user of this app, I want to manage screen variants, to provide other users convenience.

5.1. As a view user of this app, I want to view all changes regarding projects, so that I can track the history of my projects.

5.2. As a view user of this app, I want to track project revision, so that I can monitor different versions of my projects.

5.3. As a power user of this app, I want to archive custom fields without losing the history, so that I can manage the project changes effectively.

6.1. As a view user of this app, I want to use comprehensive filters, so that I can find relevant information more quickly and accurately.

6.2. As a power user of this app, I want to customize traffic light colors, so that I can have an instant understanding of solution cleanness.

6.3. As a view user of this app, I want to view a graphical chart for an extension task based on provided data, so that I can gain a clearer overview more quickly.

6.4. As a power user, I want my changes to the task relationship to be validated before being stored, so that I have a correct hierarchical sequence.

3. Approach

a) Agile roadmap

The following figure presents the finalized project roadmap, outlining the development process. It was maintained in Jira and regularly updated throughout the project to track progress and reflect adjustments, to ensure alignment with project goals and between team members.

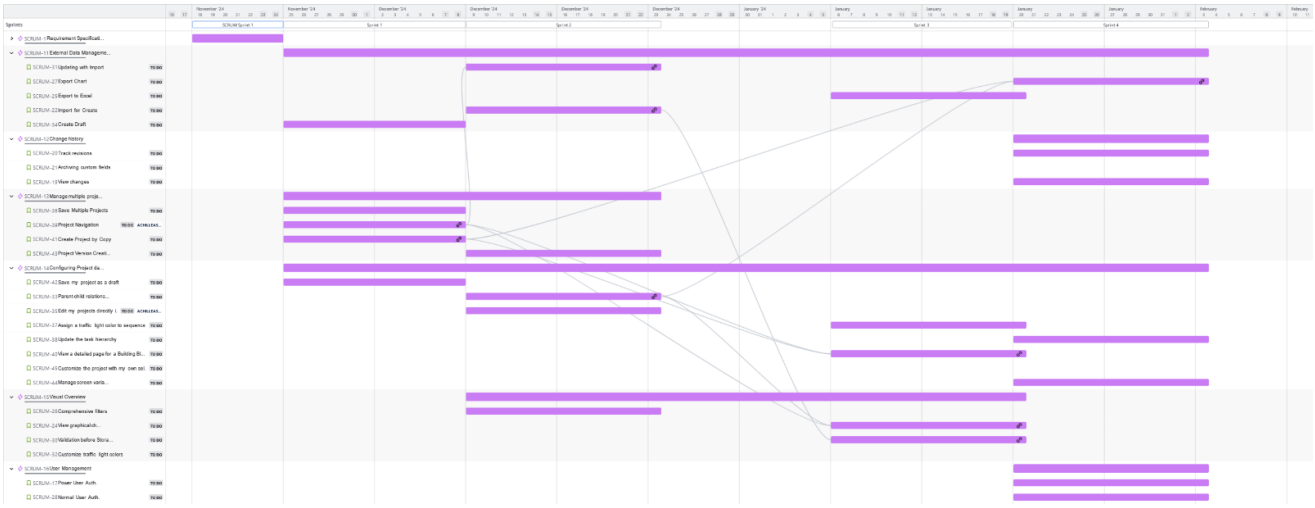


Figure 1: Agile Roadmap

b) Software development

For our project, we adopted key elements of the Scrum framework as it provided a structured yet flexible approach to software development. It enabled us to work independently while maintaining a strong focus on customer needs through regular feedback sessions.

Our development cycle was structured into four sprints, each lasting two weeks. This time frame was chosen as it balanced rapid iteration and allowing enough time to implement meaningful progress within each sprint. Additionally, we dedicated a final sprint solely for project documentation and final presentation preparation. We managed our backlog in the form of user stories. These were maintained in Jira, which provided transparency for all team members and our project partner and offered an intuitive way to visualize our agile roadmap.

To ensure our solution aligned with customer expectations, we established a feedback loop with our project partner through weekly meetings. During these meetings we presented our progress and gathered feedback for further development. In addition to the customer feedback meetings, we held internal team meetings as well. Since scheduling daily stand-ups was difficult due to our varying university commitments, we adopted a more flexible approach by meeting three times a week. This allowed us to stay coordinated while giving enough freedom to the team members to manage all commitments in their own time.

Another key deviation from the classic Scrum methodology was our approach to team roles. As all of us were developers, we did not have an independent product owner or Scrum master available. In this case, we decided to share the responsibilities among all team members, based on availability, creating a shared sense of ownership for the project.

Given that the technology stack was new to all of us, we utilized pair programming techniques. By working in pairs, team members were able to share knowledge, especially from the different learnings of the individual assignment, and exchange ideas on how to implement new topics. This approach not only helped us develop a deeper understanding of the technology but also led to improved team spirit and code quality. Additionally, to further enhance our code quality and maintainability, we adopted a peer review process. This ensured that all the code was reviewed by another team member before being accepted, allowing for early detection of potential issues.

For version control and remote collaboration, we used GitLab for our SAPUI5 applications and ABAPGit for our ABAP code. These platforms provided robust versioning capabilities, making it easy to track changes and collaborate efficiently. Since our application was a completely new development intended as an open, freely available codebase for customers to clone after finishing, we did not have to simultaneously manage a stable production version for users. Therefore, we decided to follow the Trunk-Based Development strategy for git, saving a large overhead of branches and integration allowing for a more direct and more time efficient developing process.

c) Software testing methodology

To ensure robust testing of our application, we employed a combination of methods that varied in both format and participants. Our primary focus was on validating the application against user requirements while also identifying potential customized edge cases.

Internal Testing

The first phase of testing was conducted during the implementation of user stories by the respective team members responsible for their development. This phase involved requirements-based testing, ensuring that each user story met the predefined specifications. To achieve comprehensive test coverage, we not only validated common inputs but also employed fail-based testing to assess the application's behavior in edge cases and unexpected scenarios.

Acceptance Testing

Following internal testing, we conducted acceptance tests in collaboration with our project partner, Lukas. As a representative of both our project partner and the end users, he provided valuable insights into user expectations and usability concerns. His perspective helped refine the application by identifying areas that required improvements from an end-user standpoint.

Unbiased Testing (“Monkey Testing”)

To further ensure that no critical issues were overlooked beyond the defined user stories, we finally carried out a monkey testing phase. In this phase, we provided limited access to the appplication to people who had no prior involvement in the development process. Without guidance or predefined test cases but all necessary data in the form of a filled in example Excel sheet from SAP, they interacted freely with the application, simulating real-world usage by third-party users. This approach helped to uncover unforeseen issues and concerns that structured testing might not have identified.

By integrating these testing methodologies, we ensured a thorough evaluation of our application, addressing functionality, user experience, and robustness in edge cases.

4. Software architecture and data models

a) Software architecture

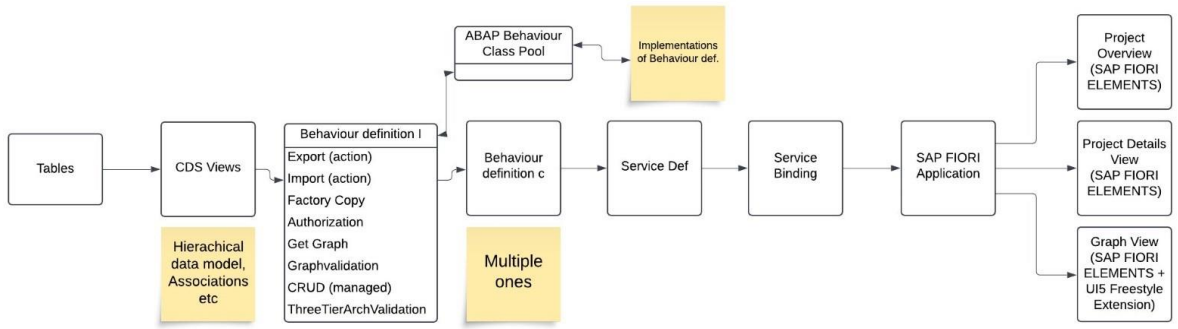


Figure 2: Software Architecture of the Application

b) Semantic data model and technical data model

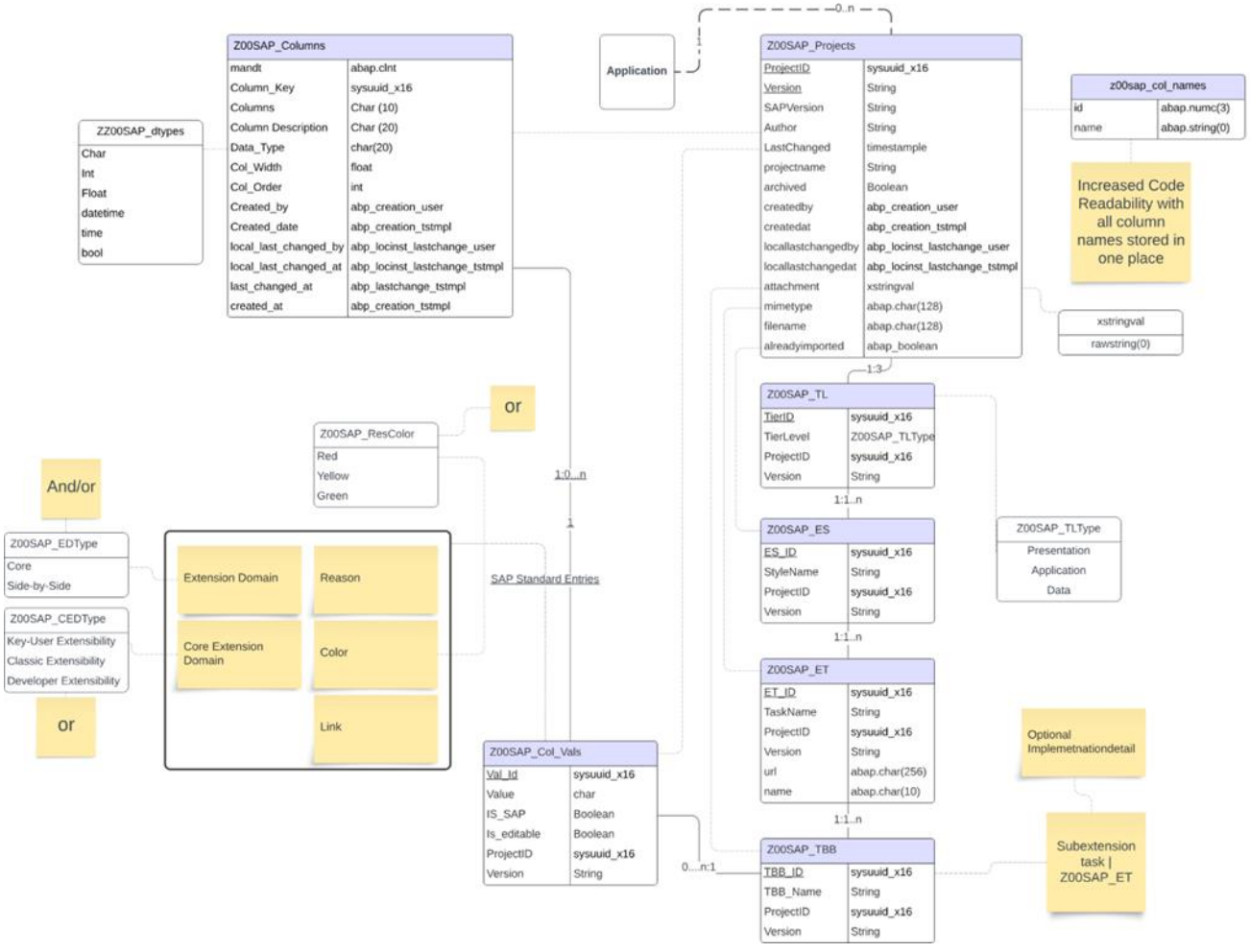


Figure 3: Semantic Data Model and Technical Data Model

5. Implementation

a) SAPUI5 Application

Main application: SAP Fiori List Report Floorplan

The core of our application was developed using the SAP Fiori List Report floorplan. With a project overview on the first level and a large number of detailed extension tasks structured in a table format on the second level, this floorplan optimally supported the visualization. Using suitable annotations in the CDS views and a metadata extension file, the Fiori Page generator automatically generated the application.

However, certain functionalities could not be automatically generated, requiring us to implement additional extensions and enhancements manually. These include:

- **Removal of routing from table on project details page**

  Reasoning: default routing from the second-level table to a more detailed third-level content view for each of the table items not required by our project partner,

  Implementation Details: removal of routing from mainfest.json file.

- **Buttons for navigation to log overview**

  Functionality: Custom actions which let the user navigate from Project Overview page to *logging pages*

  Implementation Details: actions defined via SAP Fiori Page Editor in VS Code and added to the Project overview page, call to generated URL via controller Button_Handlers in ext/controller.

Extension for graph visualization: SAPUI5 Freestyle

A key feature of the project is the structured and intuitive visualization of extension task ratings in the form of graphs. As this goes beyond the capabilities that SAP Fiori floorplans offer, the decision was made to implement this component as a SAPUI5 freestyle application. The graph visualization captures and structures the relationship between technical building blocks, presenting them in a manner that is both clear and ultimately usable as a decision guidance in subsequent development decisions.

For this the view-controller logic was utilized within the application. In addition to the main page, additional table views were generated for project and details to facilitate navigation within the UI5 freestyle application.

