Calendar

Documentation

Tiogars

Table of contents

1. Calendar Documentation

This folder contains the full project documentation in a numbered hierarchical structure.

1.1 Table of Contents

1.2 Scope

These documents describe:

- Functional capabilities of the application
- How users move between edition and preview flows
- Screen capture organization for key pages
- Runtime and development dependencies used by the project
- Automated CI/CD pipeline, GitHub Pages deployment, and release management

2. 01 - Features

This document describes the main capabilities of the Calendar Print application.

2.1 Main Features

1. Calendar configuration form
2. Set a custom calendar title
3. Add optional free text displayed after the title (max 400 characters)
4. Add optional free text displayed below calendars (max 400 characters)
5. Select a month range using from/to month and year
6. Choose page orientation: portrait or landscape
7. Choose month grid layout: `4x3` or `3x4`
8. Choose the first day of the week: Monday or Sunday (default: Monday)
9. Open a share dialog to generate a link that serializes the current form values
10. Copy the generated link to clipboard or open an email draft containing the link
11. Dynamic calendar generation
12. Builds all months between the selected start and end month (inclusive)
13. Renders each month with weekday headers and date cells
14. Displays month names with an uppercase first letter in both English and French
15. A4 print preview
16. Displays the calendar in an A4-sized page container
17. Adjusts width and height based on selected orientation
18. On narrow screens the A4 page is automatically scaled down to fit the screen width, giving an accurate preview of the print layout on mobile devices
19. Printing workflow
20. One-click print action from preview mode
21. Hides edition and action controls in print media
22. Shows a footer with the printing date/time
23. Edit and iterate loop
24. Users can switch from preview back to edition mode
25. Existing configuration is preserved for quick adjustments
26. Mobile-friendly Floating Action Buttons (FAB)
27. On small screens (mobile), a SpeedDial FAB appears in the bottom-right corner of the edition form, giving quick access to the **Preview Calendar** and **Share configuration** actions without scrolling
28. On small screens in preview mode, two FABs are shown:
29. Bottom-left: **Back to Edit** (grey)
30. Bottom-right: **Print** (primary colour)
31. Existing desktop buttons are preserved for larger screens
32. UI language behavior
33. Detects the browser language at startup (`en` or `fr` , with `en` fallback)
34. Applies the active language to MUI date picker localization

35. Lets users change language from Edition Mode to update translated app labels and calendar labels
36. Provides a header GitHub shortcut to open a new issue on the repository

2.2 Functional Notes

- Date values are stored in `YYYY-MM` format.
- Shared links serialize form values and language in a base64url query parameter.
- Month generation is inclusive of both `fromDate` and `toDate`.
- The app currently trusts user input order and does not block `fromDate > toDate`.
- Free text fields are limited to 400 characters and show a live counter.

3. 02 - Edition Modes

The application has two modes controlled by internal UI state.

3.1 1) Edition Mode

Purpose: configure calendar content and layout.

Main UI elements:

- UI language selector (English/Français)
- Calendar title text field
- Free text field (optional, max 250 characters) with live counter
- Free text field below calendars (optional, max 250 characters) with live counter
- From month/year picker
- To month/year picker
- Orientation selector (portrait or landscape)
- Grid layout selector (`4x3` or `3x4`)
- Preview button

Behavior:

- On app load, the initial UI language is detected from the browser language.
- Users can change language in Edition Mode, and form/preview labels plus MUI picker labels update immediately.
- Form data is managed with React Hook Form.
- Required field validation is applied to title and date fields.
- Free text live counters turn orange from 220 characters and red at 250.
- Clicking Preview stores the current configuration and switches to Preview Mode.

3.2 2) Preview Mode

Purpose: validate the rendered calendar before printing.

Main UI elements:

- Back to Edit button
- Print button
- Calendar page preview (A4 dimensions)
- Footer with print timestamp

Behavior:

- The month list is recalculated from the selected date range.
- Grid columns/rows are derived from the selected layout.
- Clicking Back to Edit returns to Edition Mode while keeping existing values.
- Clicking Print updates timestamp and opens the browser print dialog.

3.3 Mode Transition Rules

- Default mode at app start: Edition Mode.
- Transition `edition -> preview` : user submits the settings form.
- Transition `preview -> edition` : user clicks Back to Edit.

