MAY 29 , 2025 | 6 PM CEST | LISBON, PORTUGAL

MENDIX MEETUP

# Enhance & Empower: Quality Control & Pluggable Widgets

João Lopes
Senior Mendix Consultant
@ Stoneworx

Mark Nauta
Senior Mendix Consultant
@ Stoneworx

mx mendix + STONEWORX

# Mendix Pluggable Widgets

## From Idea to Execution

STONEXWORX

# The Need For A Widget

- (…) sometimes your app requires a widget outside of this set. **To support a more advanced UI pattern or create app-specific interactions** (…)
- Your new pluggable widget can be used while modeling pages alongside standard Mendix components. It can also be shared between multiple apps and distributed through the Marketplace.

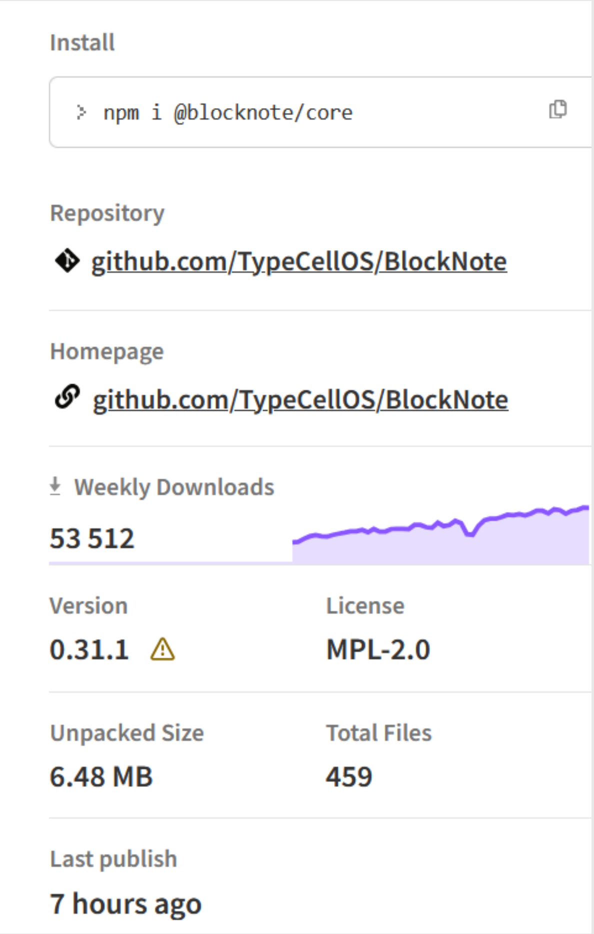
Source: Pluggable Widgets API | Mendix Documentation

# Before We Start

- Node.js (LTS)
- React / Javascript / Typescript
- IDE (VS Code)
- Mendix Pluggable Widgets Generator
  - npm install -g yo @mendix/generator-widget
- Github
- Browser Extension: React Developer Tools

# Choosing the Right NPM Package

- Check Popularity (Downloads & Github Stars)
- Maintenance
- Documentation
- Check Pricing / License



Install

```
> npm i @blocknote/core
```

Repository

github.com/TypeCellOS/BlockNote

Homepage

github.com/TypeCellOS/BlockNote

Weekly Downloads

53 512

Version

0.31.1 ⚠

License

MPL-2.0

Unpacked Size

6.48 MB

Total Files

459

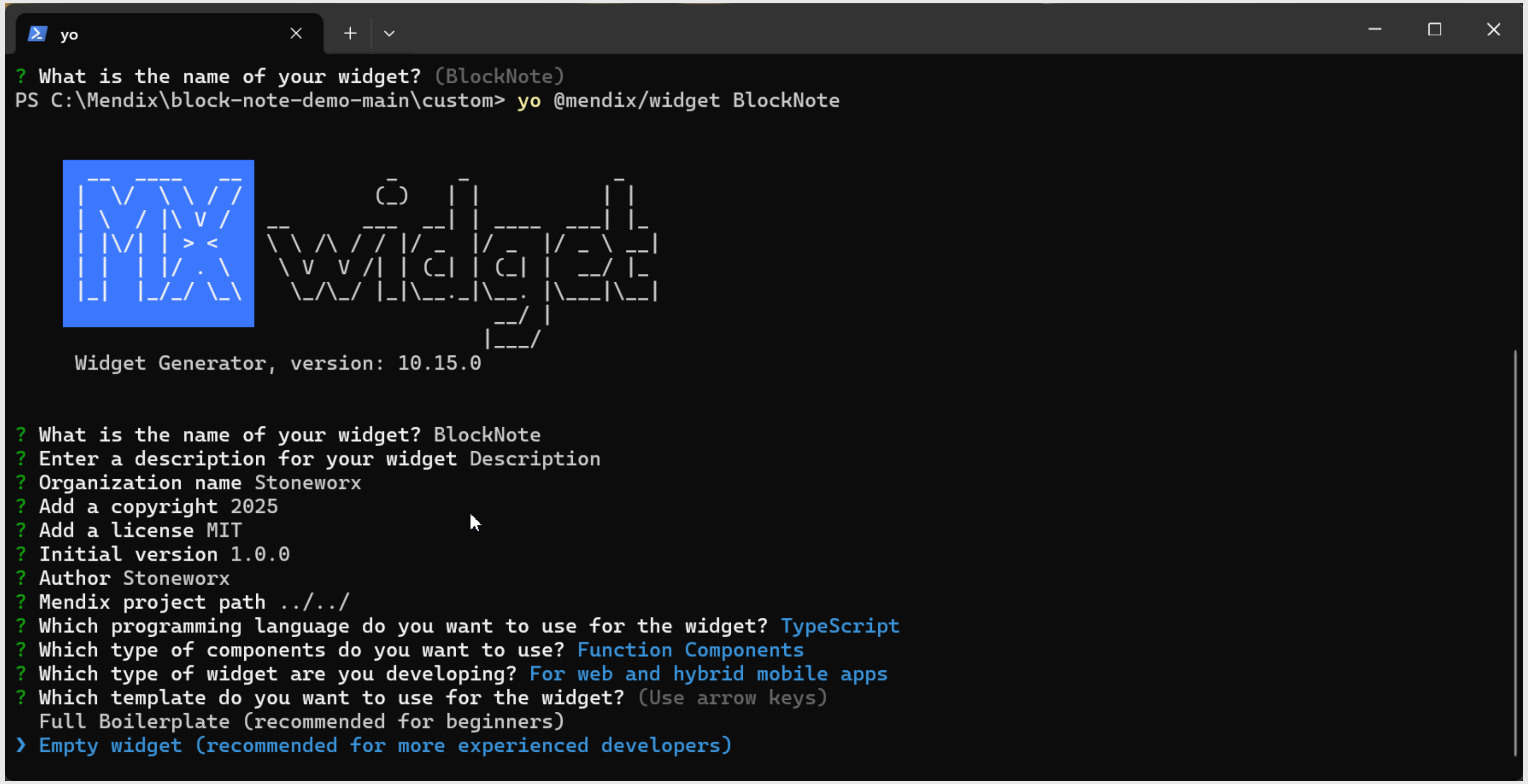
Last publish

7 hours ago

STONEXWORX

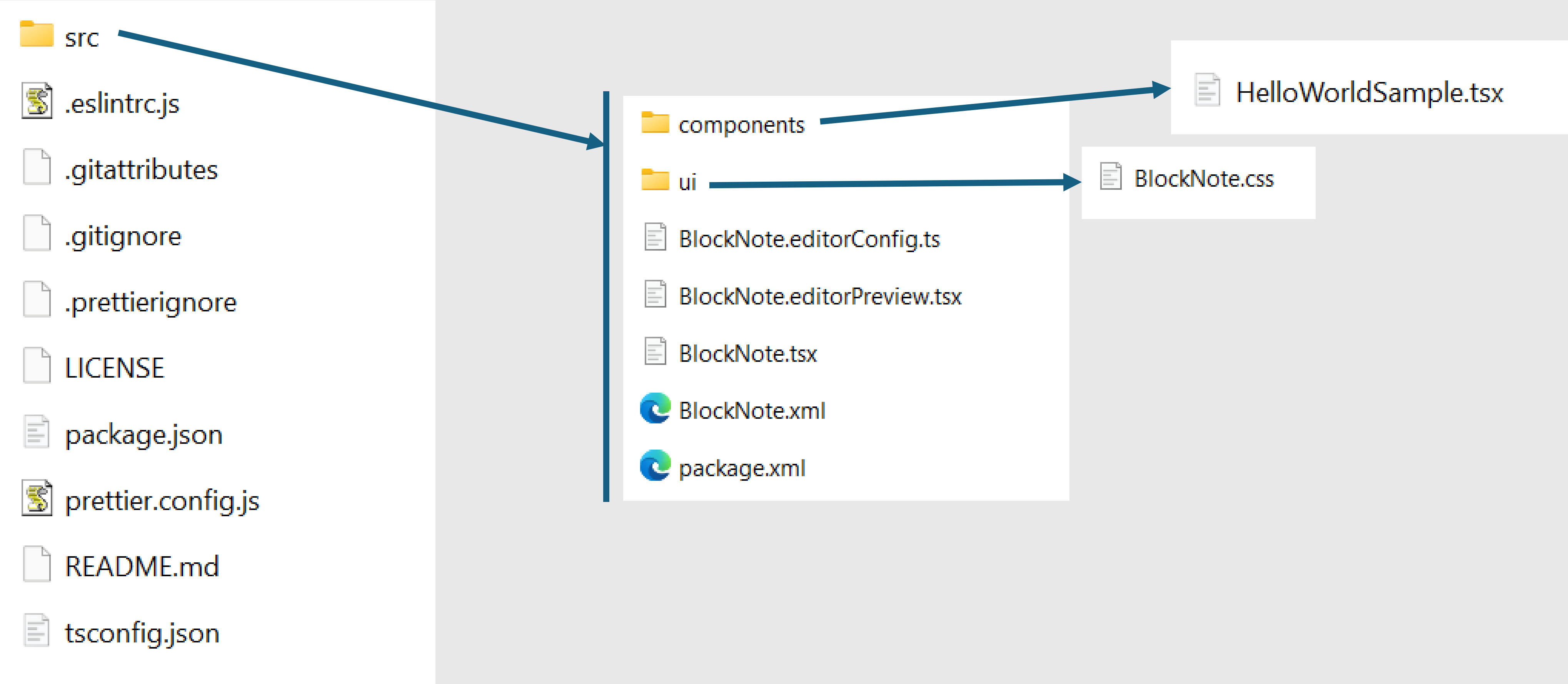
# Scaffold Widget

- yo @mendix/widget BlockNote



```
? What is the name of your widget? (BlockNote)
PS C:\Mendix\block-note-demo-main\custom> yo @mendix/widget BlockNote


      __  __  __   __          _     _            _
     |  \/  | \ \ / /         (_)   | |          | |
     | \  / |  \ V /  __      ___  __| | __ _  ___| |_
     | |\/| |   > <   \ \ /\ / / |/ _` |/ _` |/ _ \ __|
     | |  | |  / . \   \ V  V /| | (_| | (_| |  __/ |_
     |_|  |_| /_/ \_\   \_/\_/ |_|\__,_|\__, |\___|\__|
                                         __/ |
                                        |___/
     Widget Generator, version: 10.15.0

? What is the name of your widget? BlockNote
? Enter a description for your widget Description
? Organization name Stoneworx
? Add a copyright 2025
? Add a license MIT
? Initial version 1.0.0
? Author Stoneworx
? Mendix project path ../../
? Which programming language do you want to use for the widget? TypeScript
? Which type of components do you want to use? Function Components
? Which type of widget are you developing? For web and hybrid mobile apps
? Which template do you want to use for the widget? (Use arrow keys)
  Full Boilerplate (recommended for beginners)
❯ Empty widget (recommended for more experienced developers)
```