The following section provides an overview of the implementation undertaken to achieve this:

Implemented Views

1. **Project Overview View**

- *File*: `Project_Overview.view.xml`
- *Consumed Data Definition*: `Z00SAP_ProjectOverview_C`
- *Standard table fields:*
    - Project ID
    - Version
    - SAP Version
    - Author
    - Project Name

2. **Project Details View**

- *File*: `pro_det.view.xml`
- *Consumed Data Definition*: `Z00SAP_Projectdetails_C_Less`
- *Standard table fields:*
    - Version
    - Project ID
    - Extension Task ID
    - Extension Task Name
    - Project Name

3. **Task Details and Chart View**

- *File:* `TaskDetails.view.xml`
- *Consumed Data Definition:* `Z00SAP_Projectdetails_C_Less`
- *Data stored in:* JSON format (filteredResults)
- *Filtering criteria in controller:*
    - Project ID
    - Version
    - Extension Task ID

This view further consists of three tabs:

3.1.Task Details Table View Tab

- Displays processed data in a structured table format with the following fields:
    - Extension Task ID
    - Project ID
    - Version

- Hierarchy
- Task Name
- Parent Node
- Level
- TBB Name

3.2.Process Flow Chart Tab

- Displays the generated process flow
- Includes an export button to save the process flow as `.jpeg` file

3.3.Node Graph View Tab

- Displays the generated node graph
- Includes an export button to save the graph chart as a `.jpeg` file

Implemented Application Controllers

1. Base Controller

- *File*: `BaseController.js`
- *Functionality*: Serves as the main controller from which all other controllers inherit, all core functions and methods are implemented in this controller
- *Main Functions Used in Other Controllers:*

1.1. `getRouter()`: Used to navigate between pages
1.2. `uuid_encoder()`: Encodes UUIDs to simplify URL generation
1.3. `uuid_decoder()`: Decodes the encoded UUID after URL generation

2. Project Overview Controller

3. *File*: `Project_Overview.controller.js`
4. *Functionality*: Manages the project overview page
5. *Implemented Methods:*
    1. `OnDeletePress()`: Deletes the selected project using the row button in the table
    2. `OnProjectSelect()`: Navigates to the Project Details Page when a row is clicked

3. Project Details Controller

1. *File*: `pro_det.controller.js`
2. *Functionality*: Manages the project details page
3. *Implemented Methods:*

3.1. `OnRouteMatched()`: Filters project details based on the selected project and version from the Project Overview Page
3.2. `OnLineItemPress()`: Navigates to the Task Details View when a row in the table is clicked

4. Task Details & Chart View Controller

- *File*: `TaskDetails.controller.js`
- *Functionality*: The most complex controller responsible for data processing and visualization generation.
- *Implemented Key Methods:*

    - `OnRouteMatched()`:
        - Generates data based on user selection from the Project Details Page
        - Decodes parameters to simplify URL usage.
        - Calls `fetchBackendData()` with the decoded parameters

    - `fetchBackendData()`:
        - Creates filter constants (`aBackendFilters`) with Project ID, Version, and ET ID.
        - Applies filters to the `Z00SAP_Projectdetails_C_Less` data definition.
        - Handles errors and forwards the filtered data to `ProcessData()`

    - `ProcessData()`:
        - Manages core data processing by calling various sub-methods
        - Handles exceptions if no data is provided
        - Creates the root node using `CreateRootNode()`
        - Calls `enrichData()` to finalize hierarchical relationships
        - Generates enrichedData, which is used for table binding, process flow, and network graph generation

    - `CreateRootNode()`:
        - Generates a root node with the following attributes:
        - `NodeId`: Always 1
        - `TaskName`: Derived from the clicked row
        - `Hierarchy`: Always "root"
        - `ParentNode`: Always null
        - `TBBName`: Same as TaskName

    - `enrichData()`:
        - Starts with the root node and assigns unique IDs to each entry
        - Uses regex operations to determine parent-child relationships

    - `bindDataToTable()`:

        - Converts `enrichedData` into a JSON model for table consumption

    - `generateProcessFlow()`:
        - Iterates through `enrichedData` to generate process flow
        - Defines lanes based on parent node IDs
        - Assigns status and color based on color information
        - Generates nodes and lanes, which are stored in `processFlowNodes` and `process` constants
        - Creates a JSON model and sets it to the view

    - `onExportAsImage2()`:
        - Exports Network Graph View to a `.jpeg` file.
        - Uses `dom-to-image.min` to export the selected HTML element

    - `onExportAsImage()`:
        - Exports Process Flow View to a `.jpeg` file.
        - Works similarly to `onExportAsImage2()` but uses `html2canvas` instead of `dom-to-image.min`

    - `generateNetworkGraph()`:
        - Generates a network graph with enriched data
        - Defines two constants: `nodes` (for graphical elements) and `lines` (for connections)
        - Iterates through enriched data to create required nodes and connections
        - Assigns color based on status
        - Adjusts node shapes based on text length
        - Ensures the root node appears at the top with no direct hierarchy constraints

Navigation Configuration

Navigation between pages is implemented within the `manifest.json` file. The defined routes include:

- `RouteProjectDetails`: Navigates from the Project Overview to the Project Details Page
- `RouteTaskDetaisls`: Navigates from the Project Details Page to the Task Details View

Utilized Libraries

The following libraries were used in the project:

- `sap.suite.ui.commons.ProcessFlow`
- `sap.suite.ui.commons.networkgraph.Graph`

- `dom-to-image`
- `html2canvas`

b) ABAP code

This section provides an in-depth explanation of the ABAP methods implemented in the project. The methods cover functionalities such as file uploads, validations, database interactions, authorization checks, logging, project hierarchy management, and URL generation.

Method: uploadExcelData

Functionality:
This method processes an uploaded Excel file by:

1. Saving it as a binary object in the Project Table.
2. Reading the Excel data into an internal table.
3. Looping over each row and writing the data into the corresponding database tables while ensuring foreign key relationships are maintained.

Key Implementation Details:

- Handling Keys (UUIDs):
  - A new UUID is only generated when the parent record changes.
  - Example: If an extension task changes, then all technical building blocks linked to it must receive a new foreign key.
- Differentiation Between New Projects & New Versions:
  - New versions should not overwrite existing fields such as:
    - Project ID
    - Created At Timestamp
    - Created By User
  - Instead, the Project Version is modified by appending a timestamp, allowing for easier version tracking.

Method: fields

Functionality:
This method monitors file uploads in the Project Details Page and ensures that the uploaded file is processed correctly.

Key Implementation Details:

- Automatically triggers `uploadExcelData` when a new filename is detected.
- Restores the old filename and attachment after the new version is created, preventing data loss.

Method: excelVal

Functionality:
This method validates an uploaded Excel file during the save operation of a project version.

Key Implementation Details:

- Column Name Validation:
  - Ensures the uploaded Excel file has the correct column names based on the dynamically specified table `Z00SAP_COL_NAMES`.
- Dynamic Column Name Check:
  - This approach allows flexibility when column names change in the future, making maintenance easier.

Method: exportExcelData

Functionality:
Handles exporting project data into an Excel file.

Key Implementation Details:

- The SAP-recommended solution did not work due to missing updates, so a custom workaround was implemented.
- Steps in Data Export:

1. Prepare & Collect Data: Ensures the exported Excel file follows the same format as the imported one.
2. Insert Column Headers: Uses `insert_col` to manually insert headers into the exported file.
3. Create Binary String: Converts database data into a binary string for Excel export.
4. Replace the Old Attachment: Saves the newly generated Excel file as the latest attachment.

Method: insert_col

Functionality:
This helper method creates a manual field catalog for all Excel columns during export operations.

Method: createUrls

Functionality:
This method constructs a URL when a project is saved. The URL links to the graph visualization for the selected project.

Key Implementation Details:

- Uses project metadata:

    - Project ID
    - Project Version
    - Extension Task ID
- Important Consideration:
    - base URL must be updated when the application is deployed on a new system.

Method: checkHierarchies & Color

Functionality:
This method validates the project hierarchy and color settings.

Key Implementation Details:
- Triggered by a user button press.
- Calls `validateHierarchy` to analyze technical building blocks.
- Error Handling:
    - Errors are displayed in a popup.
- Color Validation:
    - Ensures only R (Red), Y (Yellow), G (Green) are used.

Method: validateHierarchy

Functionality:
Helper method that checks hierarchical relationships between technical building blocks.

Key Implementation Details:
- Handles Double Positions (e.g., A1/B1) by splitting them into single entries.
- Sorting Logic:
    - Separates characters and integers to apply proper sorting.
- Error Detection:
    - Missing first value (e.g., A01 or B01 missing).
    - Duplicate entries (e.g., A01 appearing twice).
    - Missing sequence numbers (e.g., A01, A02, A04 where A03 is missing).
    - Invalid colors (anything other than R, Y, or G).

Method: validateProjectNameFilled

Functionality:
Ensures that a project name is provided before saving.

Key Implementation Details:
- If no project name is entered, an error window appears with:
  "Missing mandatory value: No Project Name specified!"

Method: duplicateProject

Functionality:
The method duplicates a selected Project, creating new entries with unique UUIDs in the table, with the same data of the selected project.

Key Implementation Details:
- Goes through all tables based on foreign key relationship, finding all entries connected to the selected project UUID
- Generates duplicate entries for each found record with new UUID and ensures foreign key relationship between newly created duplicate records is correct

Method: deleteProject

Functionality:
The method sets the field archived to true for a selected Project, no project will be actually deleted from the database based on project partner preferences.

Key Implementation Details:
- Sets `archived` field to true in project table
- Filtering in CDS view for only non-archived projects

Method: add_line

Functionality:
Allows users to dynamically add a new Technical Building Block (TBB) in the project details page.

Key Implementation Details:
- Updates service behavior definitions: `Z00SAP_ProjectDetails_I` and `Z00SAP_ProjectDetails_C_LESS`.
- Creates a new data definition `Z00SAP_ADD_LINE_PARAMS` to handle input fields.
- Inserts new hierarchical data into `Z00SAP_TBB` when triggered.

Method: get_global_authorizations

Functionality:
This method checks if a user has permissions to perform actions like Create, Update, Delete, and special project actions.

Key Implementation Details:
- Authorization Object: Z_A_24
- Calls sub-methods like is_create_allowed, is_update_allowed

Authorization Validation Methods

Each method checks a specific permission using AUTHORITY-CHECK.

is_create_allowed	Create a project	ACTVT = '01'
is_update_allowed	Update a project	ACTVT = '02'
is_delete_allowed	Delete a project	ACTVT = '06'
is_deleteproject_allowed	Full project deletion	ACTVT = '06'
is_duplicateProject_allowed	Duplicate a project	ACTVT = '02'
is_uploadExcelData_allowed	Upload an Excel file	ACTVT = '02'
is_exportExcelData_allowed	Export project data to Excel	ACTVT = '02'
is_checkHierarchies_allowed	Validate project hierarchy	ACTVT = '02'
is_addline_allowed	Add a new row to a project	ACTVT = '02'

Separate subcheck for each action implemented to allow for more flexible customer configuration in the future.

Architecture: Structured Data Definitions

Functionality:
Enhance the scalability and maintainability of the project, the team designed and implemented a structured framework for Data Definitions.

Key Implementation Details:

- For each entity a database table was created: `Z00SAP_PROJECTS, Z00SAP_TL, Z00SAP_ES, Z00SAP_ET, Z00SAP_TBB`
- Use foreign key which referencing his parent's primary key to establish the relationship. For example, `Z00SAP_TBB` has FK `et_id` which is the primary key of his parent `Z00SAP_ET`
- Each entity will have its own Data Definition, both Consumption(C) View, and Interface(I) View. For example, the Data Definition `Z00SAP_TBB_I` and `Z00SAP_TBB_C` was created.
- In each I view use `association to` and `composition of` keywords with proper configuration to associate to related table and get needed information for example `Z00SAP_ET_I` contains `association to parent Z00SAP_ES_I` and `composition [0..*] of Z00SAP_TBB_I`
- In each C view use the `redirect to` keywords with proper configuration to define the link which expose in the I view. For example, in `Z00SAP_ET_I associations: es, tbb` were exposed, then in `Z00SAP_ET_C` define these associations with: `es: redirected to parent Z00SAP_ES_C, tbb: redirected to composition child Z00SAP_TBB_C`
- In the root entity `Z00SAP_Projects_I` define the entry point to the structure

However, due to time constraints and project prioritization, this functionality was not deployed in the current version of the application. While it remains inactive in the present release, the framework has been saved in the codebase, ensuring it is readily available for future use. For the current version we use a join to fetch all relevant data.

6. Challenges during the project

1. Ensuring Referential Integrity in a Complex Data Model

One of the biggest challenges we faced was managing relationships between tables, especially when it came to deleting or duplicating records. There was always a risk of orphaned records or inconsistencies, which could cause major problems down the line. Given the complexity of our data model, maintaining referential integrity wasn't straightforward. We had to be extremely careful, validating every operation, running extensive tests, and taking a structured approach to data handling to make sure everything stayed consistent.

2. Adapting to New Technologies and ABAP RAP Development

Since ABAP and SAP RAP were new to most of the team, we were constantly figuring things out as we went. The unfamiliarity led to frequent changes in our architecture because we kept discovering better ways to implement things. On top of that, missing libraries and ABAP's own limitations, like not being able to use group by with abap.string, made things more complicated. Each roadblock forced us to rethink parts of our approach, which slowed us down but also gave us a much deeper understanding of the technology.

3. Limited Documentation and Hard-to-Find Resources

Another frustrating aspect was the lack of clear, easy-to-follow documentation for ABAP RAP. A lot of the material we found was either too technical, too vague, or just didn't cover what we needed. This meant we spent a lot of time troubleshooting, experimenting, and relying on trial and error to solve issues. Debugging was especially time-consuming, as we often had to manually dig through code just to figure out variable names and types. Sharing knowledge within the team became essential, since finding reliable external resources was often a struggle.