4. 03 - Pages and Screen Captures

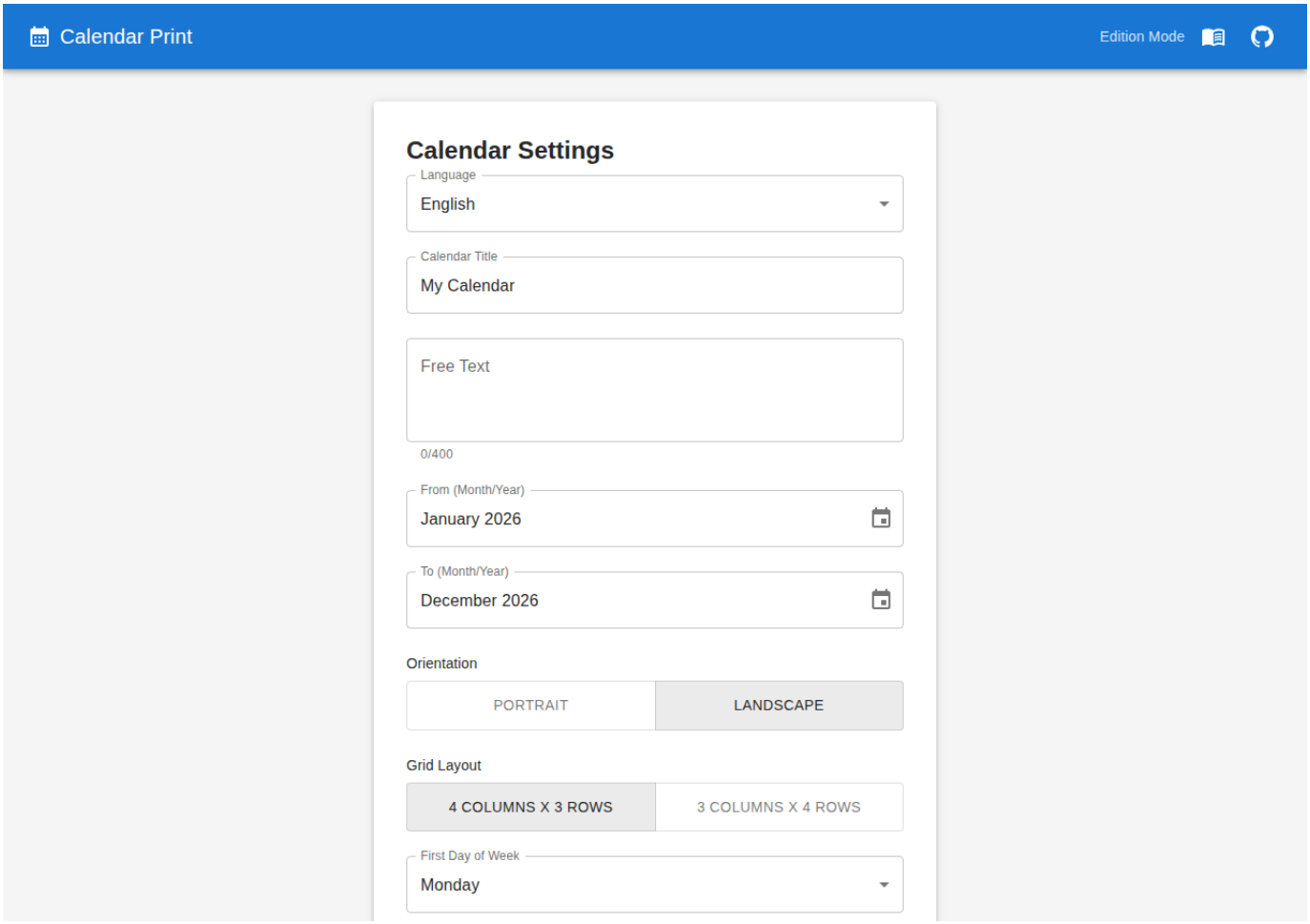
This app is a single-page React application with two functional views.

4.1 1) Edition Page

Description:

- Presents the calendar settings form.
- Used to define title, date range, orientation, and grid layout.