# Folder Structure Overview



src
.eslintrc.js
.gitattributes
.gitignore
.prettierignore
LICENSE
package.json
prettier.config.js
README.md
tsconfig.json

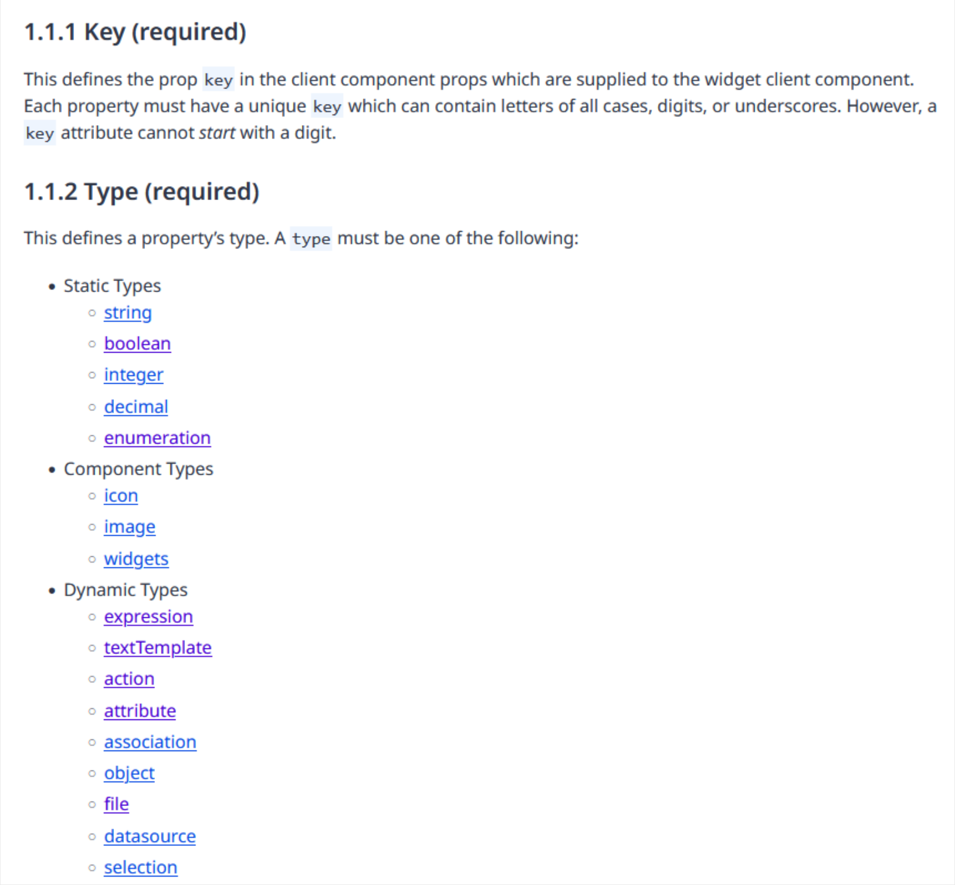
components
ui
BlockNote.editorConfig.ts
BlockNote.editorPreview.tsx
BlockNote.tsx
BlockNote.xml
package.xml

HelloWorldSample.tsx

BlockNote.css

# Package.XML

- Property Types
- API / Docs Mendix



### 1.1.1 Key (required)

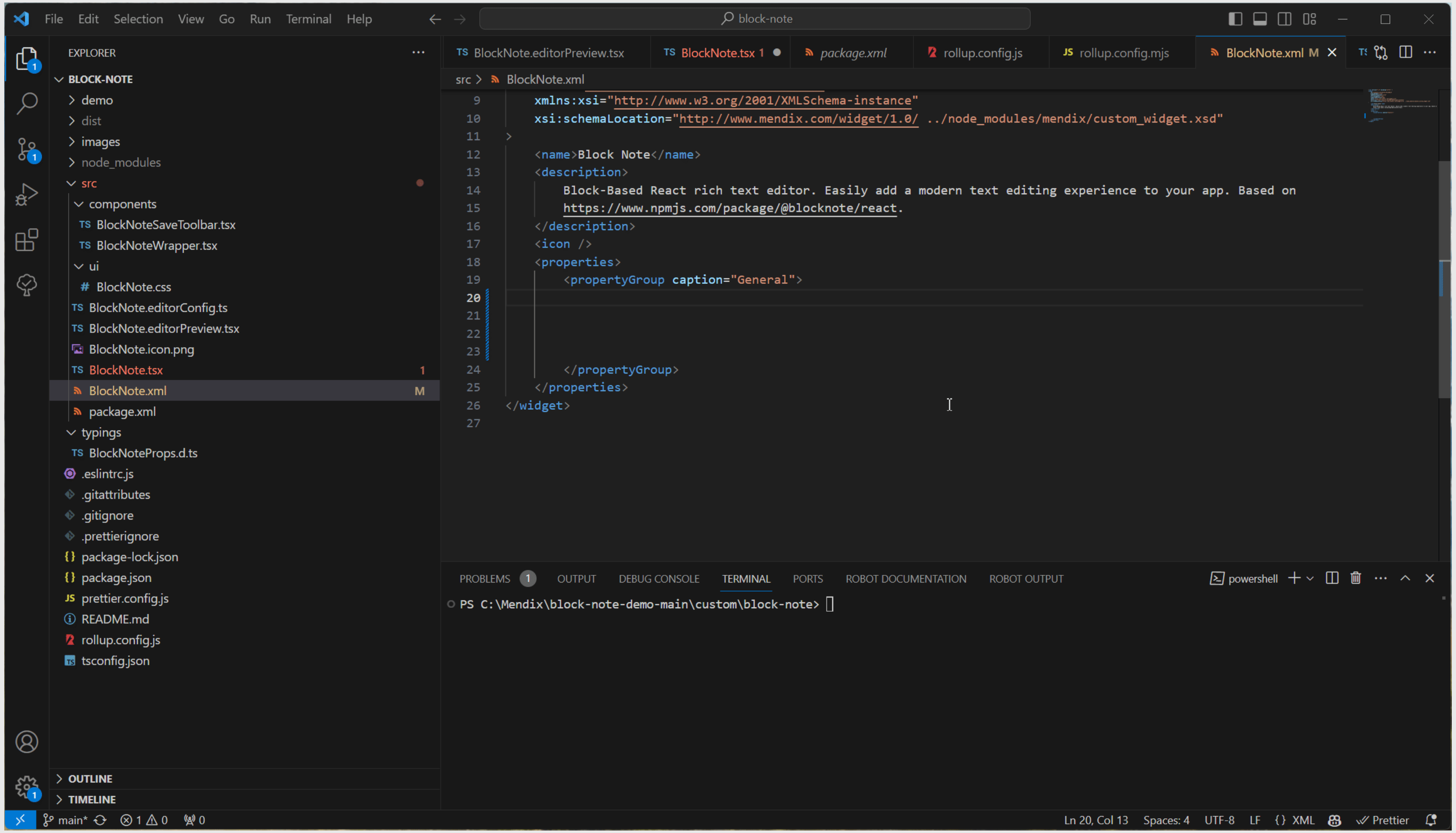
This defines the prop `key` in the client component props which are supplied to the widget client component. Each property must have a unique `key` which can contain letters of all cases, digits, or underscores. However, a `key` attribute cannot *start* with a digit.

### 1.1.2 Type (required)

This defines a property's type. A `type` must be one of the following:

- Static Types
  - string
  - boolean
  - integer
  - decimal
  - enumeration
- Component Types
  - icon
  - image
  - widgets
- Dynamic Types
  - expression
  - textTemplate
  - action
  - attribute
  - association
  - object
  - file
  - datasource
  - selection

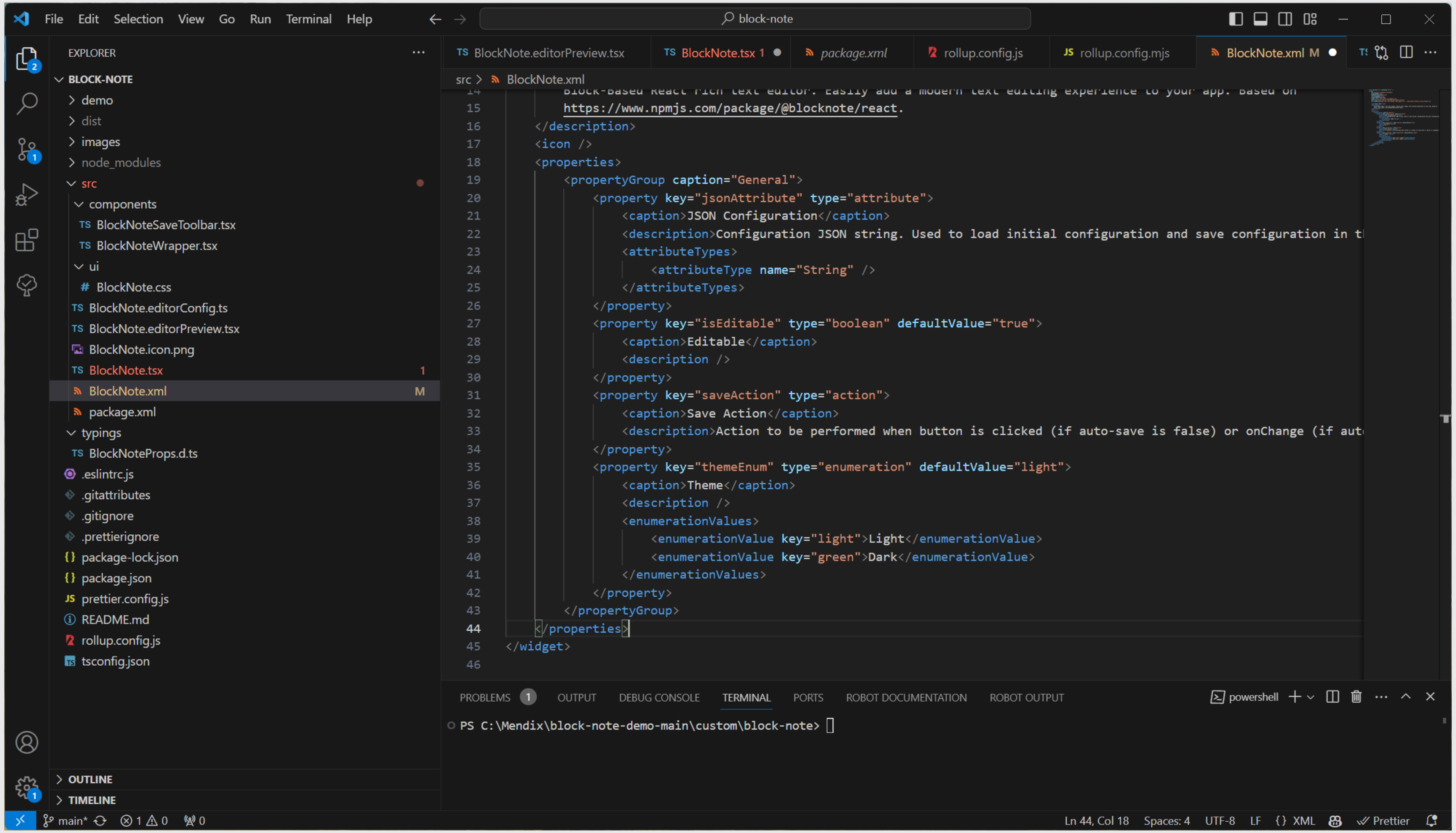
# Package.XML



```xml
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.mendix.com/widget/1.0/ ../node_modules/mendix/custom_widget.xsd"
>
    <name>Block Note</name>
    <description>
        Block-Based React rich text editor. Easily add a modern text editing experience to your app. Based on
        https://www.npmjs.com/package/@blocknote/react.
    </description>
    <icon />
    <properties>
        <propertyGroup caption="General">



        </propertyGroup>
    </properties>
</widget>