4. Data Cleaning and Transformation Without Losing Accuracy

Cleaning and transforming data may sound simple, but in reality, it was one of the more delicate parts of the project. We needed to remove redundant or irrelevant information without affecting the accuracy of our data, which was easier said than done. Any mistake in this process could lead to misleading visualizations and ultimately poor decision-making. Given how complex our data model was, we had to be extra cautious in making sure that no valuable information was lost during transformation.

5. Enhancing Visualization Readability While Maintaining Flexibility

We wanted to make sure our data visualizations were easy to read, but at the same time, we didn't want to limit how users could explore the data. Striking that balance was harder than expected. Simplifying the layouts improved readability, but too much simplification took away important details. It took multiple iterations and a lot of user feedback to get it right—adjusting layouts, fine-tuning data presentation, and making sure everything was both clear and flexible for different perspectives.

6. Project Constraints and Organizational Challenges

Aside from the technical hurdles, we also dealt with organizational challenges. At times, missing updates in the S/4HANA system slowed us down, and the uneven workload meant some team members had to take on extra tasks. Losing a team member added even more pressure, requiring constant adjustments and coordination to keep the project moving forward. These challenges made teamwork and communication even more critical, as we had to stay aligned and adapt quickly to keep everything on track.

7. Cool implementation features

Compared to the previous excel version, our project offers multiple added advantages. While the basic functionalities are important, the cool features our project offers are grounded in functionalities that go beyond what was previously possible. These features include:

1. **Chart Representation:** We implemented graphical charts in two distinct versions, allowing users to visualize data effectively. Additionally, the charts can be exported to the "jpeg" format, so users can share it with other stakeholders. The graphical charts are implemented in a process diagram and network graph format.

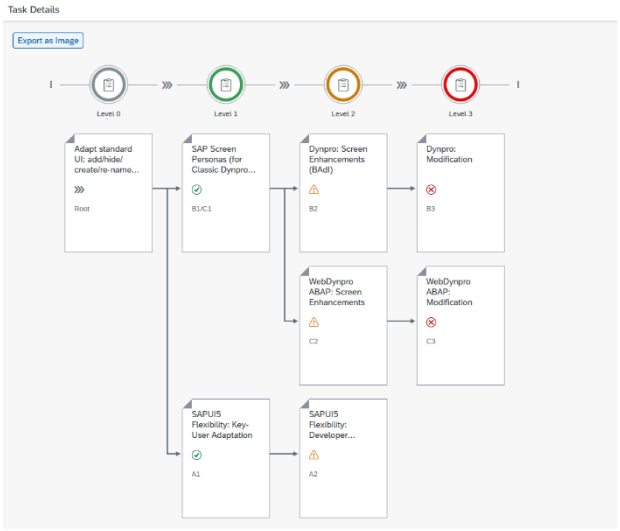


Figure 4: Process Flow Chart

This exemplary chart visually represents the hierarchical structure of the extension methodology in a network graph format. At the top, the root node signifies the selected application extension task, branching out into various possible implementation methods. The lines connecting the nodes illustrate the relationships between methodologies, while color coding aligns with SAP's Clean Core principles.

2. **Import and Export of Excel:** The application supports Excel file import and export, allowing for seamless transition from the previous way of strategy definition, as well as sharing offline versions of it with external stakeholders.

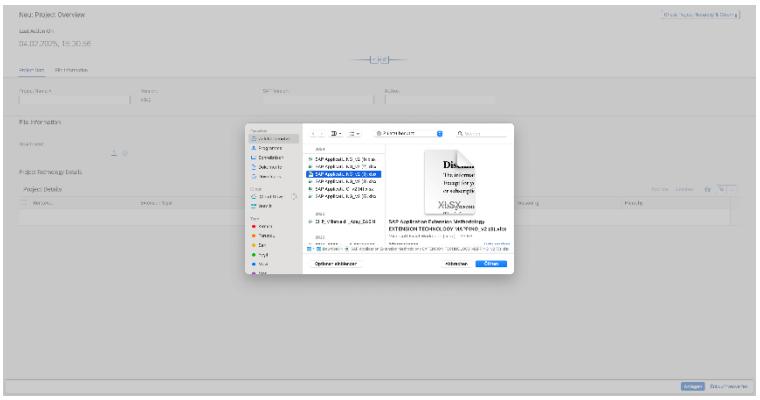


Figure 5: Excel Import

Additionally, after updating the project in the app, it is also possible to synchronize the changes to the uploaded excel file and then export the updated version.

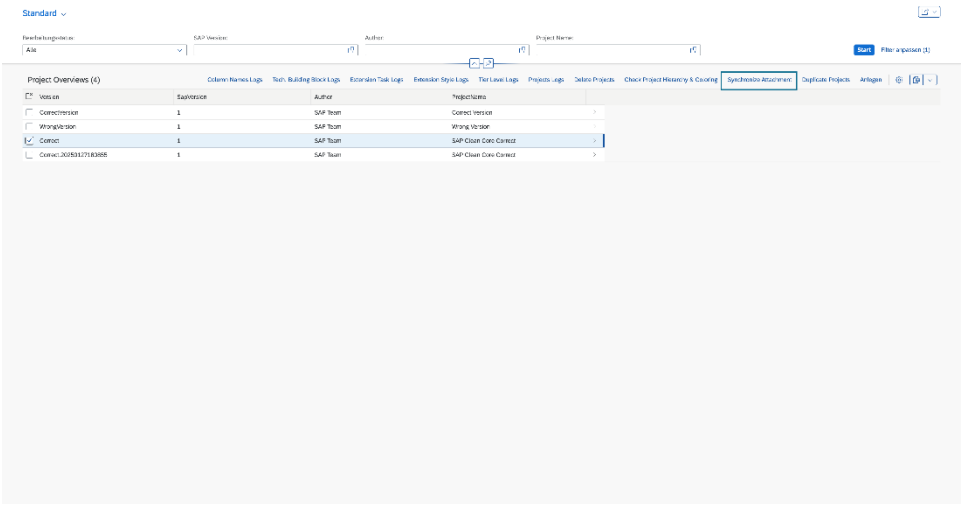


Figure 6: Synchronize Attachment Button

3. **Integration of Freestyle Application within List Report Page:** Our solution utilizes the list report page as a basis and integrates a freestyle SAPUI5 application through external links, extending the standard functionality while maintaining a structured UI. This integration allows for customized user interactions without compromising SAP Fiori design principles. For this integration, url is also encoded from SAP UI5 freestyle side in order to eliminate the additional effort to handle special characters in backend (ABAP) side.

4. **User Guidance and helpful validation functions:** Usability was a key focus throughout our implementation. As the previous Excel tool did not offer any guidance or validation checks on inputs, we put specific emphasis on integrating user guidance

into our application. Input is now validated both syntactically and semantically and feedback provided to enhance the user experience. Here are some examples of our validations and feedback:

The feedback after the user checks the hierarchy of its project:

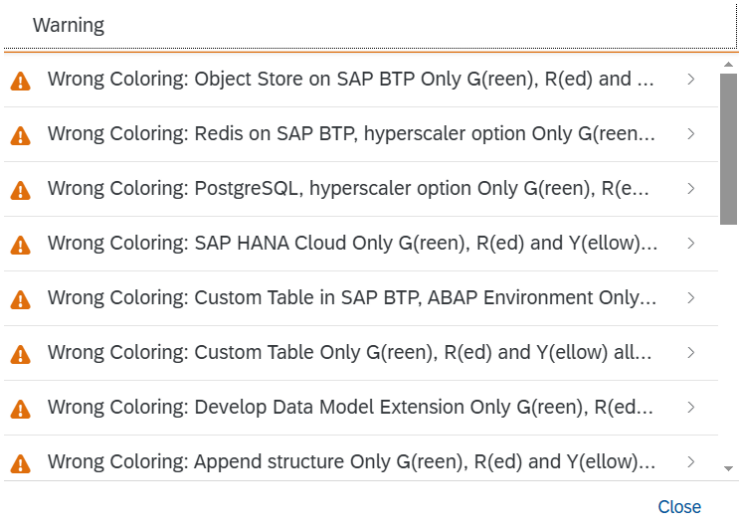


Figure 7: Feedback Hierarchy Control

The validation if the excel format is not correct:

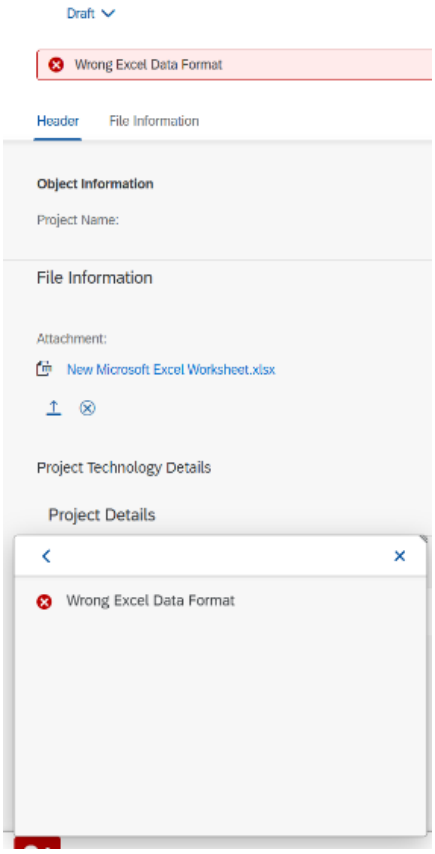


Figure 8: Feedback Excel Format

8. Responsibilities

All team members:

- Implementation of the ABAP foundation in a group coding session
- Creation of User Stories and Epics
- Review of presentations
- Creation of Software Architecture

Achilleas:

- Meeting Organizations
- Semantic Model
- Technical Model
- Method uploadexceldata
- Method fields
- Method excelVal
- Method exportExcelData
- Method createUrls
- Method insert_col
- Method checkHierarchies & Color
- Method validateHierarchy
- Method validateProjectNameFilled
- Data Definition Z00SAP_ProjectDetails_I
- Metadata Extensions of Overview for Validation
- Metadata Extensions of Project Details for Upload
- Removed UUID from Overview and details (with Lea)
- Grouped by Project Name and made the Project Name mandatory with validation (Pair programming with Lea)
- Communication with Lukas (SAP Supervisor)
- Cleared all testing data and generated data for presentation
- Detailed unformatted documentation of my implementation
- Hierarchy coloring for Fiori Application
- Table z00sap_col_names for dynamic name checking, importing, and exporting.
- Logging Functionality
- Buttons for Logging
- Message Class Z00SAP_ERROR_MESSAGE
- Removed routing from Project details

Jiangang:

- Design software architecture
- Generate database
- Create Semantic Model
- Create Technical Model
- ABAP RAP
    - Method deleteProject (With Lea)

    - Method duplicateProject (With Lea)
    - Data Definition Z00SAP_ProjectOverview_I (With Lea)
    - Data Definition Z00SAP_ProjectOverview_C (With Lea)
    - Metadata Extentions Z00SAP_ProjectOverview_C (With Lea)
    - All Database table
    - Behavior Definition Z00SAP_ProjectOverview_I
    - Behavior Definition Z00SAP_ProjectOverview_C
    - All Data definitions which saved for future use (e.g. Z00SAP_TBB_I, Z00SAP_ET_C, …)
- SAP UI5 Freestyle Application (With Yunus)
    - Process Flow View UI5 freestyle
    - Network Graph View UI5 freestyle
    - Project Detail Controller
- Task Details Controller
    - ProcessData()
    - CreateRootNode()
    - enrichData()
    - bindDataToTable()
    - generateProcessFlow()
    - onExportAsImage2()
    - onExportAsImage()
    - generateNetworkGraph()
- Documentation
    - Sprint Document 1
    - Sprint Document 2
    - Sprint Document 3
    - Sprint Document 4

Kadir:

- Meeting Organizations
- Creation of Presentations
- Epics and User Stories
- Agile Roadmap (Collaborative with team on JIRA)
- Mockup Pages for Sign-In, Project Overview, Project Details, Extension Graph
- Mockup Animation Videos for Import, Update, Export, Logging
- Progress Presentation
- Creation Videos for Demo in Progress Presentation
- Authentication and Authorization (with Lea)
- Creation of Authorization Methods and Authorization Object (with Lea)
- Definition and creation of Admin and View Only Roles (with Lea)
- Creation of Authorization Profile (with Lea)
- Implementation of Parent-Child relationships (Collaborative with team)
- Documentation Creation
- Motivation and Situation (with Lea)
- ABAP Code (Documentation)

- Challenges (Documentation)
- User Manual (Documentation)

Lea:

- Creation and design of all Presentations
- Epics and User Stories Refinement
- Agile Roadmap (specifically updates during project)
- Maintaining supporting Sharepoint for external documentation
- Method deleteProject (with Jiangang)
- Method duplicateProject (with Jiangang)
- Method add_line (with Yunus)
- Authentication and Authorization (with Kadir)
- Creation of all Authorization Methods and Authorization Object (with Kadir)
- Definition and creation of Admin and View Only Roles (with Kadir)
- Creation of Authorization Profile (with Kadir)
- UI for Z00_SAP_ProjectOverview_C Object Page in SAP Fiori List Report
- Removed UUID from Overview and details (with Achilleas)
- Grouped by Project Name and made the Project Name mandatory with validation (with Achilleas)
- Documentation creation
    - Motivation (with Kadir)
    - Requirements Analysis
    - Approach
    - Implementation SAPUI5 (with Input from team members)
    - Cool Implementation Features
    - Access Information

Yunus:

- Meeting Organizations
- Semantic Model
- Technical Model
- Method add_line
- Entire SAP UI5 Freestyle Application:
    - Project Overview view UI5 freestyle
    - Project Details view UI5 freestyle
    - Task Details Table view UI5 freestyle
    - Process Flow View UI5 freestyle
    - Network Graph View UI5 freestyle
    - Manifest file customization
    - Base Controller
        - getRouter()
        - uuid_encoder()
        - uuid_decoder()
    - Project Overview Controller
        - OnDeletePress()

        - OnProjectSelect()
    - Project Detail Controller
        - OnRouteMatched()
        - OnLineItemPress()
    - Task Details Controller
        - OnRouteMatched()
        - fetchBackendData()
        - ProcessData()
        - CreateRootNode()
        - enrichData()
        - bindDataToTable()
        - generateProcessFlow()
        - onExportAsImage2()
        - onExportAsImage()
        - generateNetworkGraph()
- Data Definition Z00SAP_Projectdetails_C_Less
- Behavior Definition Z00SAP_Projectdetails_C_Less
- Data Definition Z00SAP_ADD_LINE_PARAMS

9. Acccess Information

The application is currently deployed on the S40 system and can be accessed via the following link. To obtain editing permissions, users must be assigned the Z_Admin role. Moreover, the Freestyle application is integrated with the main application and can be accessed through internal links.