Screenshot (desktop, 1280 × 900):

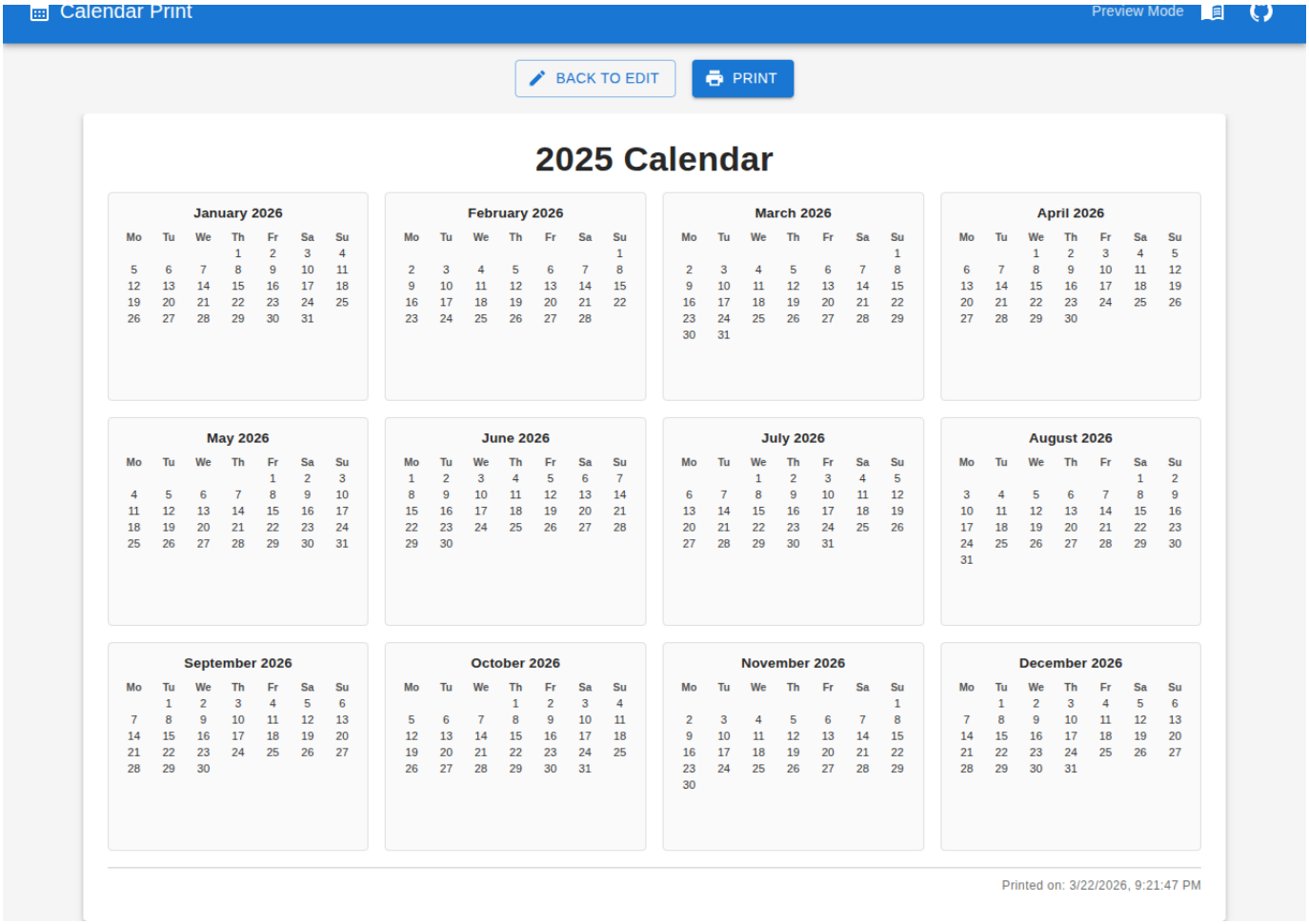


4.2 2) Preview Page

Description:

- Shows an A4 preview with all generated months.
- Includes actions to return to edition mode or print.