```

```
PS C:\Mendix\block-note-demo-main\custom\block-note>
```

# Package.XML

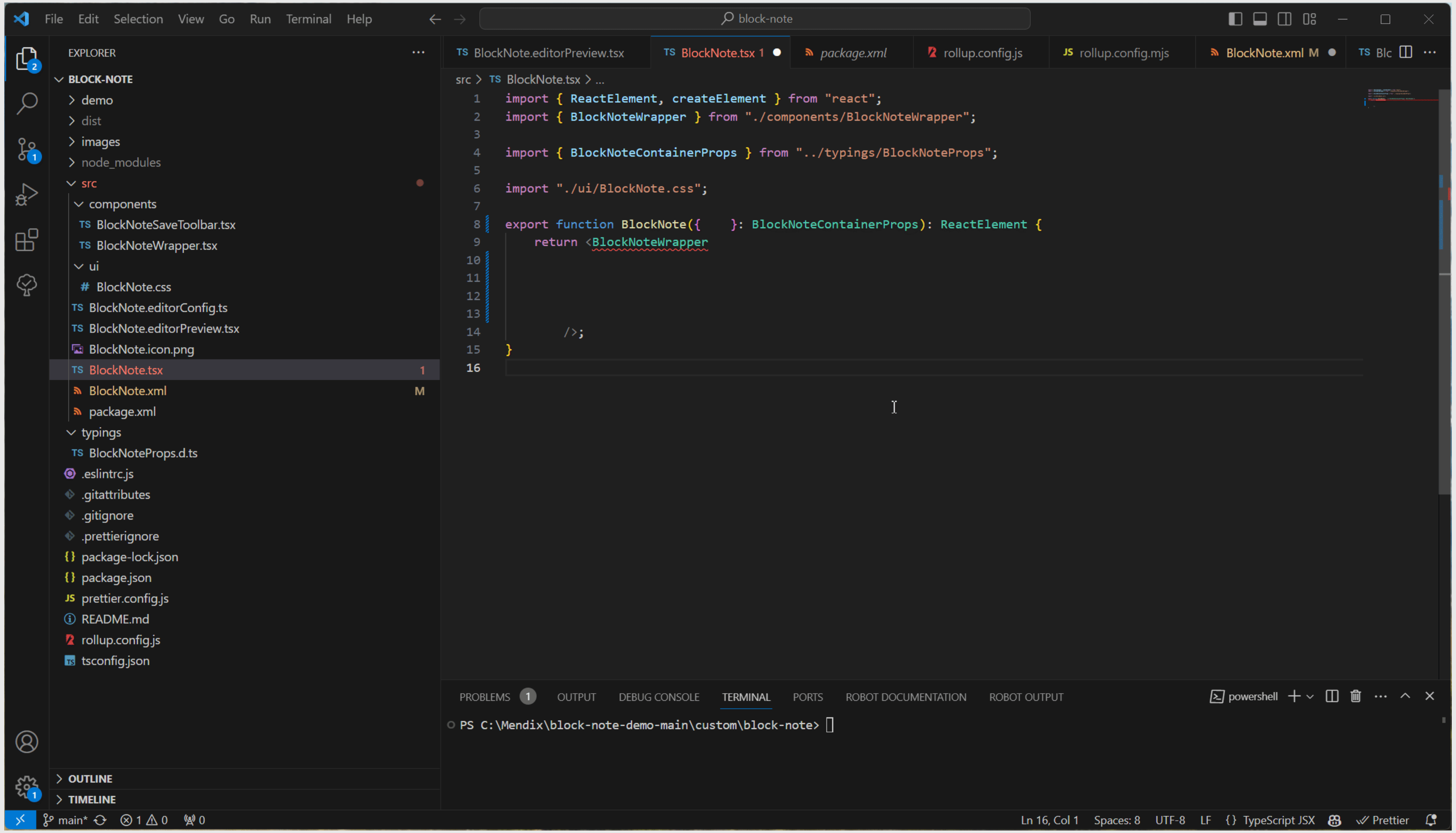


```xml
        Block-Based React rich text editor. Easily add a modern text editing experience to your app. Based on
        https://www.npmjs.com/package/@blocknote/react.
    </description>
    <icon />
    <properties>
        <propertyGroup caption="General">
            <property key="jsonAttribute" type="attribute">
                <caption>JSON Configuration</caption>
                <description>Configuration JSON string. Used to load initial configuration and save configuration in t
                <attributeTypes>
                    <attributeType name="String" />
                </attributeTypes>
            </property>
            <property key="isEditable" type="boolean" defaultValue="true">
                <caption>Editable</caption>
                <description />
            </property>
            <property key="saveAction" type="action">
                <caption>Save Action</caption>
                <description>Action to be performed when button is clicked (if auto-save is false) or onChange (if aut
            </property>
            <property key="themeEnum" type="enumeration" defaultValue="light">
                <caption>Theme</caption>
                <description />
                <enumerationValues>
                    <enumerationValue key="light">Light</enumerationValue>
                    <enumerationValue key="green">Dark</enumerationValue>
                </enumerationValues>
            </property>
        </propertyGroup>
    </properties>
</widget>
```

```
PS C:\Mendix\block-note-demo-main\custom\block-note>
```

STONEWORX

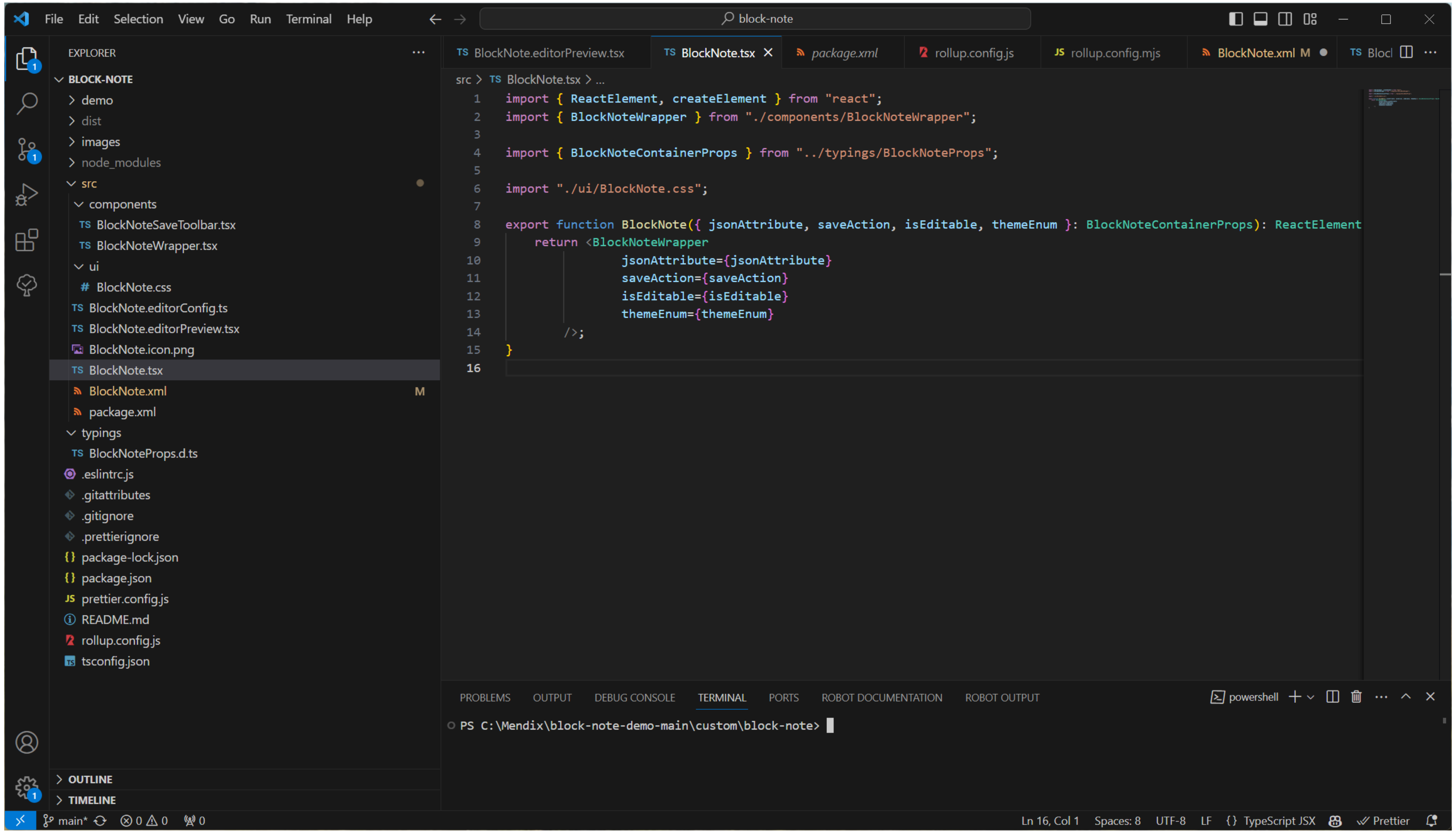
# Widget’s Main Component (.TSX File)



```tsx
import { ReactElement, createElement } from "react";
import { BlockNoteWrapper } from "./components/BlockNoteWrapper";

import { BlockNoteContainerProps } from "../typings/BlockNoteProps";

import "./ui/BlockNote.css";

export function BlockNote({    }: BlockNoteContainerProps): ReactElement {
    return <BlockNoteWrapper




        />;
}