Access Link Deployed Application:
https://s40lp1.ucc.cit.tum.de:8100/sap/bc/ui5_ui5/sap/z_mproov_00/index.html?sap-client=300

In addition to the deployed application, we have made our code repository available on GitHub. Ultimately, this is intended to make this application available to SAP customers, enabling them to adapt the base code to their own needs before integrating it into their systems.

Access Link Git Repositories:

Frontend Repository: https://github.com/cody3807/sap_graph_chart

Backend Repository: https://github.com/AchiTsa/SAP_TUM_AEM_Config

10. User Manual

This user manual provides detailed instructions on how to use the **Application Extension Methodology** application. It covers key functionalities, navigation, and troubleshooting to ensure smooth usage.

System Requirements

- Supported Browsers: Chrome, Edge, Firefox
- User Roles:
    - **Admin Role:** Full access, including creating, editing, and deleting records, as well as managing settings.
    - **View Only Role:** Read-only access to the application and data, without the ability to make changes.

Navigation

The navigation of the system is very intuitive with two main pages **Project Overview** and **Project Details.**

1. Project Overview:

Once you press “Start” in the Overview section, all projects are displayed and grouped by their ID, ensuring that different versions of the same project are organized together.

At the top, you will find filtering options, and below them, various actions that can be performed. These actions will be explained in detail in the “User Actions in the Project Overview” section.

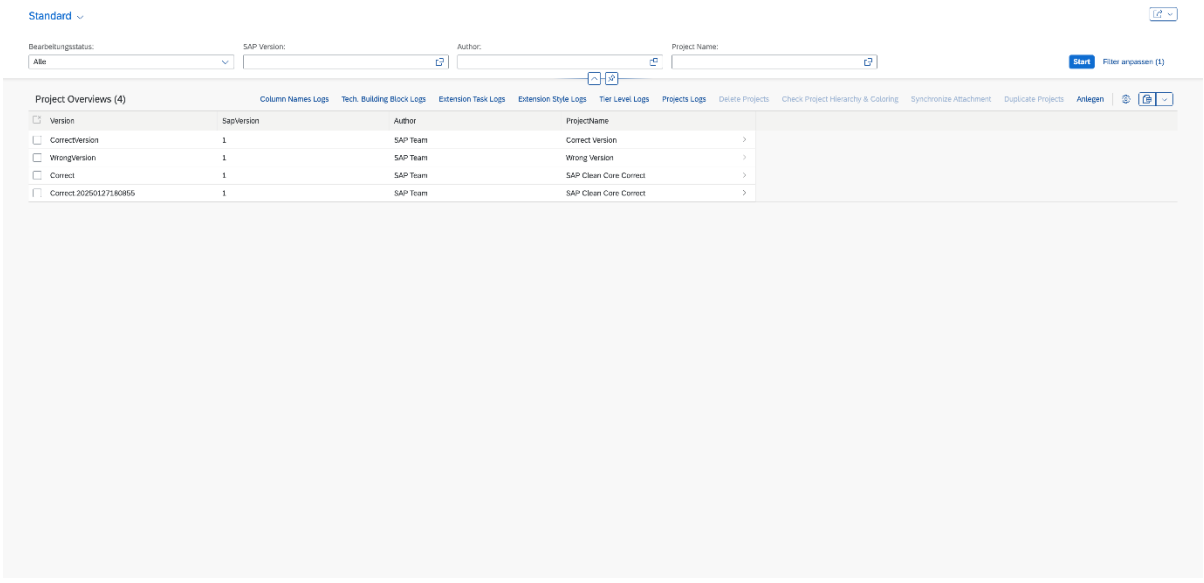


Figure 9: Project Overview

Once you click on a project, the details of the project are displayed:

2. Project Details:

You find here the attributes of the project and the file that is uploaded. There are also two user actions which are the “Edit (Bearbeiten)” and “Check Project Hierarchy & Coloring”, which will be also covered in the “User Actions in Project Details” section.

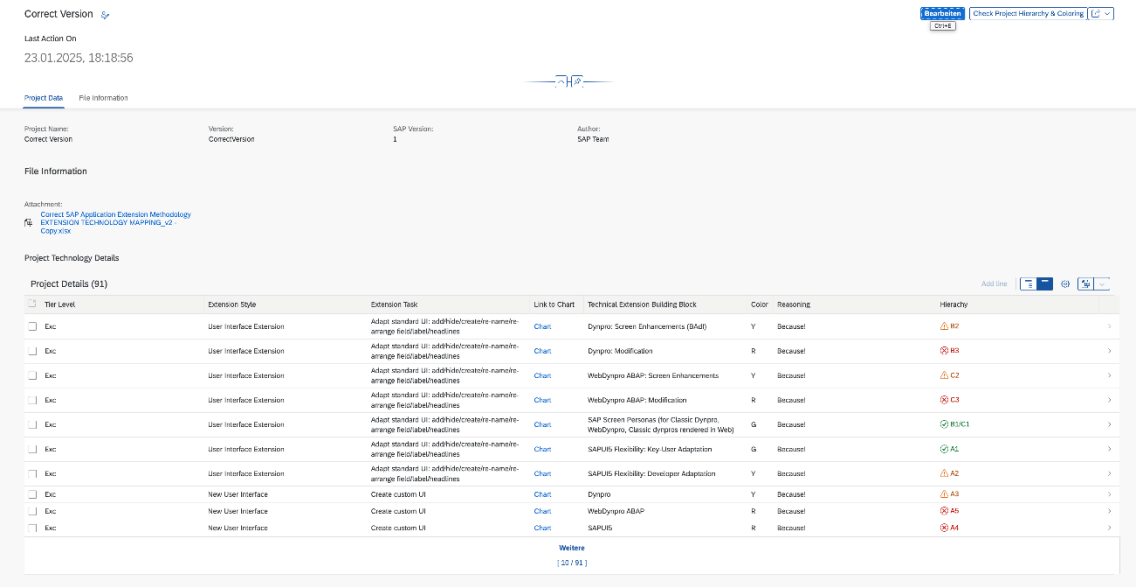


Figure 10: Project Details

User Actions in Project Overview

Displaying Change Logs

To view the changes in different attributes of a project, select the desired attribute to access its logs. To do this, click on the project’s box and then choose one of the following options:

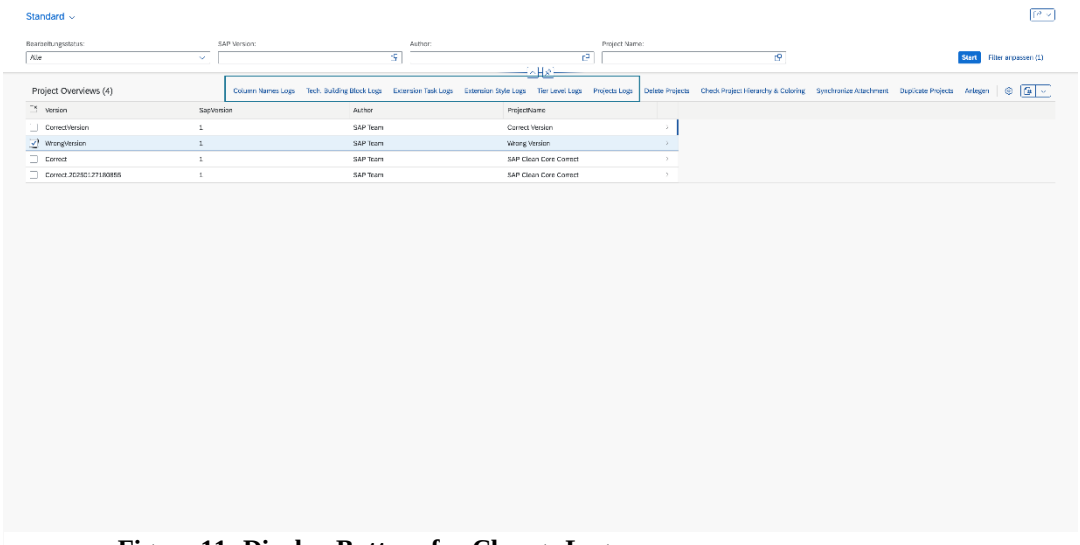


Figure 11: Display Buttons for Change Logs

'Column Names Logs', 'Tech. Building Block Logs', 'Extension Task Logs', 'Tier Level Logs', or 'Project Logs’:

Once you click on one of the Logs buttons, a new page will open in the SAP GUI web interface, directing you to the internal SAP logging for this table. Here, you can view all changes along with the modified data, providing a detailed overview of the updates:

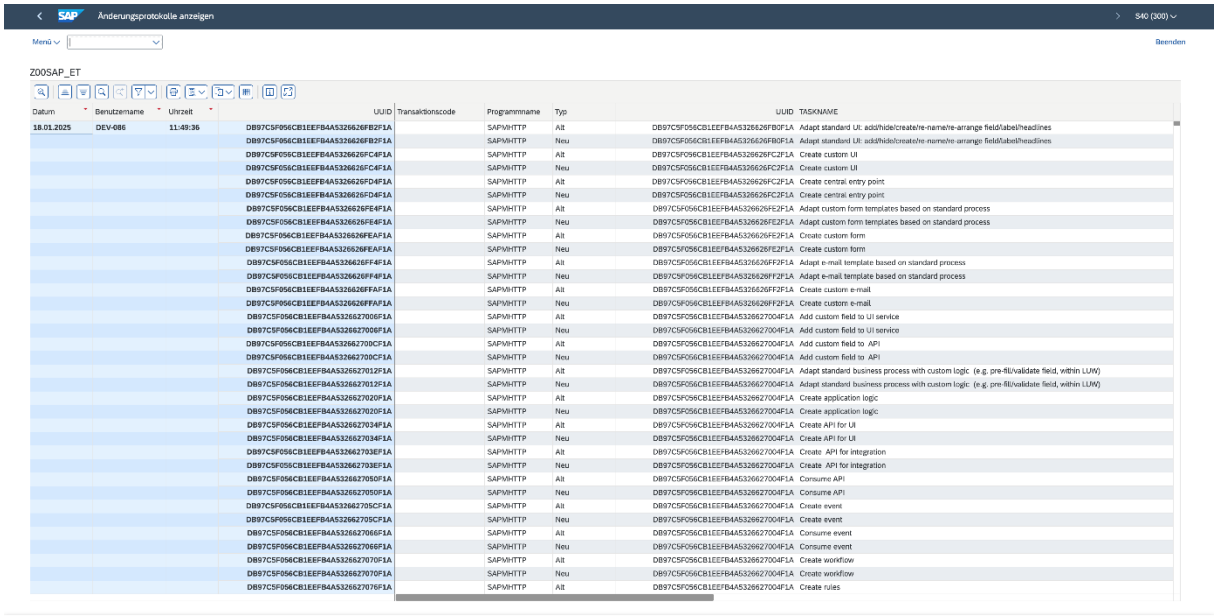


Figure 12: Change Log Example

Delete and Duplicate Projects

To delete a project, click on the project's box. You will notice that the previously grey 'Delete Project' button turns blue, indicating it is now available. Click on the 'Delete Project' button and then refresh the page to update the overview.

The same process applies for duplicating projects. Click on the project's box, then click on the 'Duplicate Project' button. The selected project will be duplicated. Refresh the page to see the updated overview.