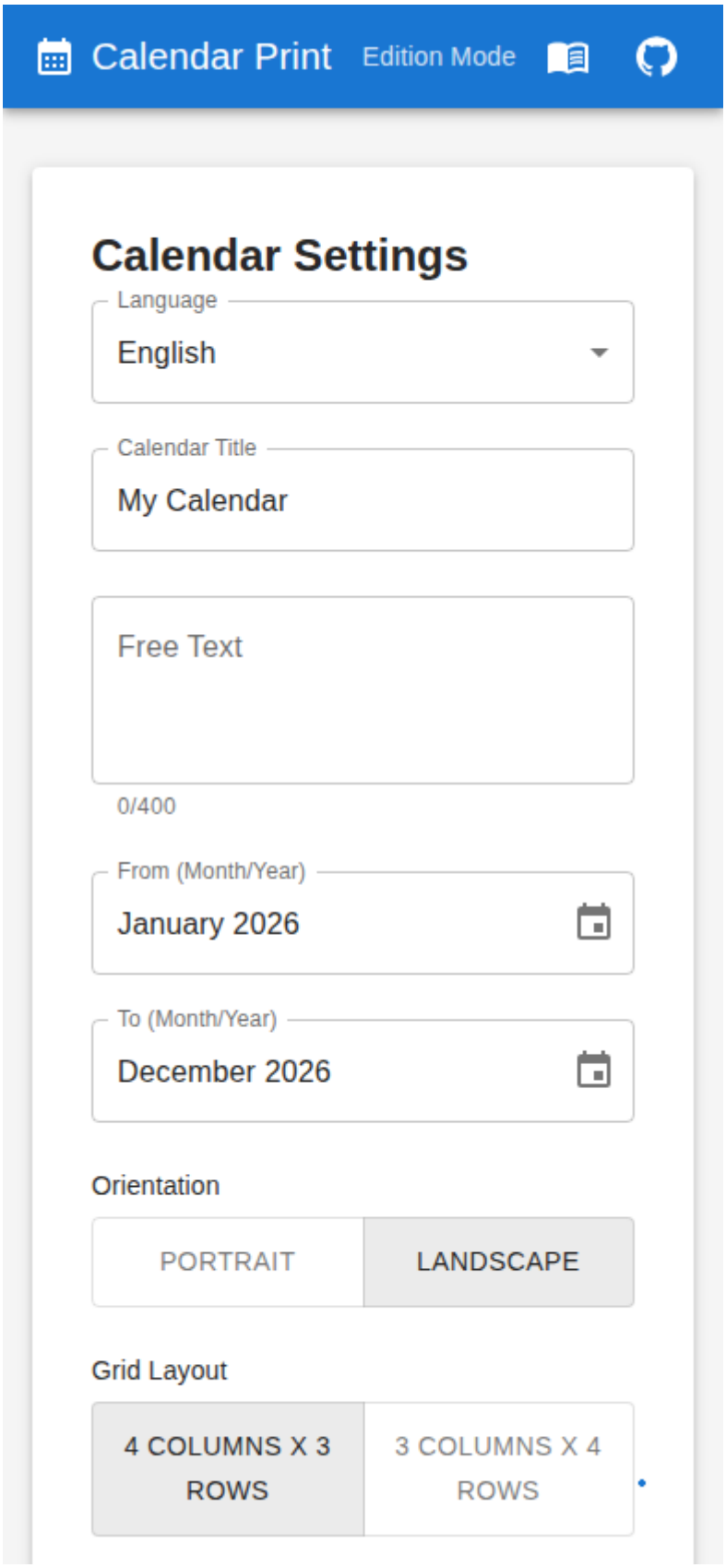
Screenshot (desktop, 1280 × 900):