```

```
PS C:\Mendix\block-note-demo-main\custom\block-note>
```

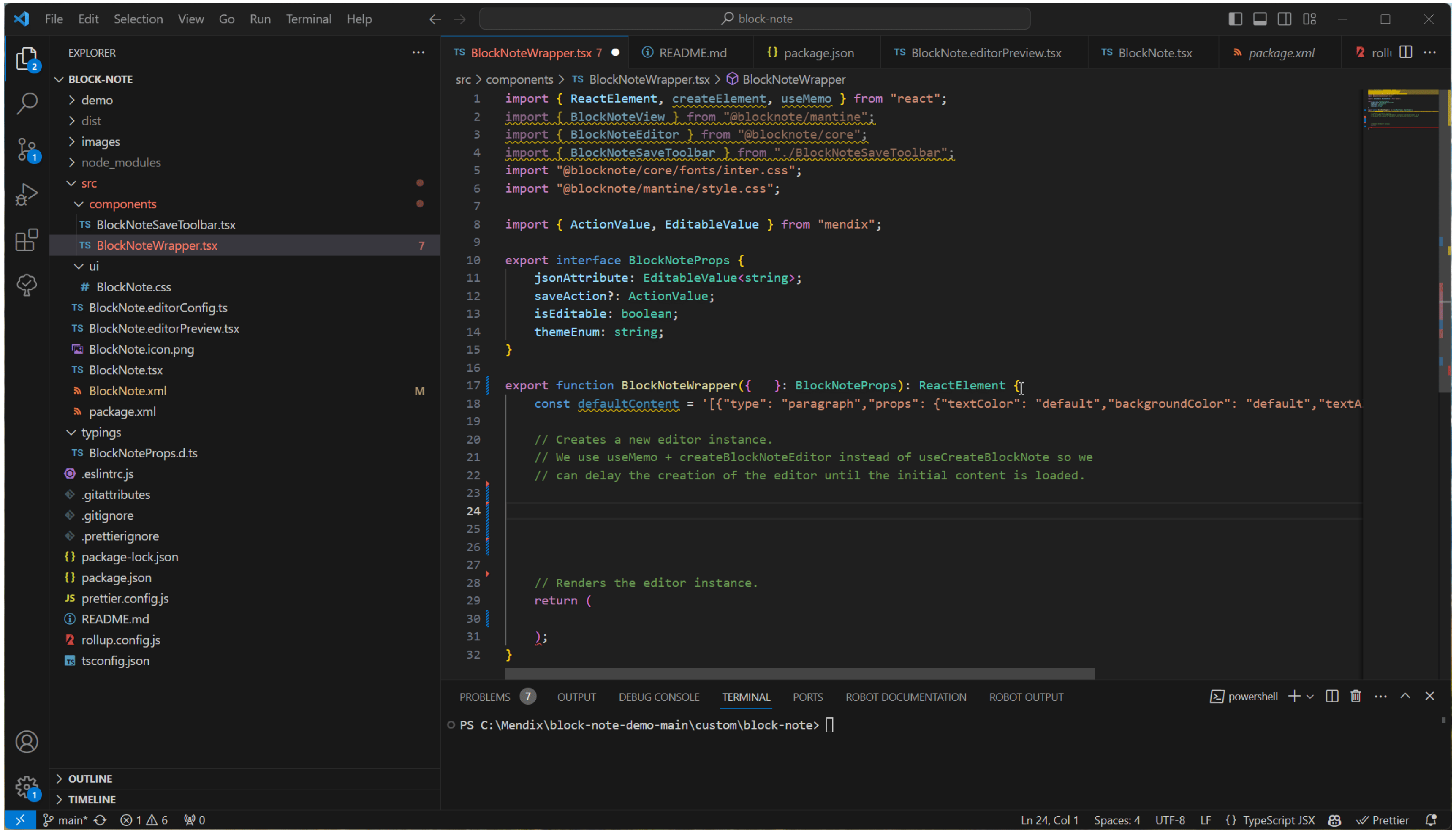
# Widget’s Main Component (.TSX File)



```tsx
import { ReactElement, createElement } from "react";
import { BlockNoteWrapper } from "./components/BlockNoteWrapper";

import { BlockNoteContainerProps } from "../typings/BlockNoteProps";

import "./ui/BlockNote.css";

export function BlockNote({ jsonAttribute, saveAction, isEditable, themeEnum }: BlockNoteContainerProps): ReactElement
    return <BlockNoteWrapper
                jsonAttribute={jsonAttribute}
                saveAction={saveAction}
                isEditable={isEditable}
                themeEnum={themeEnum}
        />;
}

```

```
PS C:\Mendix\block-note-demo-main\custom\block-note>
```

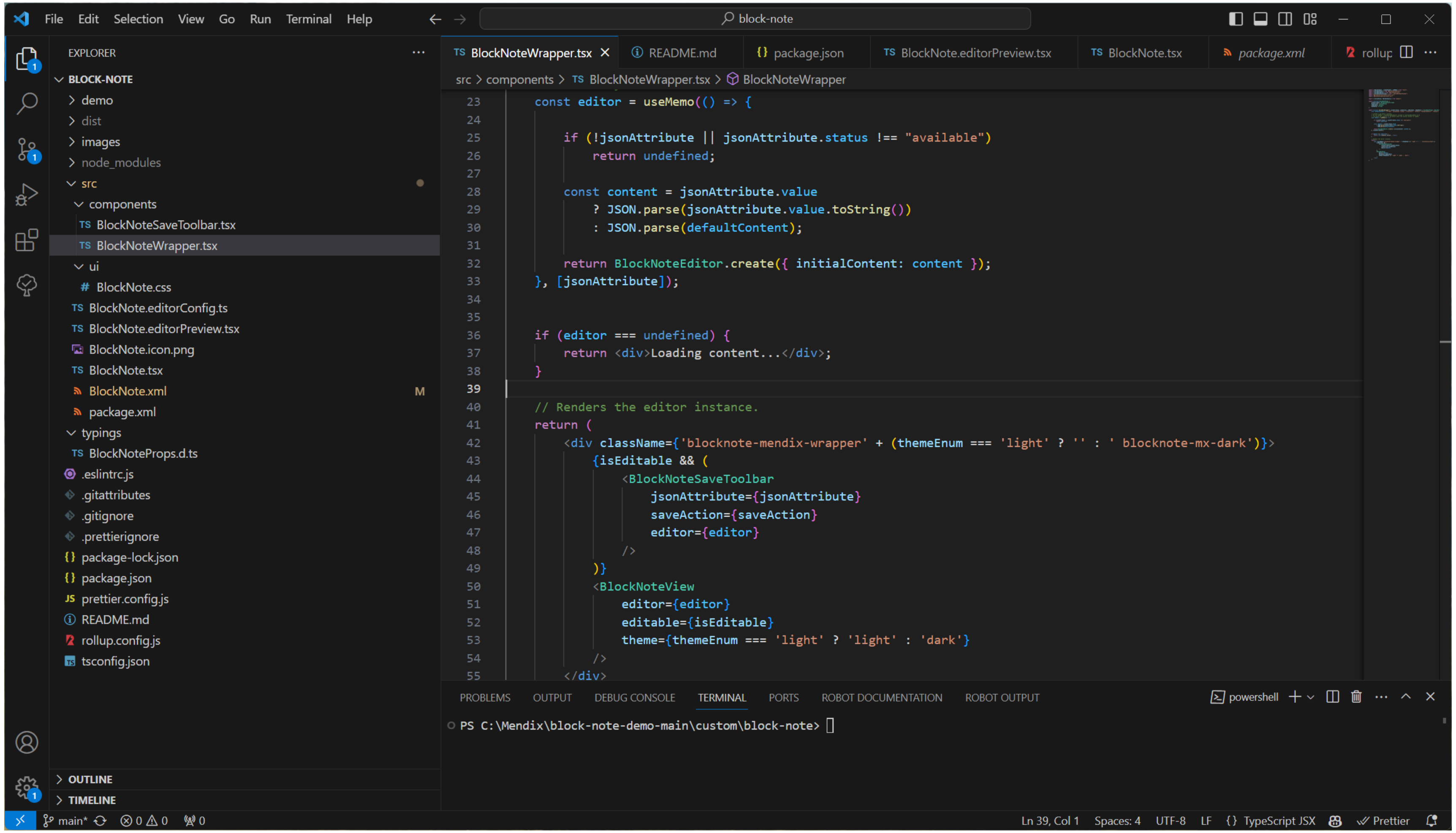
# Editor Wrapper Component (.TSX File)



```tsx
import { ReactElement, createElement, useMemo } from "react";
import { BlockNoteView } from "@blocknote/mantine";
import { BlockNoteEditor } from "@blocknote/core";
import { BlockNoteSaveToolbar } from "./BlockNoteSaveToolbar";
import "@blocknote/core/fonts/inter.css";
import "@blocknote/mantine/style.css";

import { ActionValue, EditableValue } from "mendix";

export interface BlockNoteProps {
    jsonAttribute: EditableValue<string>;
    saveAction?: ActionValue;
    isEditable: boolean;
    themeEnum: string;
}

export function BlockNoteWrapper({   }: BlockNoteProps): ReactElement {
    const defaultContent = '[{"type": "paragraph","props": {"textColor": "default","backgroundColor": "default","textA

    // Creates a new editor instance.
    // We use useMemo + createBlockNoteEditor instead of useCreateBlockNote so we
    // can delay the creation of the editor until the initial content is loaded.




    // Renders the editor instance.
    return (

    );
}
```

```
PS C:\Mendix\block-note-demo-main\custom\block-note>
```

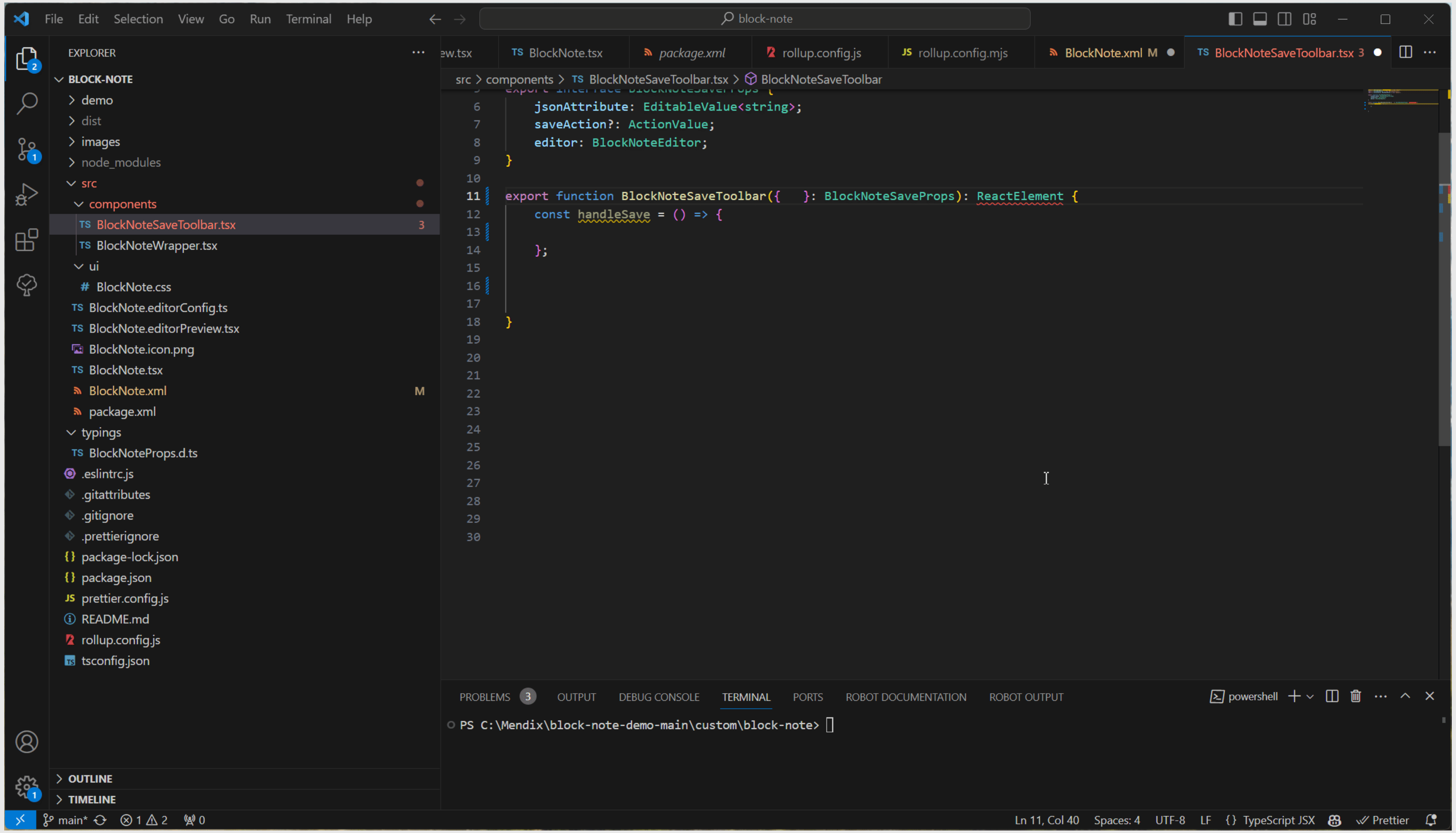
# Editor Wrapper Component (.TSX File)



```tsx
    const editor = useMemo(() => {

        if (!jsonAttribute || jsonAttribute.status !== "available")
            return undefined;

        const content = jsonAttribute.value
            ? JSON.parse(jsonAttribute.value.toString())
            : JSON.parse(defaultContent);

        return BlockNoteEditor.create({ initialContent: content });
    }, [jsonAttribute]);


    if (editor === undefined) {
        return <div>Loading content...</div>;
    }

    // Renders the editor instance.
    return (
        <div className={'blocknote-mendix-wrapper' + (themeEnum === 'light' ? '' : ' blocknote-mx-dark')}>
            {isEditable && (
                <BlockNoteSaveToolbar
                    jsonAttribute={jsonAttribute}
                    saveAction={saveAction}
                    editor={editor}
                />
            )}
            <BlockNoteView
                editor={editor}
                editable={isEditable}
                theme={themeEnum === 'light' ? 'light' : 'dark'}
            />
        </div>
```

```
PS C:\Mendix\block-note-demo-main\custom\block-note>
```