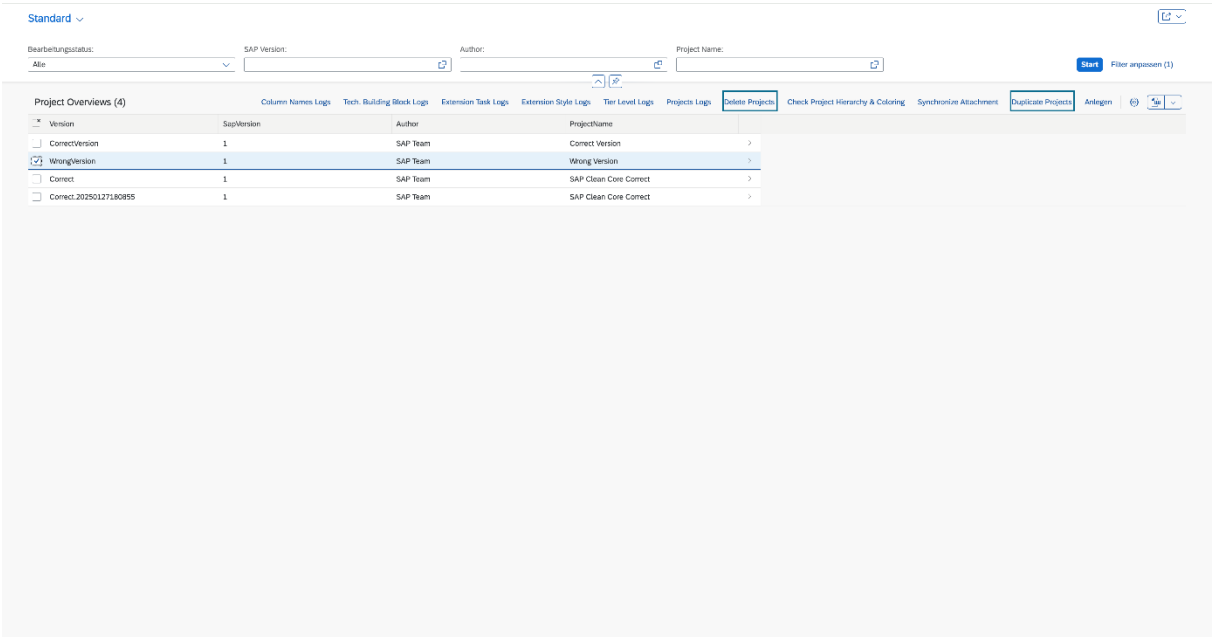


Figure 13: Delete and Duplicate Project Buttons

Check Project Hierarchy and Coloring

To check the hierarchy and coloring of a specific version of a project select the desired project by clicking on its box. Then click on "Check Project Hierarchy & Coloring" to verify the hierarchy and color assignments of your Technical Extension Building Blocks.

Error Handling and Validation

- If the check fails, a pop-up will display the detected errors. To resolve them, you will see the affected extension task within the hierarchy along with details about the underlying issue.

- Example of an Error Message:

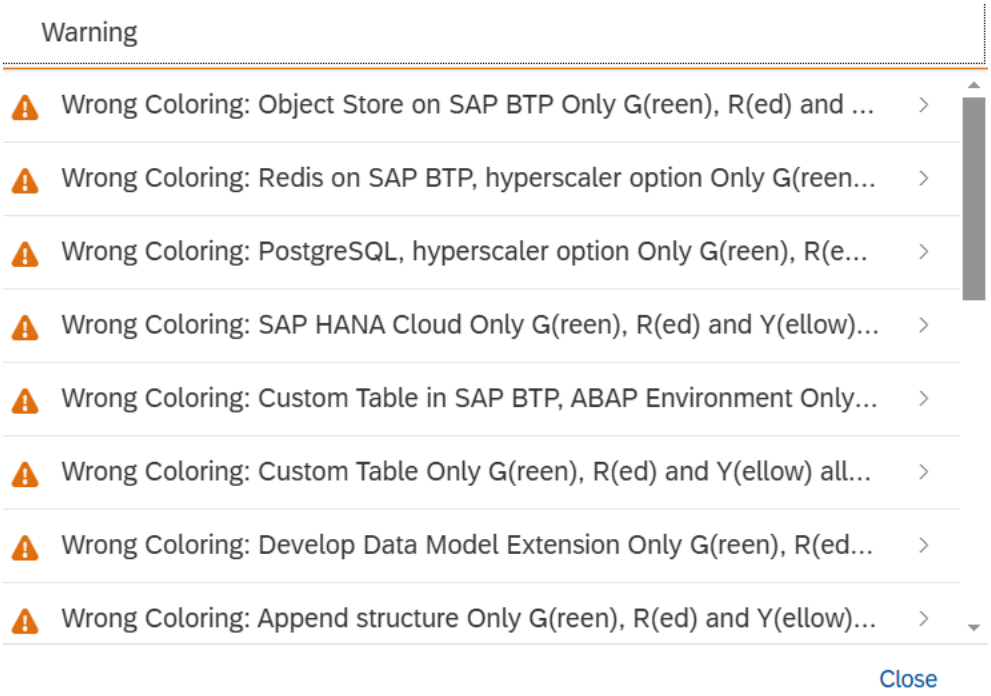


Figure 14: Feedback for Hiearchy Check

Important Notes:

- Projects are always imported, regardless of hierarchy or color structure.
- Structure and coloring are crucial for the Graph function. Additionally, correct coloring will also be reflected in the Project Detail View.

Common Errors That Can Occur:

1. **Duplicate Position:** A position appears more than once (e.g., A01 appearing twice).
2. **Missing Sequence Number:** A break in numbering, such as A01, A02, A04, where A03 is missing.
3. **Missing First Value:** The first value in a sequence is absent (e.g., A01 or B01 is missing).
4. **Wrong Coloring**: An invalid color code is used instead of R (Red), Y (Yellow), or G (Green).

If the check is successful, it confirms that all technical building blocks within the selected project version follow a valid hierarchy and coloring structure. A confirmation message will appear at the bottom center of the screen:

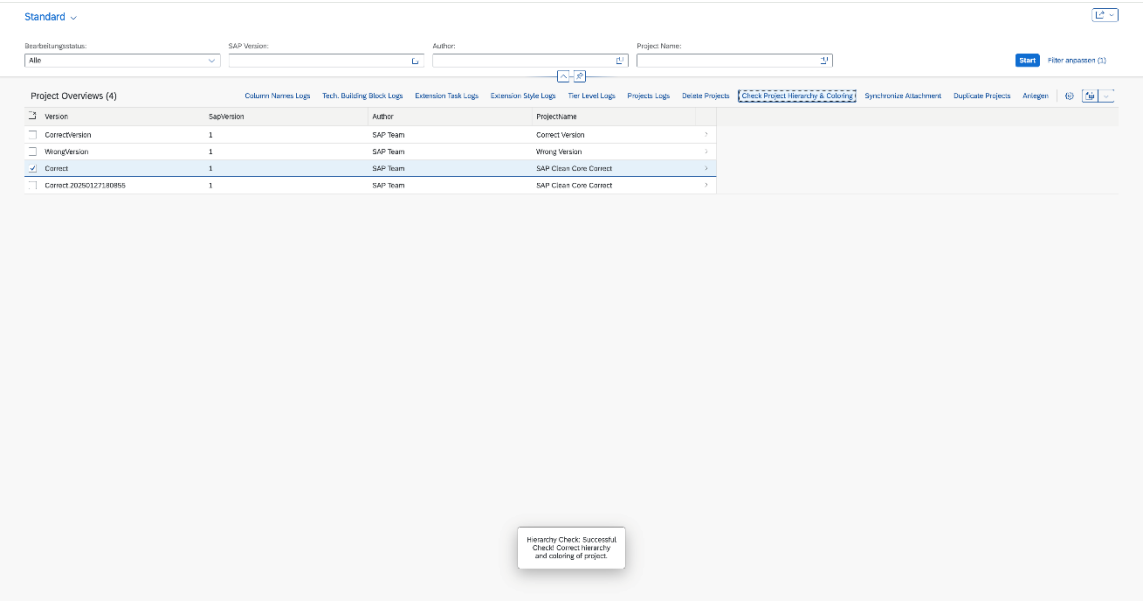


Figure 15: Feedback Hierarchy Check without Error

Export Excel Data

This action is not primarily performed on the **Project Overview** page. However, before exporting your project as an Excel file, you should ensure that your changes in the app are synchronized with the previously uploaded Excel document. This synchronization should be done in Project Overview page

To do this, click on the desired project's box and click on the "Synchronize Attachment" button. This will transfer all changes from the SAP database into an Excel document and save it as a large object in the attachment field.

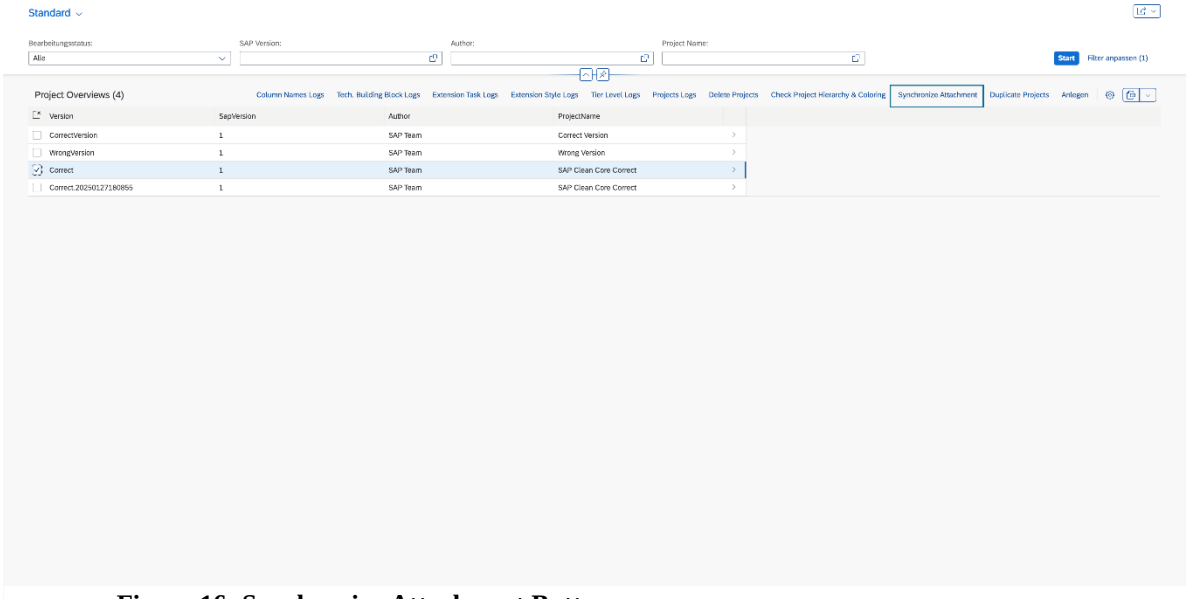


Figure 16: Synchronize Attachment Button

Next, select the project you just synchronized and enter the Project Detail View. In the File Information section, click on the blue-highlighted Excel file name to download it.

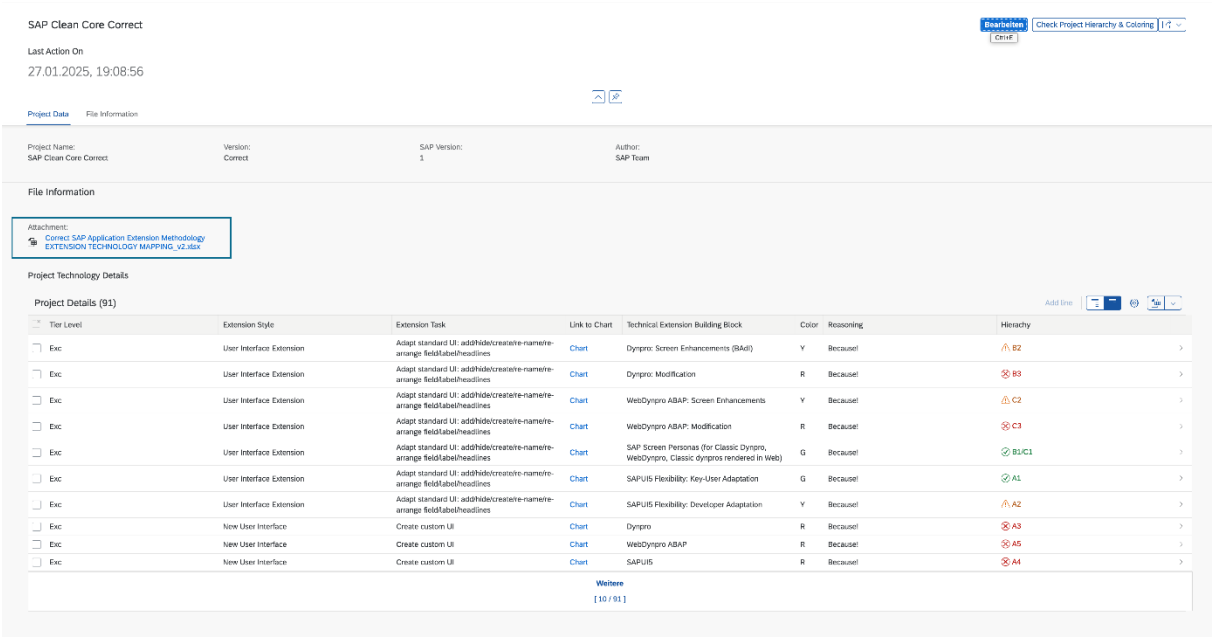


Figure 17: Download of an Updated Excel

Once you click the link, your browser will notify you that the file has been successfully downloaded.

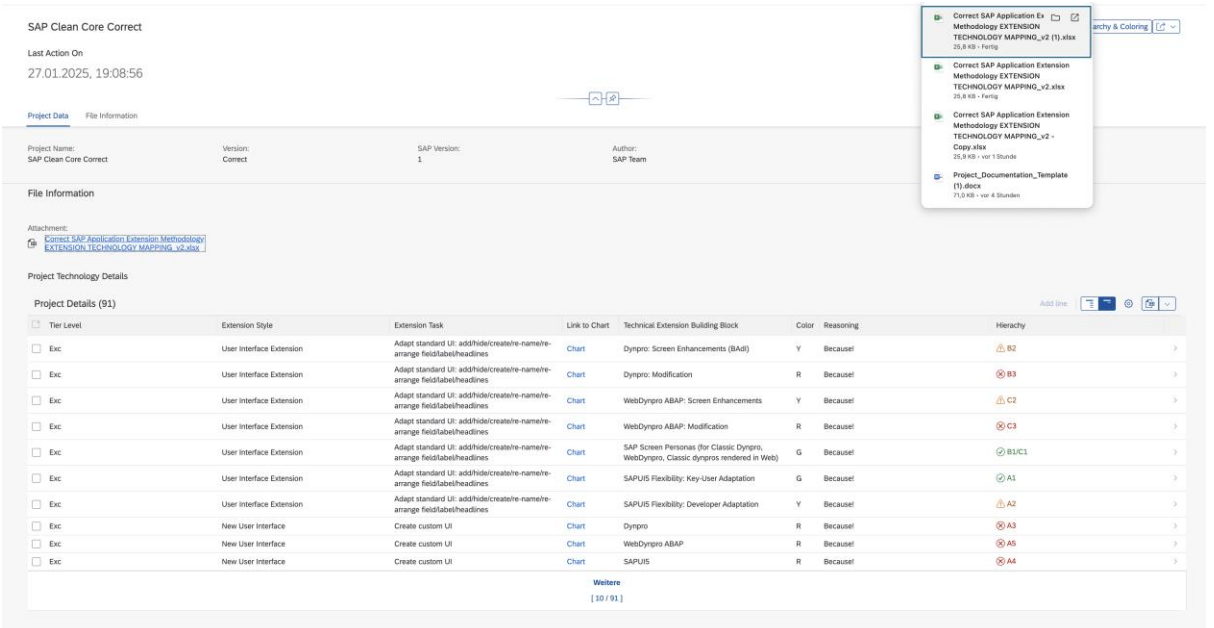


Figure 18: Successful Download

Create a Project

To create a new project, click the "Create" button located at the top right of the Project Overview page. Next, select your desired version and click "Continue."

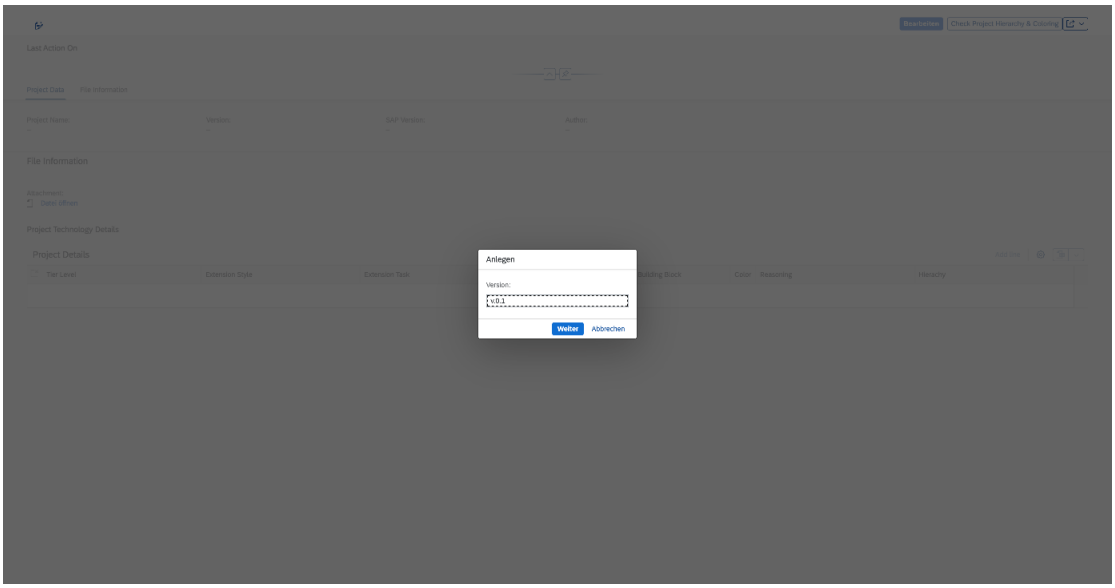


Figure 19: Version Input

You will then be directed to the Project Creation page, where you need to provide the following details:

- Project name (mandatory) – Every project must have a name.
- SAP version (optional) – Specify the SAP version if needed.
- Author (optional) – Enter the author's name if applicable.
- An Attachment field for uploading your Excel File of Project Details, please read the next subsection for details.

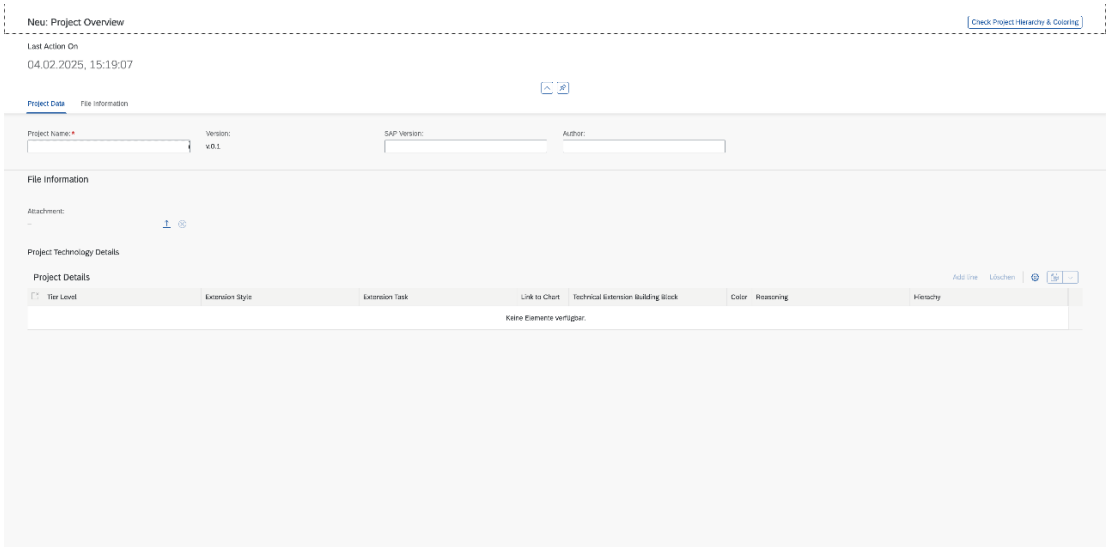


Figure 20: Project Creation Page

Upload an Excel File (While Project Creation)

In the middle of the screen on the left side in the attachment section, click on the arrow up to select an Excel sheet of your choice.

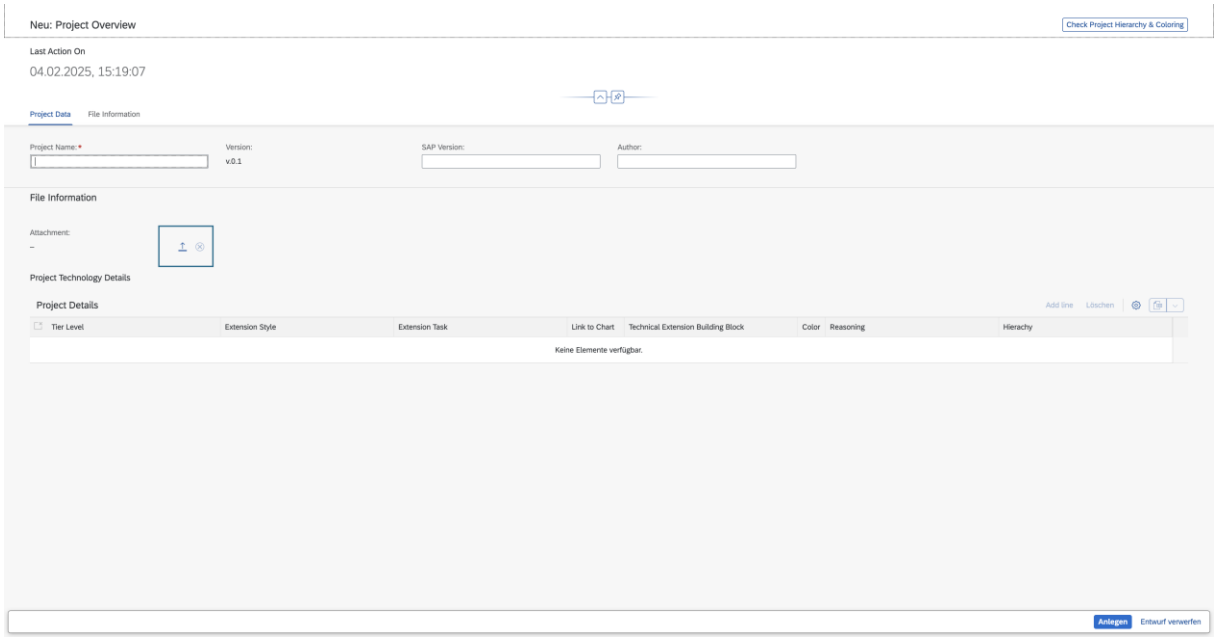


Figure 21: Upload Button for Excel

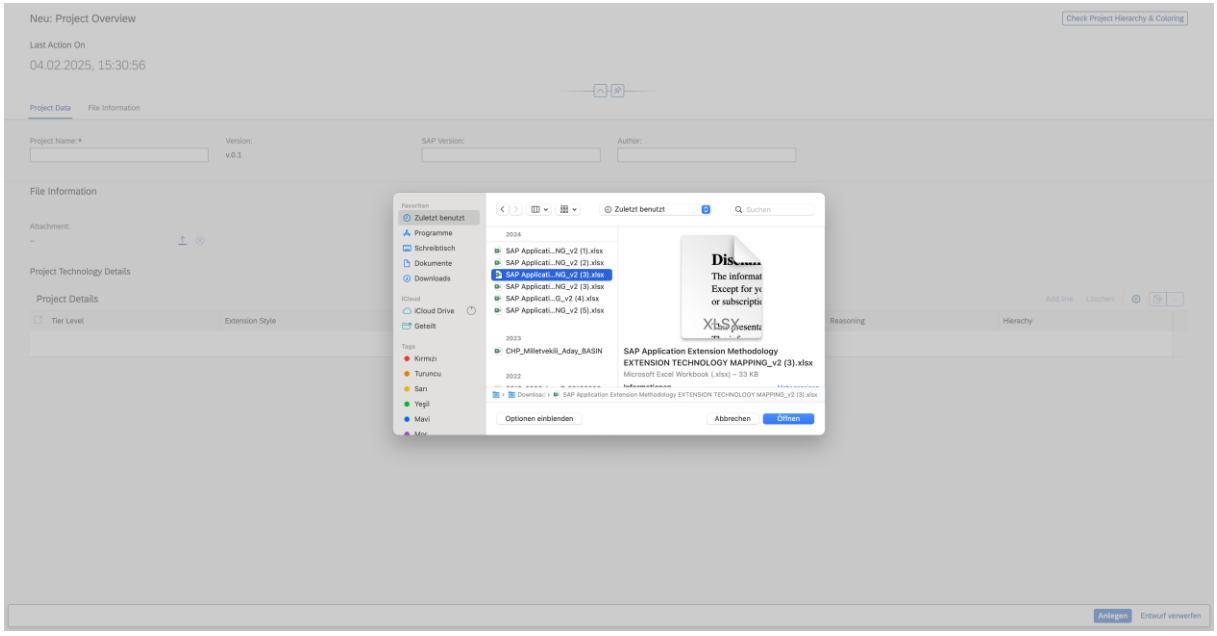


Figure 22: Excel Upload from Browser

Click open in the bottom left of your file explorer and the file is automatically uploaded to SAP.

Note that only xlsx files are allowed, otherwise upload is not successful, and you will be notified like this:

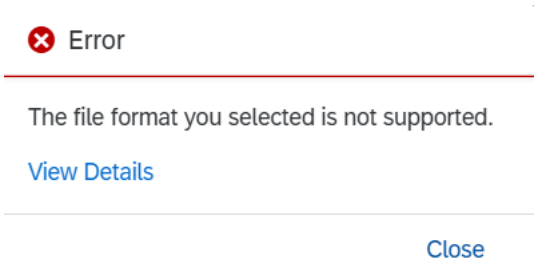


Figure 23: Error if user upload other file type than Excel

Note that the excel upload is also checking whether the excel sheet has the correct columns in correct order:

- Three-Tier Architecture
- Extension Style
- Extension Task
- Extension Domain
- Core Extension Domain
- Technical Extension Building Block
- Color
- Reasoning
- Hierarchy
- Link