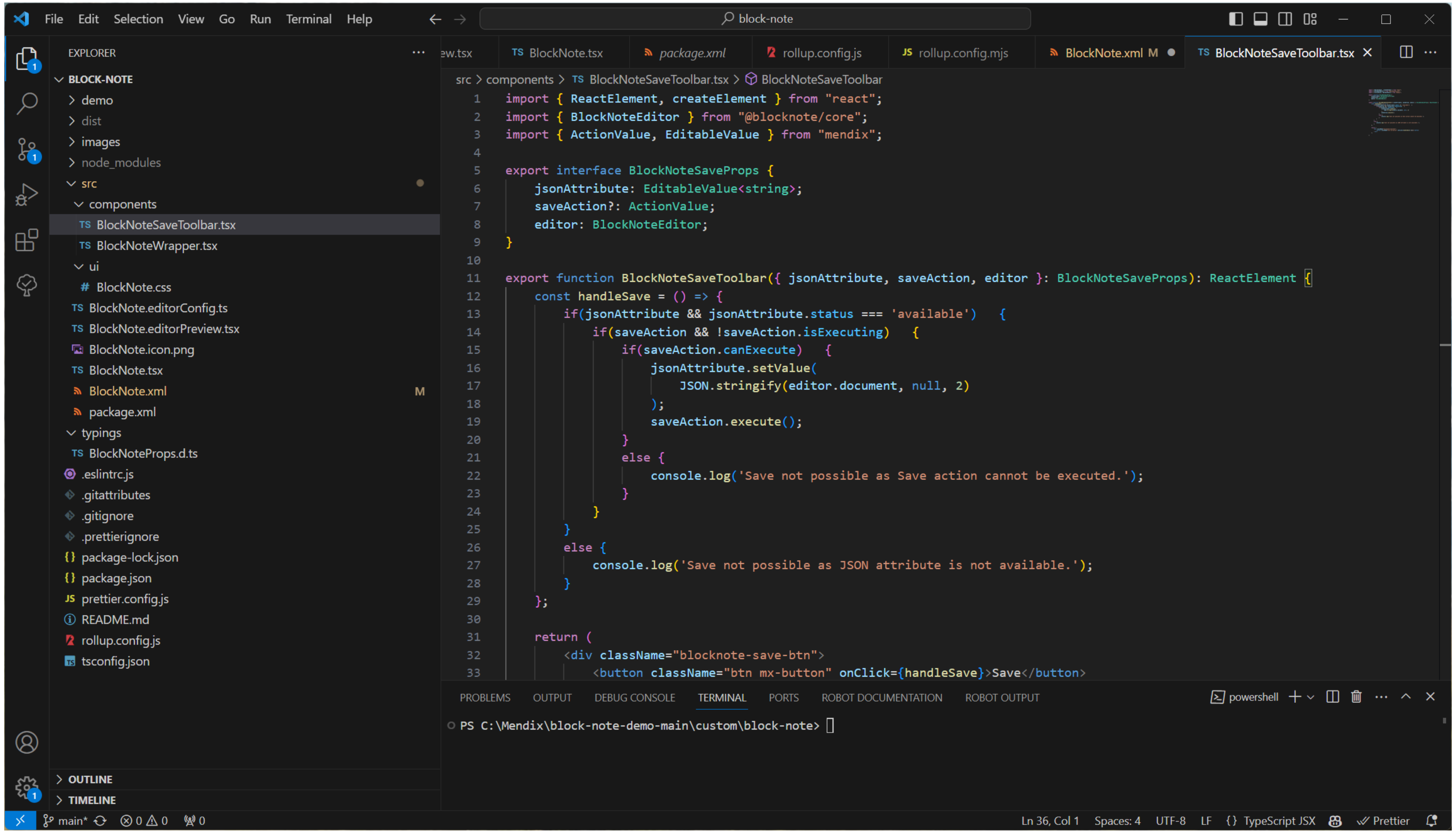
# Toolbar Component (.TSX File)



```tsx
export interface BlockNoteSaveProps {
    jsonAttribute: EditableValue<string>;
    saveAction?: ActionValue;
    editor: BlockNoteEditor;
}

export function BlockNoteSaveToolbar({   }: BlockNoteSaveProps): ReactElement {
    const handleSave = () => {

    };



}
```

```
PS C:\Mendix\block-note-demo-main\custom\block-note>
```

# Toolbar Component (.TSX File)



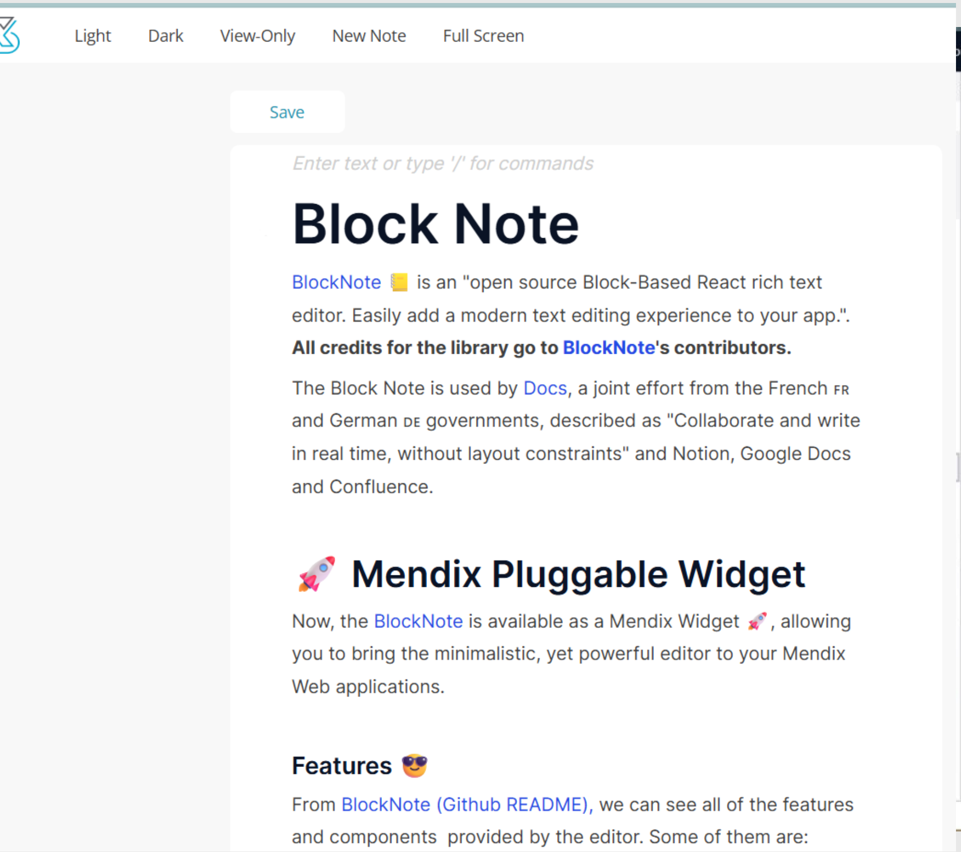
```tsx
import { ReactElement, createElement } from "react";
import { BlockNoteEditor } from "@blocknote/core";
import { ActionValue, EditableValue } from "mendix";

export interface BlockNoteSaveProps {
    jsonAttribute: EditableValue<string>;
    saveAction?: ActionValue;
    editor: BlockNoteEditor;
}

export function BlockNoteSaveToolbar({ jsonAttribute, saveAction, editor }: BlockNoteSaveProps): ReactElement {
    const handleSave = () => {
        if(jsonAttribute && jsonAttribute.status === 'available')   {
            if(saveAction && !saveAction.isExecuting)   {
                if(saveAction.canExecute)   {
                    jsonAttribute.setValue(
                        JSON.stringify(editor.document, null, 2)
                    );
                    saveAction.execute();
                }
                else {
                    console.log('Save not possible as Save action cannot be executed.');
                }
            }
        }
        else {
            console.log('Save not possible as JSON attribute is not available.');
        }
    };

    return (
        <div className="blocknote-save-btn">
            <button className="btn mx-button" onClick={handleSave}>Save</button>
```

```
PS C:\Mendix\block-note-demo-main\custom\block-note>
```

# Compile, Run, Test, Deploy

- npm start
- npm run build
  - Synchronize App Directory

# Widget’s Ready!



*Enter text or type '/' for commands*

# Block Note

BlockNote 📒 is an "open source Block-Based React rich text editor. Easily add a modern text editing experience to your app.". **All credits for the library go to BlockNote's contributors.**

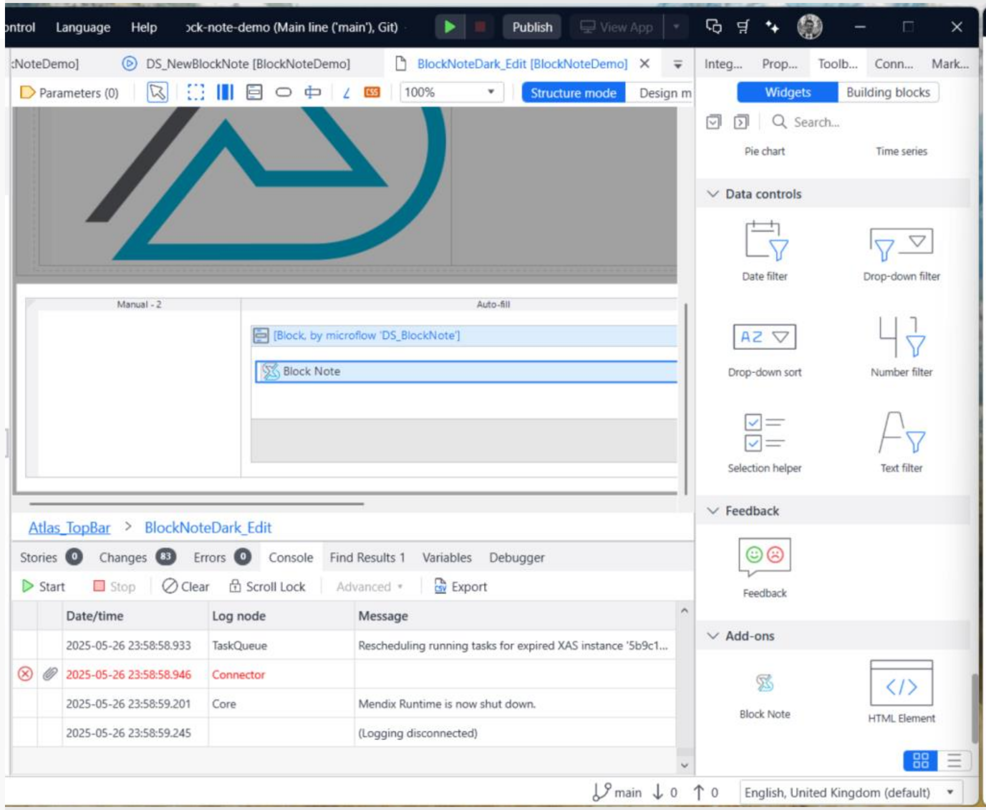
The Block Note is used by Docs, a joint effort from the French FR and German DE governments, described as "Collaborate and write in real time, without layout constraints" and Notion, Google Docs and Confluence.