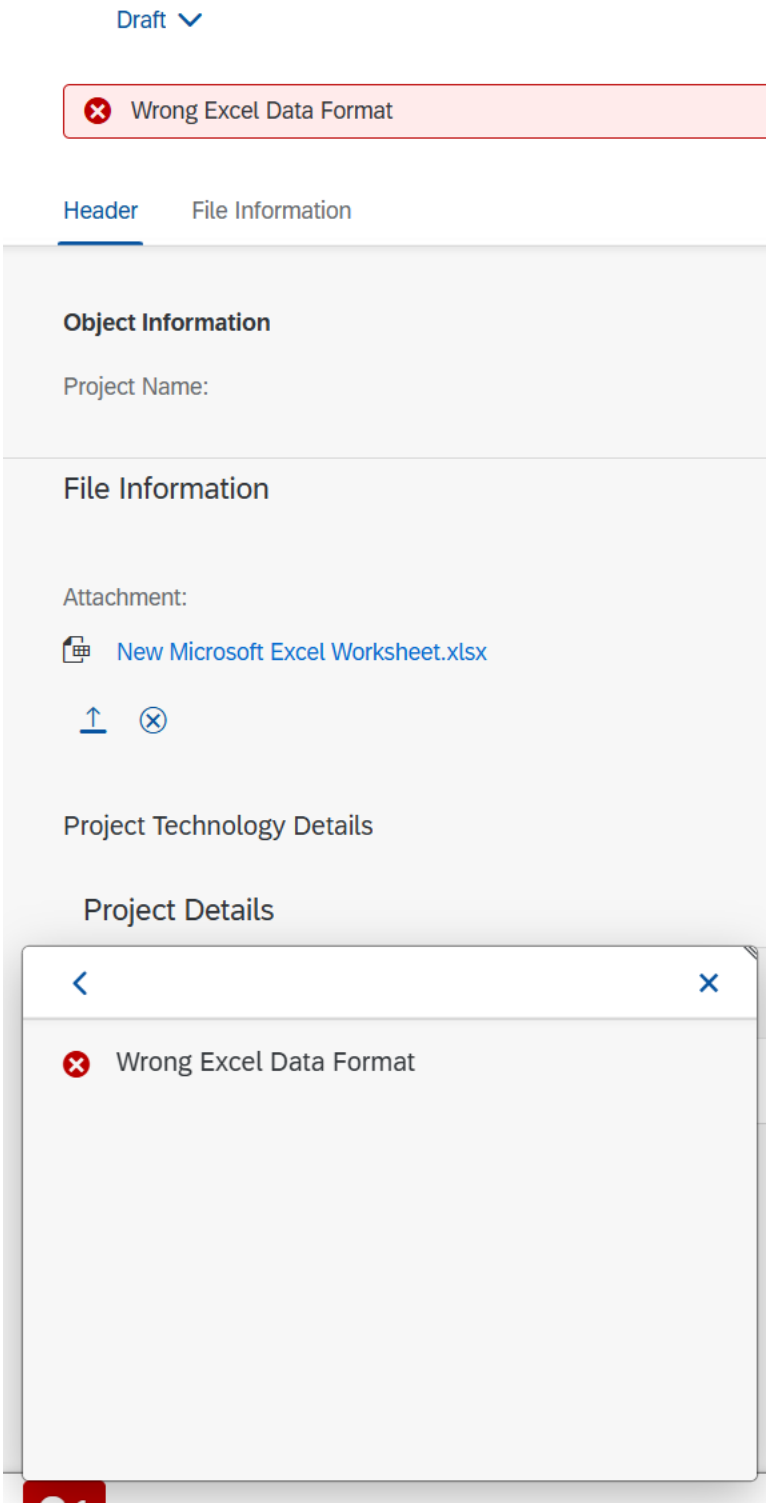


Figure 24: Error for Wrong Format of Excel Table

Note that adding additional rows and types of, for example extension styles is not a problem and intended use case.

User Actions in Project Details

Upload / Update an Existing Excel File

To update an existing project using an upload, first, select the project you want to modify. In the top right corner of the screen, click **Edit (Bearbeiten)** to enter editing mode.

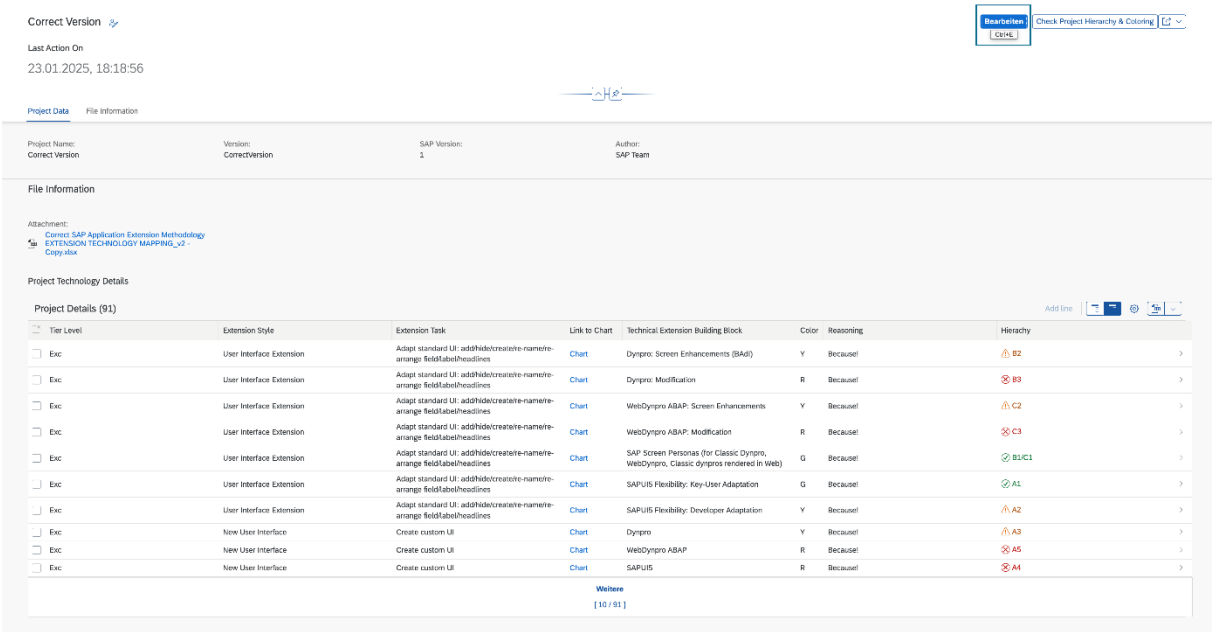


Figure 25: Edit Button

In the attachment section, located in the middle of the screen on the left side, click **the upload arrow** to select an Excel file. Only files in **.xlsx** format are allowed. After selecting the file, click **Open** in the bottom left corner of your file explorer. The file will be automatically uploaded.

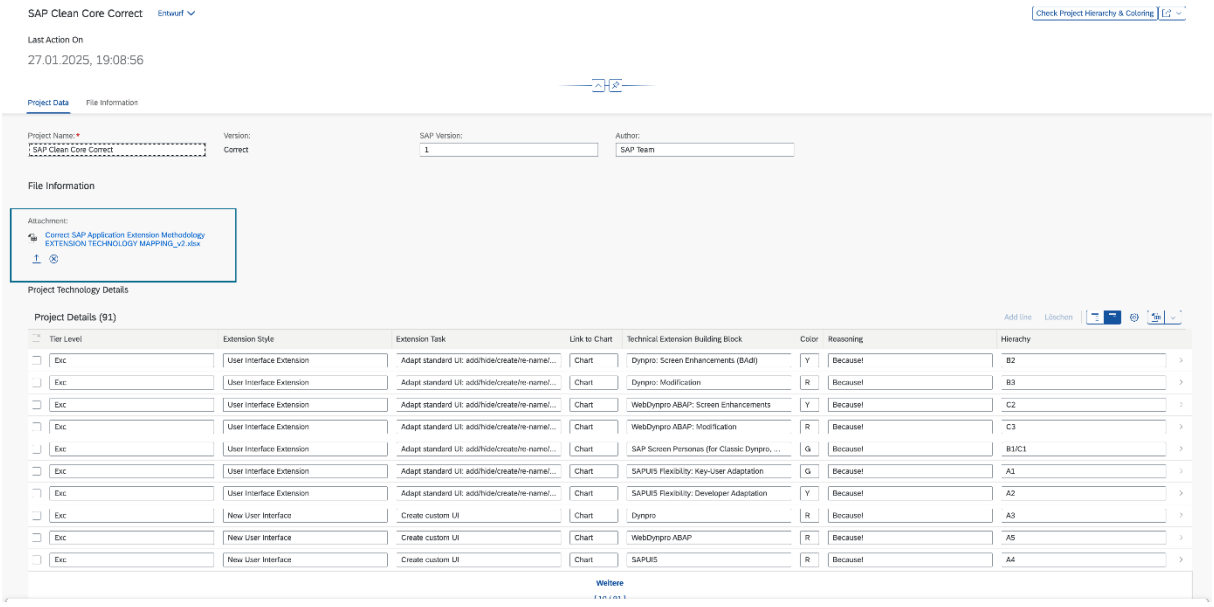


Figure 26: Updated Excel File

Once the file is uploaded, click **Save (Sichern)** in the bottom right corner of the screen.

After saving, a new version of the project will be created with a timestamp. The updated version will be saved under the **Project Overview**, linked to the correct project.

This ensures that all project versions and their timestamps are available for reference, providing a clear overview of changes over time.

Note that the excel upload is also checking whether the excel sheet has the correct columns in correct order (same as uploading during creation):

- Three-Tier Architecture
- Extension Style
- Extension Task
- Extension Domain
- Core Extension Domain
- Technical Extension Building Block
- Color
- Reasoning
- Hierarchy
- Link

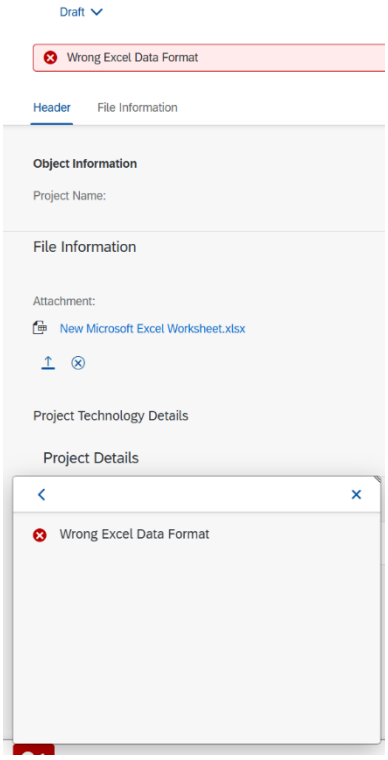


Figure 27: Feedback for Wrong Excel Format

Note that adding additional rows and types of, for example extension styles is not a problem and intended use case.

Displaying the Chart of the Application Extension Methodologies

In the Project Details Page, each project has a dedicated link to its Application Extension Methodology Chart.

The chart visually represents the selected application extension methodology, showing all possible extension implementations that branch from it.

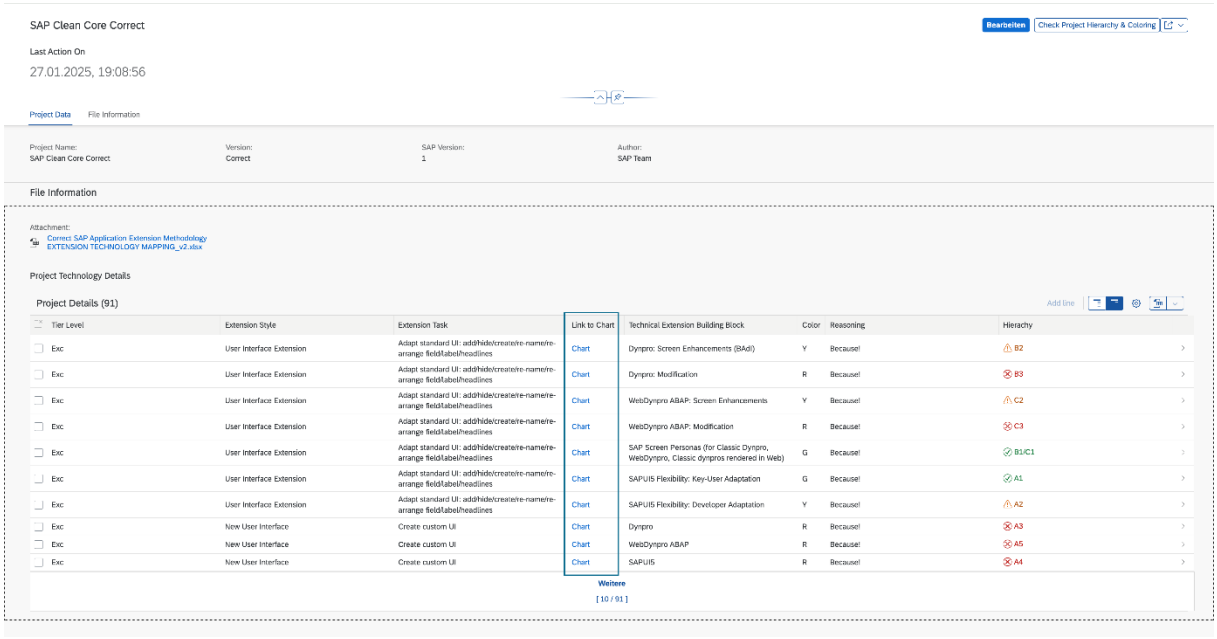


Figure 28: Chart Buttons

The **Extension Task** represents the specific application extension task assigned to the project, while the **Extension Building Block** refers to the method used for implementation.

The **Hierarchy** column indicates where the user-defined Application Extension Methodology fits within the overall structure.

The **Color** coding provides guidance on compliance with SAP's Clean Core methodology:

- Green: Fully compliant with SAP's Clean Core methodology and recommended.
- Yellow: Feasible, as the technology might be required for the project but may require careful evaluation due to potential maintenance challenges.
- Red: Not recommended, as it conflicts with SAP's Clean Core principles and may introduce

After clicking the "Chart" button, the user will be directed to a Task Detail where you have three sections:

- **Table View:** The Table View presents a structured list of all extension tasks associated with a project. Each row represents an individual extension task with detailed attributes.
    - Extension Task ID: Unique identifier for each extension task.
    - Project ID: Links the task to its corresponding project.
    - Version: Displays the version status (e.g., "Correct").
    - Hierarchy: Represents the position of the task within the methodology structure.
    - Task Name: Describes the action or function of the extension task.
    - Parent Node: Indicates the task's preceding step in the hierarchy.
    - Level: Represents the depth of the extension task within the structure.
    - TBB Name (Technical Building Block): The method used for implementing the extension.