## 🚀 Mendix Pluggable Widget

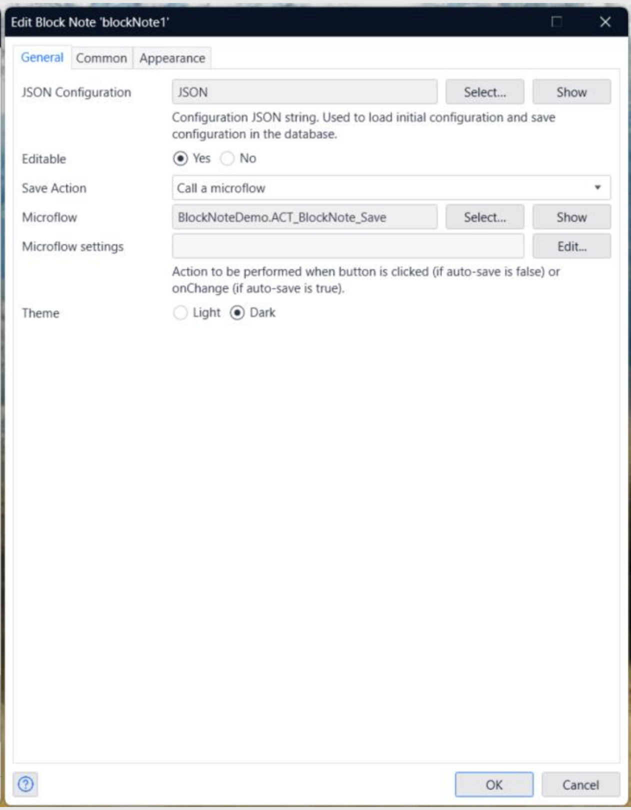
Now, the BlockNote is available as a Mendix Widget 🚀, allowing you to bring the minimalistic, yet powerful editor to your Mendix Web applications.

### Features 😎

From BlockNote (Github README), we can see all of the features and components  provided by the editor. Some of them are:



```
Date/time                 Log node     Message
2025-05-26 23:58:58.933   TaskQueue    Rescheduling running tasks for expired XAS instance '5b9c1...
2025-05-26 23:58:58.946   Connector
2025-05-26 23:58:59.201   Core         Mendix Runtime is now shut down.
2025-05-26 23:58:59.245                (Logging disconnected)
```



JSON Configuration: JSON

Configuration JSON string. Used to load initial configuration and save configuration in the database.

Editable: ● Yes ○ No

Save Action: Call a microflow

Microflow: BlockNoteDemo.ACT_BlockNote_Save

Microflow settings:

Action to be performed when button is clicked (if auto-save is false) or onChange (if auto-save is true).

Theme: ○ Light ● Dark

# WRITING HIGH QUALITY MENDIX APPS

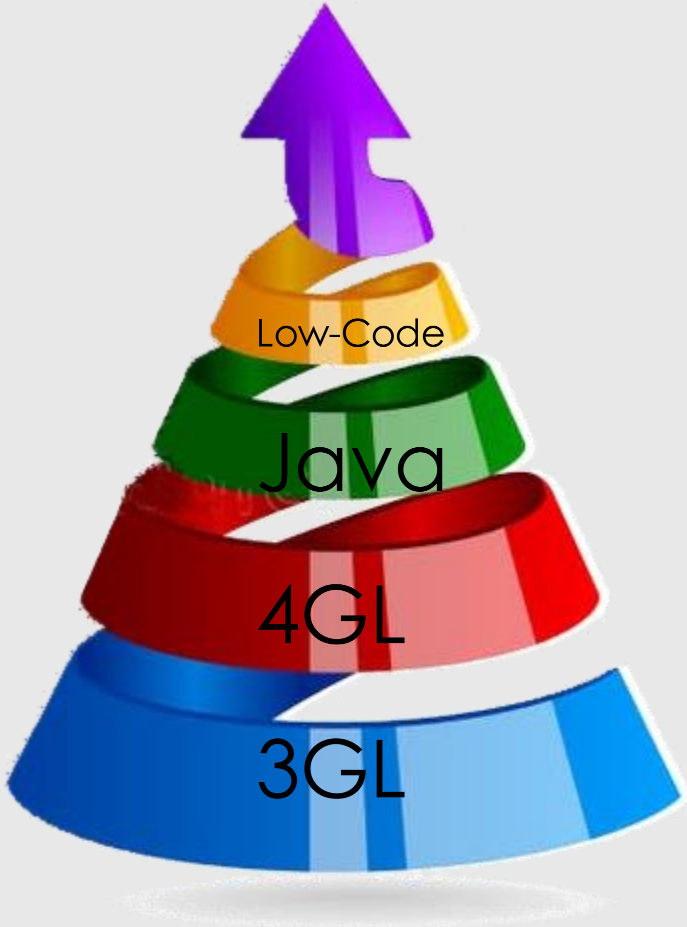
## And proving it with SAT!

# Contents

- Introduction
- Stoneworx Assessment Tool
- What is good quality
- Why another tool?
- SAT
- Demo
- Further development

## Introduction

- In IT since 1994
- 4 years of which in Mendix roles
- 1.5 year for Stoneworx
  - ➢ First months: no paid project +
  - ➢ Upskill trainee code assessments +
  - ➢ Wanted to learn about MX SDK’s



# Stoneworx Assessment Tool

- First take: Quality Assessment Tool
- Evolution:
  - Insight in quality issues
  - Security
  - Microflow structures

# What is good quality code?

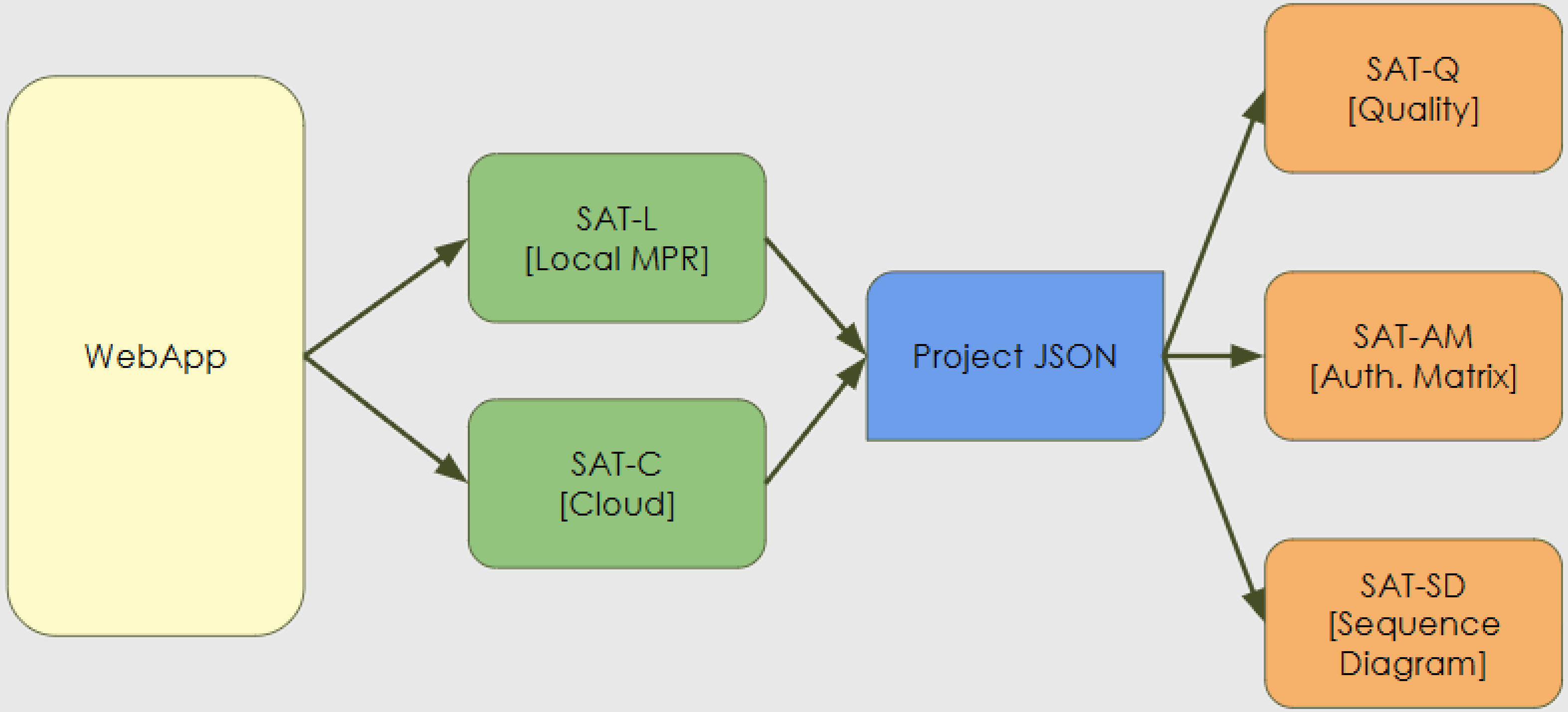
- Above all: Subjective as can be
- What you and your peers agree upon
  - Mendix community
  - Your company / the company you work for
  - Your project team

So we need a flexible set of quality controls that reflect the common ground of the project environment

# Why another tool

- This one is free / open source
- it is adaptable
- It runs local: direct feedback
- It can scan both local mpr files or on team server
- It installs easy: docker or nodejs
- It will get better and better

# Stoneworx Assessment Tool: Quick Overview



WebApp

SAT-L
[Local MPR]

SAT-C
[Cloud]

Project JSON

SAT-Q
[Quality]

SAT-AM
[Auth. Matrix]

SAT-SD
[Sequence
Diagram]