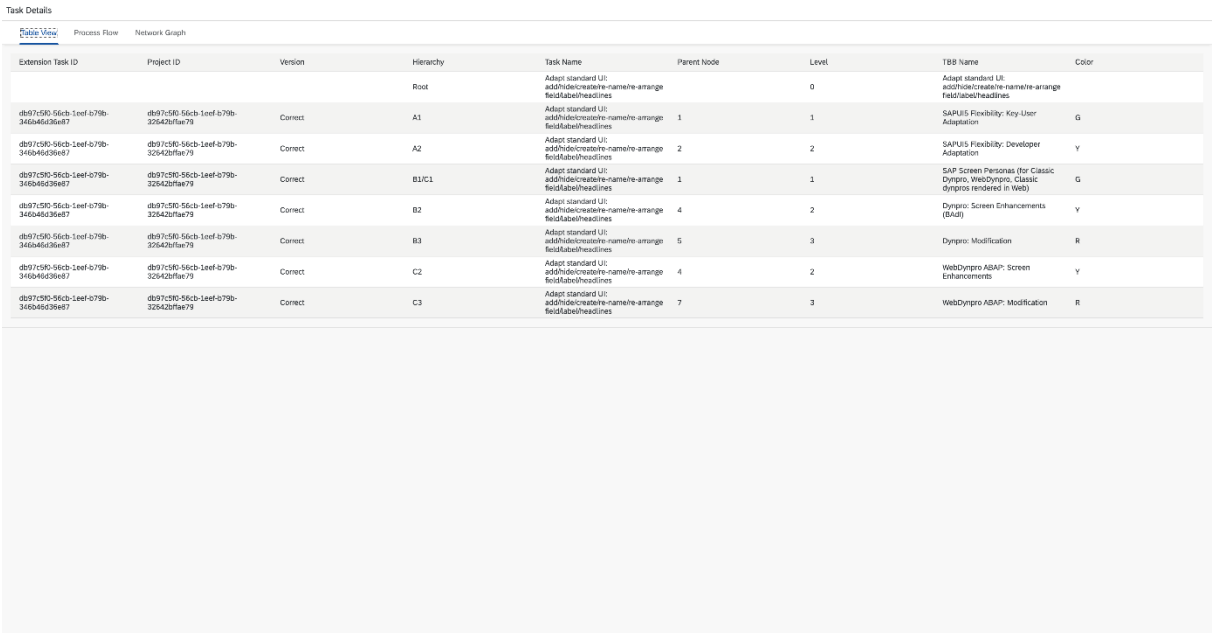


Figure 29: Table View of Project Hierarchy

- **Process Flow**: The Process Flow provides a visual representation of the extension methodology in a step-by-step flowchart format.
    - The top node (Level 0) represents the selected application extension methodology.
    - The subsequent levels (Level 1, Level 2, etc.) represent implementation options that branch out from the selected methodology.
    - Each task is color-coded according to its compliance with SAP's Clean Core methodology

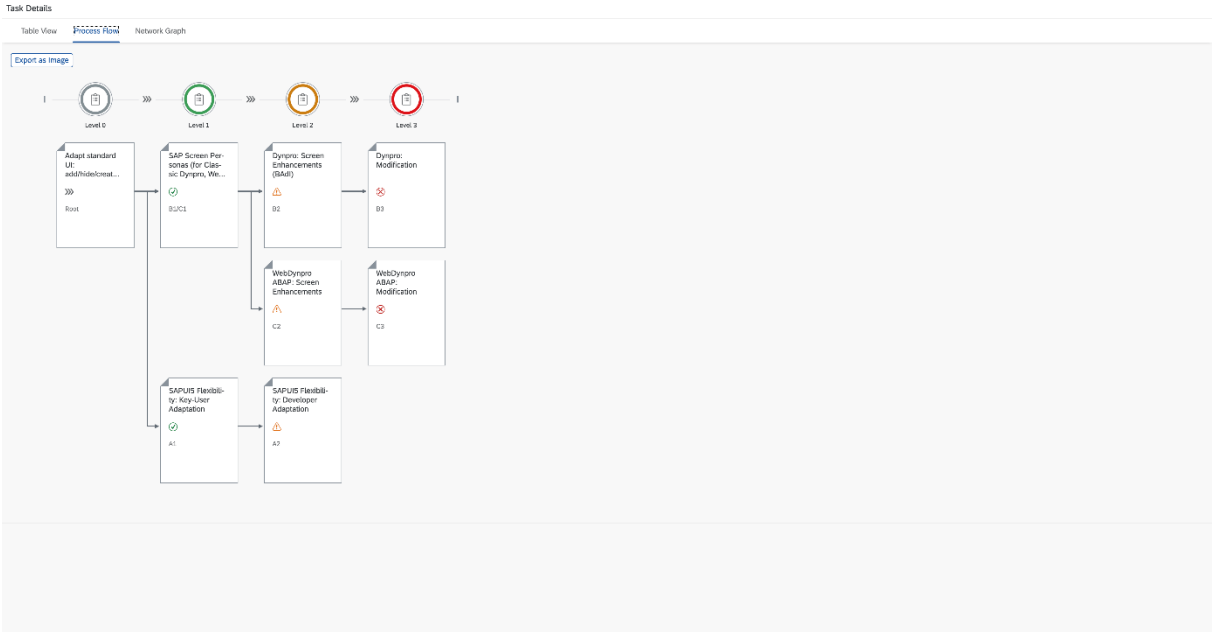


Figure 30: Process Flow Chart

- **Network Graph:** The Network Graph displays the hierarchical structure of the extension methodology in a tree diagram format.
    - The root node at the top represents the selected application extension methodology.
    - Nodes branch out into possible implementation methods.
    - Lines connecting nodes indicate relationships between methodologies.
    - Color coding follows SAP's Clean Core principles

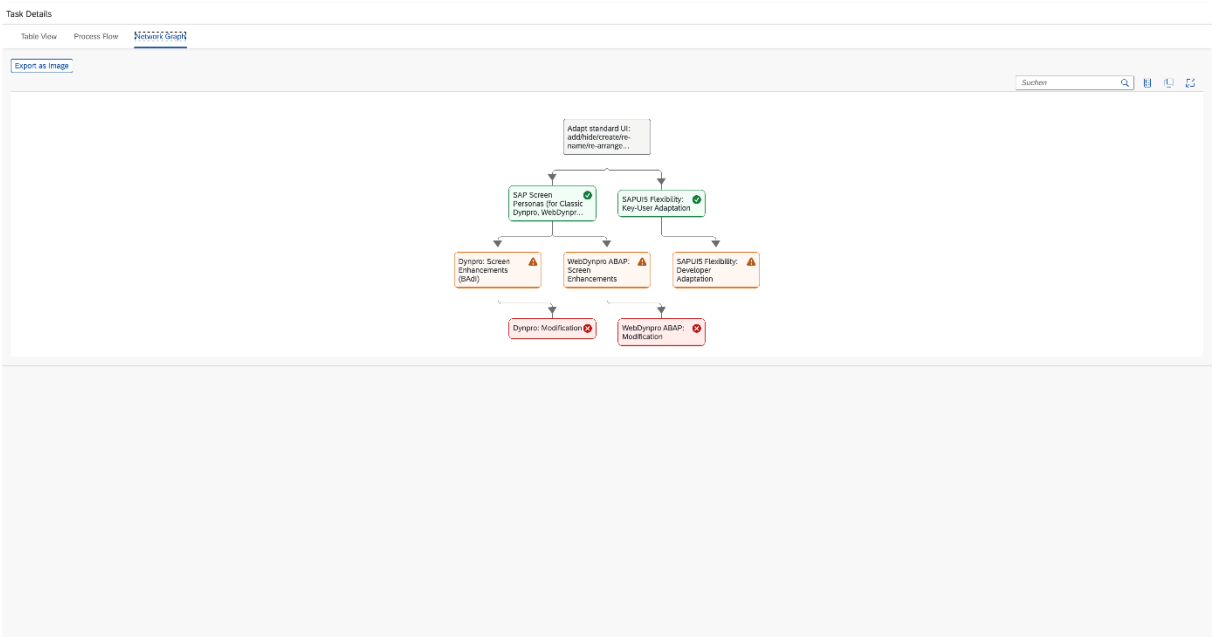


Figure 31: Network Graph

References

Gerbershagen, J. (2020). *SAP®-Praxishandbuch ABAP Core Data Services (CDS)*. Espresso Tutorials.

Danho, J., & Dörrschuck, P. (2024). *Schnelleinstieg in das SAP ABAP RESTful Application Programming Model (RAP)*. Espresso Tutorials.

Drumm, C. (2023). *ABAP Steampunk*. Espresso Tutorials.

Software Heroes. (n.d.). ABAP RAP – Excel Datei laden. Retrieved from *https://software-heroes.com/blog/abap-rap-excel-datei-laden*. Accessed Date: 29.11.2024

SAP ABAP Central. (n.d.). Streams in RAP: Uploading PDF, Excel, and more. Retrieved *from https://sapabapcentral.blogspot.com/2022/09/streams-in-rap-uploading-pdf-excel-and.html.* Accessed Date: 29.11.2024

SAP Help Portal. (n.d.). Excel Upload in ABAP RAP. Retrieved from *https://help.sap.com/docs/ABAP_PLATFORM_NEW.* Accessed Date: 29.11.2024

SAP Community. (n.d.). Excel Upload using SAP RAP only. Retrieved from *https://community.sap.com/t5/technology-blogs-by-members/upload-excel-using-sap-rap-only/bap/13666565.* Accessed Date: 29.11.2024

Software Heroes. (n.d.). ABAP RAP Messages – Error Handling. Retrieved *from https://software-heroes.com/en/blog/abap-rap-messages.* Accessed Date: 29.11.2024

SAP Help Portal. (n.d.). SAP Application Extension Methodology Overview. Retrieved from *https://help.sap.com/docs/sap-btp-guidance-framework/sap-application-extension-methodology/sap-application-extension-methodology-overview.* Accessed Date: 30.11.2024

SAP Learning. (n.d.). Practicing Clean Core Extensibility for SAP S/4HANA Cloud – Introducing the Clean Core Approach. Retrieved from *https://learning.sap.com/learning-journeys/practicing-clean-core-extensibility-for-sap-s-4hana-cloud/introducing-the-clean-core-approach_fcb6c662-7041-4c99-88bd-345636fae7f3*. Accessed Date: 30.11.2024

Software Heroes. (n.d.). ABAP RAP Validation. Retrieved *from https://software-heroes.com/en/blog/abap-rap-validation*. Accessed Date: 03.12.2024

CloudDNA. (n.d.). SAP Fiori Elements – List Reports. Retrieved from *https://www.clouddna.at/sapui5-fiori/sap-fiori-elements-abapconf/.* Accessed Date: 08.12.2024

SAP. (n.d.). SAP UI5 Freestyle Application SDK. Retrieved from *https://sapui5.hana.ondemand.com/sdk/#/.* Accessed Date: 20.12.2024

Lange, N. (2020). Excel-Arbeitsmappe mit ABAP erstellen. Retrieved from *https://www.lange-net.de/2020/02/18/excelarbeitsmappe-mit-abap-erstellen/.* Accessed Date: 26.12.2024

SAP Community. (n.d.). Building the Field Catalog Manually in ALV. Retrieved from *https://community.sap.com/t5/application-development-discussions/building-the-filed-catalog-manually-in-alv/m-p/1270148.* Accessed Date: 26.12.2024

SAP Help Portal. (n.d.). Excel Export in SAP BTP ABAP Environment. Retrieved from *https://help.sap.com/docs/sap-btp-abap-environment/abap-environment/xlsx-write-access. Accessed* Date: 27.12.2024

Software Heroes. (n.d.). ABAP XCO Excel Export. Retrieved *from https://software-heroes.com/en/blog/abap-xco-excel-en.* Accessed Date: 27.12.2024

Software Heroes. (n.d.). Restoring of Old File Names in ABAP RAP. Retrieved from *https://software-heroes.com/blog/abap-rap-entity-manipulation-language*. Accessed Date: 27.12.2024

SAP Help Portal. (n.d.). Authorization Basics in SAP Business Technology Platform (BTP). Retrieved *from https://help.sap.com/docs/btp/sap-business-technology-platform/authorization-basics.* Accessed Date: 03.01.2025

SAP Help Portal. (n.d.). Implementing Global Authorization in ABAP RAP. Retrieved from *https://help.sap.com/docs/abap-cloud/abap-rap/implementing-global-authorization.* Accessed Date: 04.01.2025

SAP Developers. (n.d.). ABAP Authorization Model – Creating Authorizations in ABAP Environment. Retrieved from *https://developers.sap.com/group.abap-env-authorizations.html.* Accessed Date: 04.01.2025

SAP Community. (n.d.). ABAP RAP Custom Entity Call External URL. Retrieved from *https://community.sap.com/t5/technology-q-a/abap-rap-custom-entity-call-external-url/qaq-p/12408104.* Accessed Date: 22.01.2025

SAP Community. (n.d.). URL to Access a Transaction via Web. Retrieved from *https://community.sap.com/t5/technology-q-a/url-to-access-a-transaction-via-web/qaq-p/6943653.* Accessed Date: 22.01.2025

SAP Help Portal. (n.d.). Transaction Access via Web. Retrieved from *https://help.sap.com/docs/SAP_NETWEAVER_740.* Accessed Date: 22.01.2025

SAP. (n.d.). WebGUI SAP [Video]. *Retrieved from https://www.youtube.com/watch?v=F-uMiOlaESM.* Accessed Date: 22.01.2025

ERP-Up. (n.d.). Logging in SAP ERP. Retrieved from *https://erp-up.de/protokollierung-sap-erp/.* Accessed Date: 22.01.2025

SAP Community. (n.d.). Second Screen WebGUI SAP – Passing Parameters. Retrieved from *https://community.sap.com/t5/technology-q-a/suggestion-needed-to-pass-parameters-for-second-screen-in-sap-webgui-url/qaq-p/11215897.* Accessed Date: 22.01.2025