# Stoneworx Assessment Tool: Quality

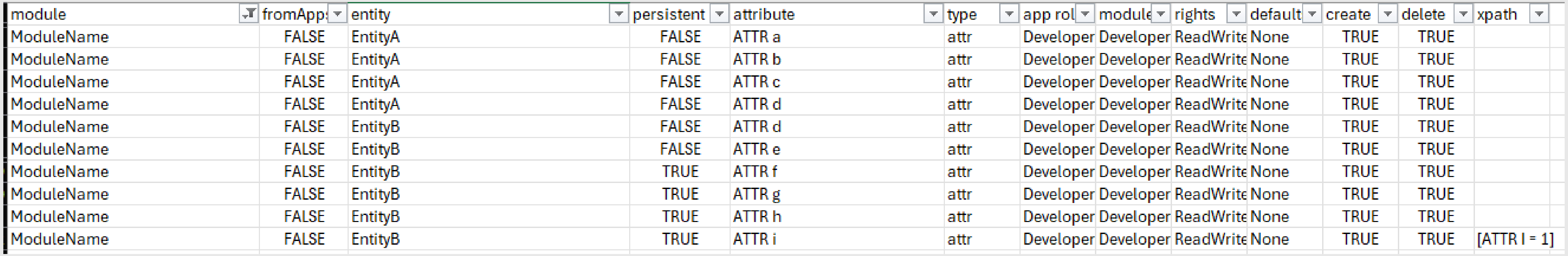
Improve quality:

- Reliability
- Security
- Maintainability
- Project Hygiene

| | | |
|---|---|---|
| Hygiene | NC1 | format: [PRE]_[Entity(s)]_description |
| Hygiene | NC2 | Prefix must be allowed |
| Hygiene | NC3 | Entity must exist |
| Hygiene | NC4 | Entity must exist in same module |
| Maintainability | CM1 | Commit not on correct hierarchy level(ACT or one level down) |
| Maintainability | IP1 | Show Page action outside of ACT |
| Maintainability | IP2 | Close Page action outside of ACT |
| Maintainability | ND1 | Nesting of subs too deep |
| Maintainability | CX1 | Too many actions in a single microflow |
| Maintainability | CX2 | Too complex microflow |
| Maintainability | CX3 | Too complex expression in Create/ Change Object |
| Maintainability | CX4 | Too complex expression in Create / Change Variable |
| Reliability | EH1 | Java Action without custom error handling |
| Security | DU1 | Demo users not allowed in production app |
| Security | PM1 | Microflow of this type should contain permissions |

# Stoneworx Assessment Tool: Security

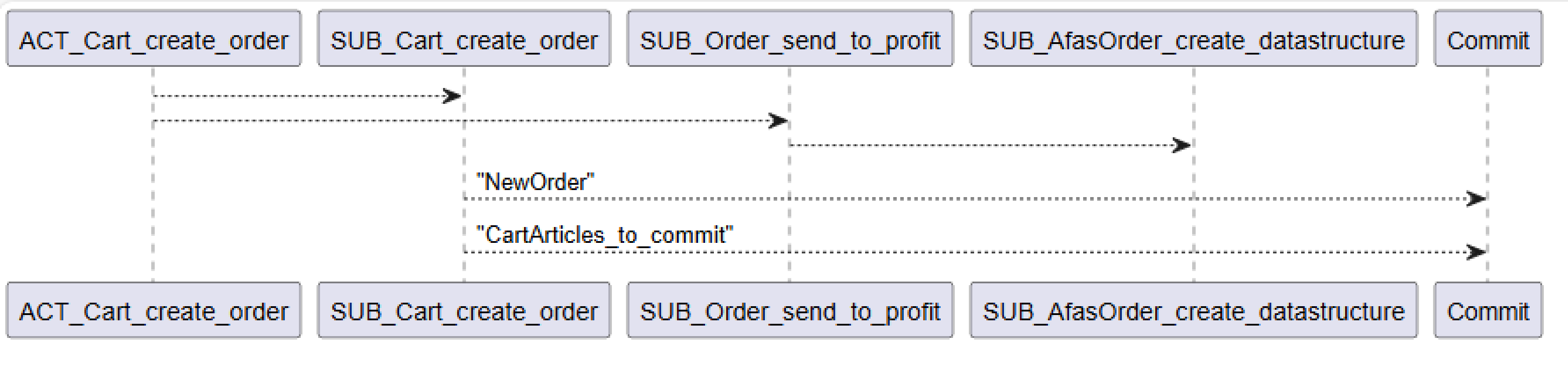
Export a csv with a access rules per Entity/Attribute. Create your own reports from there…



| module | fromApps | entity | persistent | attribute | type | app rol | module | rights | default | create | delete | xpath |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ModuleName | FALSE | EntityA | FALSE | ATTR a | attr | Developer | Developer | ReadWrite | None | TRUE | TRUE | |
| ModuleName | FALSE | EntityA | FALSE | ATTR b | attr | Developer | Developer | ReadWrite | None | TRUE | TRUE | |
| ModuleName | FALSE | EntityA | FALSE | ATTR c | attr | Developer | Developer | ReadWrite | None | TRUE | TRUE | |
| ModuleName | FALSE | EntityA | FALSE | ATTR d | attr | Developer | Developer | ReadWrite | None | TRUE | TRUE | |
| ModuleName | FALSE | EntityB | FALSE | ATTR d | attr | Developer | Developer | ReadWrite | None | TRUE | TRUE | |
| ModuleName | FALSE | EntityB | FALSE | ATTR e | attr | Developer | Developer | ReadWrite | None | TRUE | TRUE | |
| ModuleName | FALSE | EntityB | TRUE | ATTR f | attr | Developer | Developer | ReadWrite | None | TRUE | TRUE | |
| ModuleName | FALSE | EntityB | TRUE | ATTR g | attr | Developer | Developer | ReadWrite | None | TRUE | TRUE | |
| ModuleName | FALSE | EntityB | TRUE | ATTR h | attr | Developer | Developer | ReadWrite | None | TRUE | TRUE | |
| ModuleName | FALSE | EntityB | TRUE | ATTR i | attr | Developer | Developer | ReadWrite | None | TRUE | TRUE | [ATTR I = 1] |

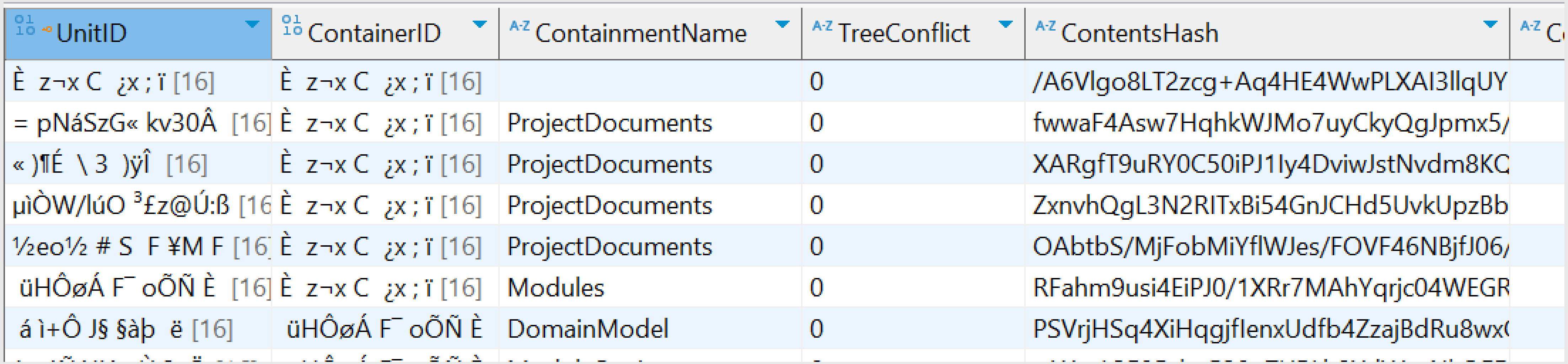
# Stoneworx Assessment Tool: Microflows

Generate a text file that can be rendered into a Sequence Diagram



# SAT-L

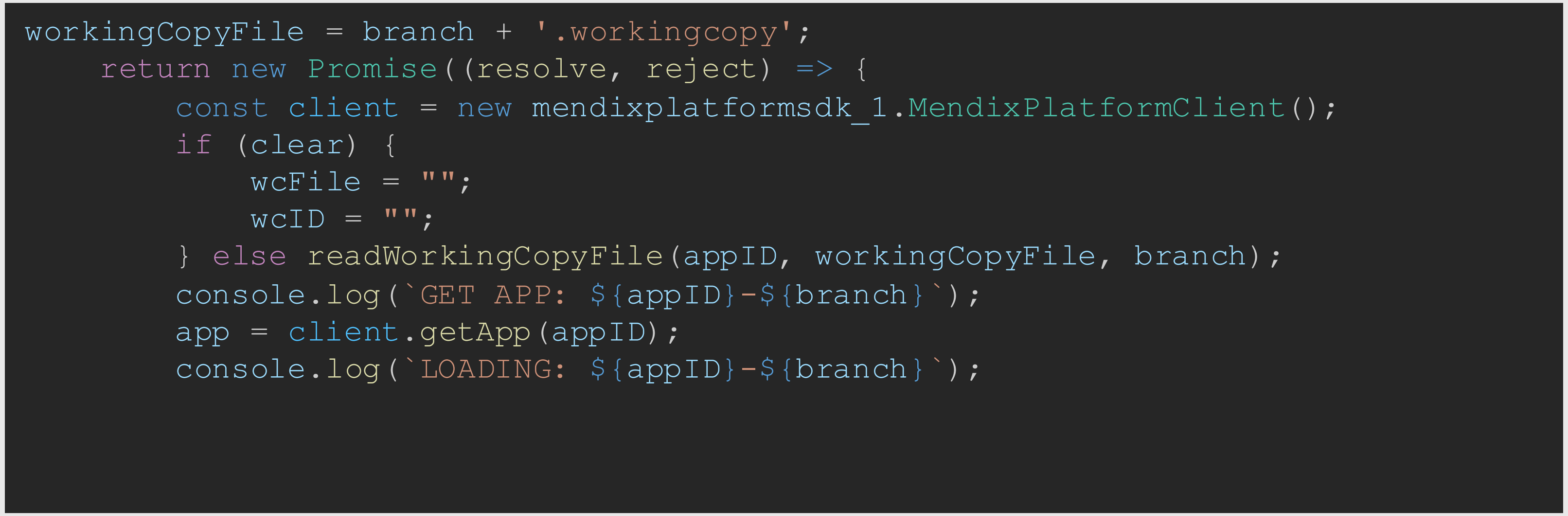
Parses the MPR which is actually a SQLite DB with a lot of bson info.



| UnitID | ContainerID | ContainmentName | TreeConflict | ContentsHash | C |
|---|---|---|---|---|---|
| È z¬x C ¿x ; ï [16] | È z¬x C ¿x ; ï [16] |  | 0 | /A6Vlgo8LT2zcg+Aq4HE4WwPLXAI3llqUY |  |
| = pNáSzG« kv30Â [16] | È z¬x C ¿x ; ï [16] | ProjectDocuments | 0 | fwwaF4Asw7HqhkWJMo7uyCkyQgJpmx5/ |  |
| « )¶É \ 3 )ÿÎ [16] | È z¬x C ¿x ; ï [16] | ProjectDocuments | 0 | XARgfT9uRY0C50iPJ1Iy4DviwJstNvdm8KC |  |
| µìÒW/lúO ³£z@Ú:ß [16 | È z¬x C ¿x ; ï [16] | ProjectDocuments | 0 | ZxnvhQgL3N2RITxBi54GnJCHd5UvkUpzBb |  |
| ½eo½ # S F ¥M F [16] | È z¬x C ¿x ; ï [16] | ProjectDocuments | 0 | OAbtbS/MjFobMiYflWJes/FOVF46NBjfJ06/ |  |
| üHÔøÁ F¯ oÕÑ È [16] | È z¬x C ¿x ; ï [16] | Modules | 0 | RFahm9usi4EiPJ0/1XRr7MAhYqrjc04WEGR |  |
| á ì+Ô J§ §àþ ë [16] | üHÔøÁ F¯ oÕÑ È | DomainModel | 0 | PSVrjHSq4XiHqgjflenxUdfb4ZzajBdRu8wx |  |

# SAT-C

Connects to working copy in cloud with Mendix SDK's.

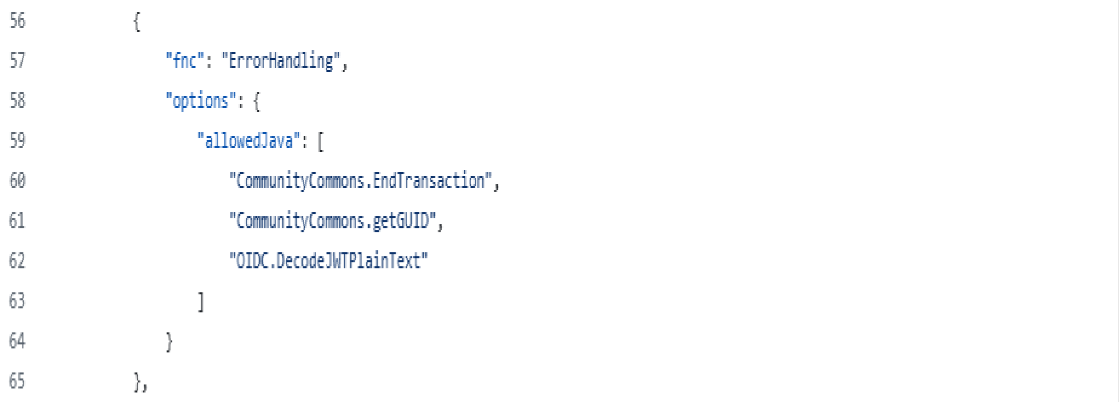


```
workingCopyFile = branch + '.workingcopy';
    return new Promise((resolve, reject) => {
        const client = new mendixplatformsdk_1.MendixPlatformClient();
        if (clear) {
            wcFile = "";
            wcID = "";
        } else readWorkingCopyFile(appID, workingCopyFile, branch);
        console.log(`GET APP: ${appID}-${branch}`);
        app = client.getApp(appID);
        console.log(`LOADING: ${appID}-${branch}`);
```

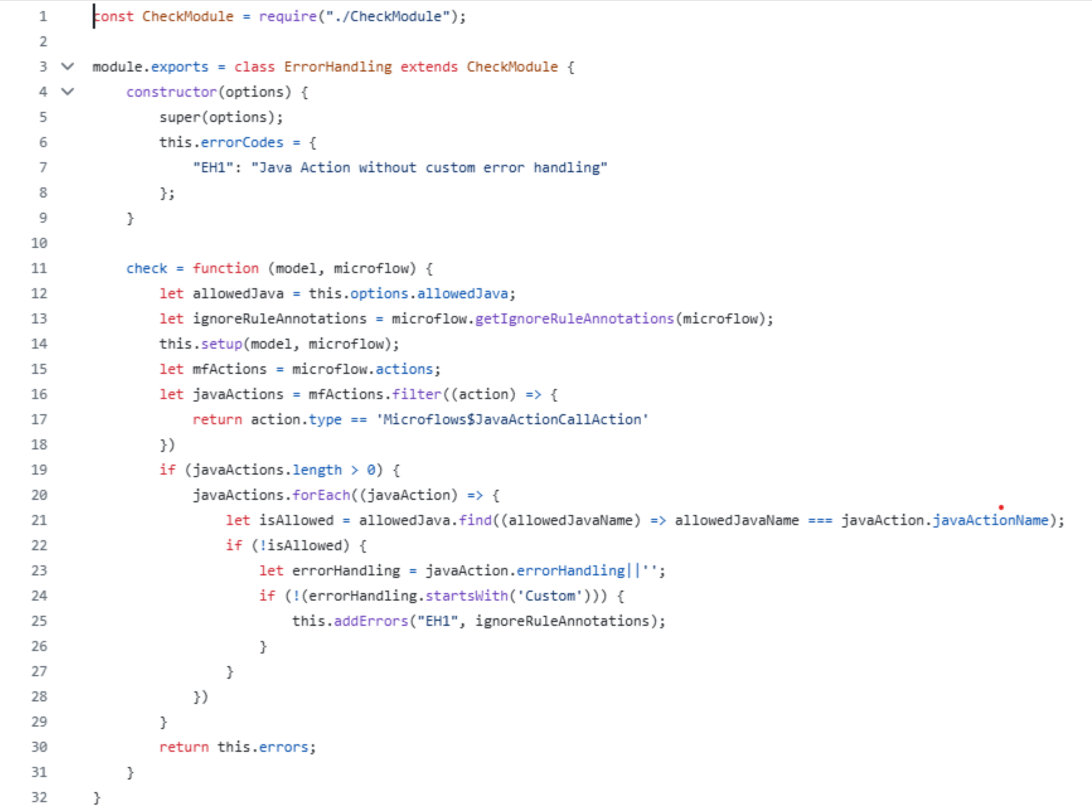
STONEXWORX

# SAT-Q: Configuration & Extensions

- Config in a json file: default.json
- Modules in nodejs modules in SAT-Q:

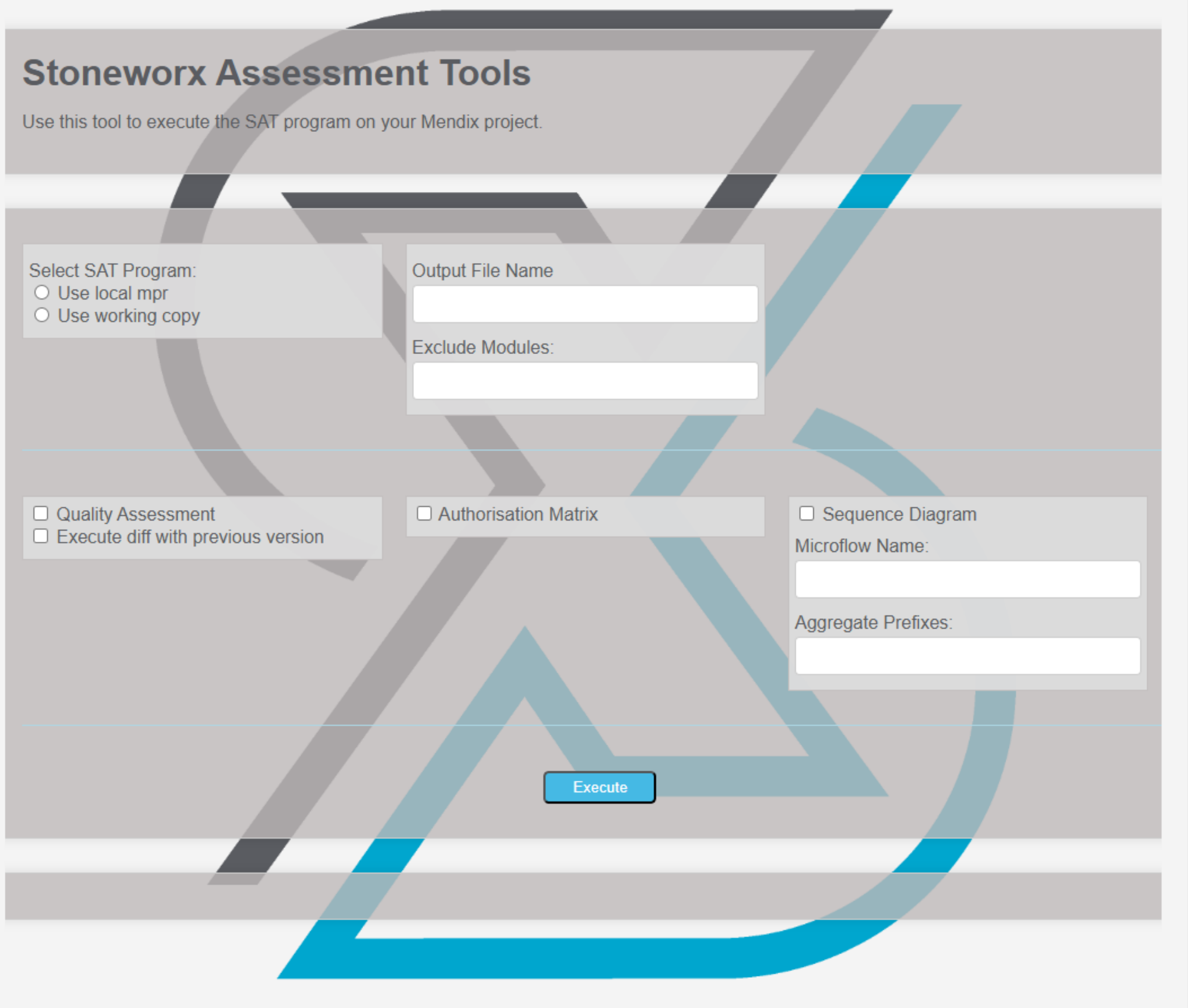


```
        {
            "fnc": "ErrorHandling",
            "options": {
                "allowedJava": [
                    "CommunityCommons.EndTransaction",
                    "CommunityCommons.getGUID",
                    "OIDC.DecodeJWTPlainText"
                ]
            }
        },
```



```
const CheckModule = require("./CheckModule");

module.exports = class ErrorHandling extends CheckModule {
    constructor(options) {
        super(options);
        this.errorCodes = {
            "EH1": "Java Action without custom error handling"
        };
    }

    check = function (model, microflow) {
        let allowedJava = this.options.allowedJava;
        let ignoreRuleAnnotations = microflow.getIgnoreRuleAnnotations(microflow);
        this.setup(model, microflow);
        let mfActions = microflow.actions;
        let javaActions = mfActions.filter((action) => {
            return action.type == 'Microflows$JavaActionCallAction'
        })
        if (javaActions.length > 0) {
            javaActions.forEach((javaAction) => {
                let isAllowed = allowedJava.find((allowedJavaName) => allowedJavaName === javaAction.javaActionName);
                if (!isAllowed) {
                    let errorHandling = javaAction.errorHandling||'';
                    if (!(errorHandling.startsWith('Custom'))) {
                        this.addErrors("EH1", ignoreRuleAnnotations);
                    }
                }
            })
        }
        return this.errors;
    }
}
```

# Demo!



## Stoneworx Assessment Tools

Use this tool to execute the SAT program on your Mendix project.

Select SAT Program:
○ Use local mpr
○ Use working copy

Output File Name

Exclude Modules:

□ Quality Assessment
□ Execute diff with previous version

□ Authorisation Matrix

□ Sequence Diagram

Microflow Name:

Aggregate Prefixes:

# Further development

- Add more checks (get inspired by the commercial tools
- Add more diagrams (like Class Diagram)
- Suggestions? -> Mail me! mnauta@stoneworx.nl

Slide deck available in Github Repo:
https://github.com/StoneworxNL/SAT