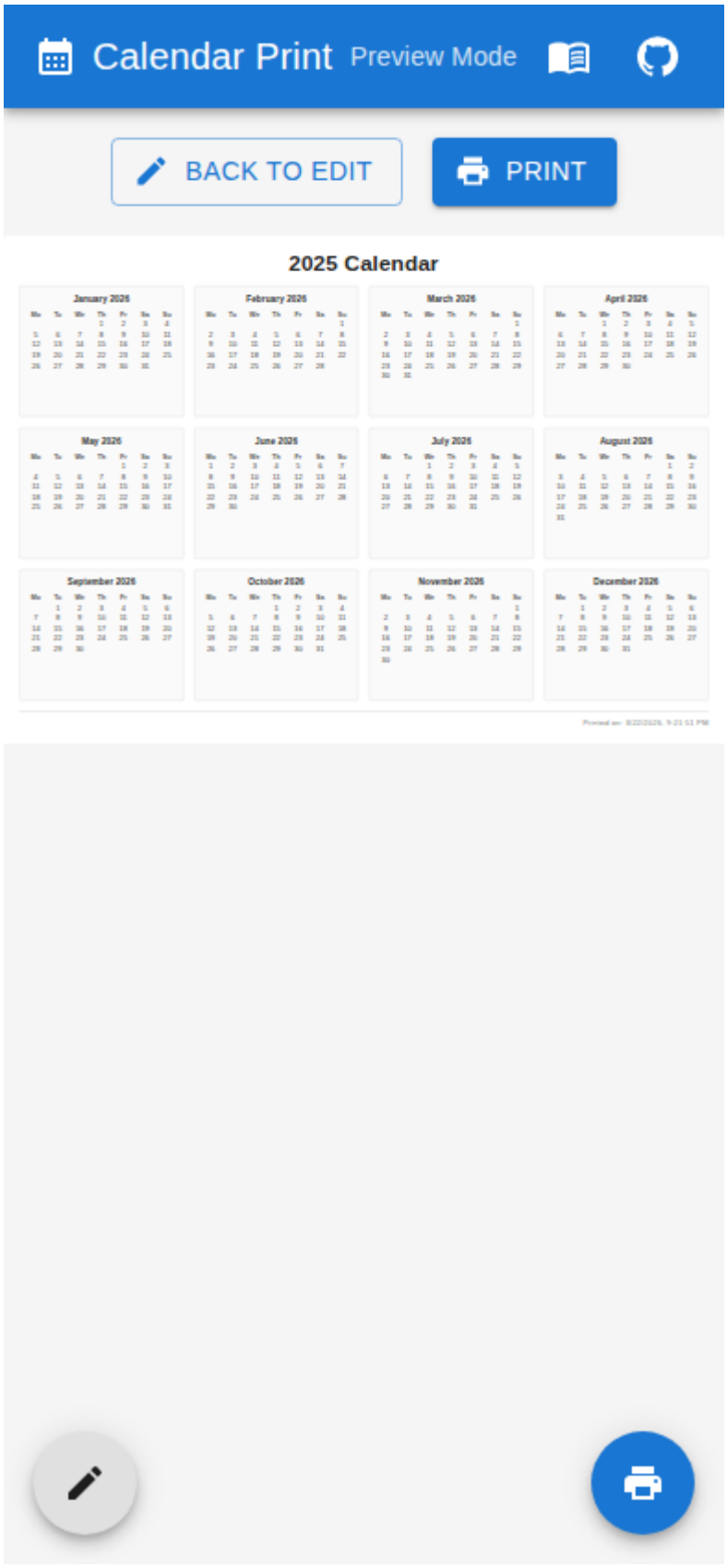
4.3 3) Mobile Views

On small screens the layout adapts: a SpeedDial FAB replaces desktop action buttons, and the A4 preview is automatically scaled to fit the viewport width.

4.3.1 Mobile Edition Page (390 × 844)



4.3.2 Mobile Preview Page (390 × 844)



4.4 4) Print Result (Optional)

Description:

- Captures the rendered print output or print preview dialog result.
- Useful to verify final layout and footer timestamp visibility.

Suggested capture file:

- `docs/03-pages-and-screen-captures/01-screenshots/print-result.png`

Example image reference (after adding the screenshot):

Print Result

4.5 How To Capture

1. Run the app in development mode.
2. Open Edition Page and capture the full form.
3. Switch to Preview Page and capture the full A4 preview area.
4. Resize to a mobile viewport (≤ 600 px wide) and repeat steps 2–3.
5. Open print preview and capture the printable result (if your browser allows it).

4.6 Screenshot Folder

For filename conventions and capture guidelines, see 01-screenshots/index.md.

5. 04 - Technical Dependencies

This document summarizes the runtime and development dependencies currently declared in `package.json`.

5.1 Runtime Dependencies

- `react` / `react-dom`: Core UI rendering and component model.
- `@mui/material`: Material UI components used for layout and controls.
- `@mui/icons-material`: Icon set used in action buttons and app bar.
- `@emotion/react` / `@emotion/styled`: Styling engine used by Material UI.
- `@mui/x-date-pickers`: Month/year picker components for date range selection.
- `dayjs`: Date parsing and formatting used for month generation and display.
- `i18next`: Core internationalization engine used for runtime language resources.
- `react-i18next`: React bindings for translation hooks and UI re-render on language switch.
- `react-hook-form`: Form state management and validation in Edition Mode.

5.2 Development Dependencies

- `vite` and `@vitejs/plugin-react`: Development server and production bundling.
- `typescript` and `@types/*` packages: Type-safe development and tooling support.
- `eslint`, `@eslint/js`, `typescript-eslint`, `eslint-plugin-react-hooks`, `eslint-plugin-react-refresh`, `globals`: Linting and code quality checks.
- `@types/node`: Node.js type declarations for tooling/config files.

5.3 Tooling Scripts

- `pnpm dev` or `npm run dev`: Start local development server.
- `pnpm build` or `npm run build`: Type-check and create production build.
- `pnpm lint` or `npm run lint`: Run linting rules.
- `pnpm preview` or `npm run preview`: Preview production build locally.

5.4 Integration Notes

- Date pickers are wrapped with `LocalizationProvider` and the Day.js adapter.
- App text translations are handled by `i18next` and `react-i18next`.
- Print behavior is controlled primarily by CSS `@media print` rules.
- The current `@page` print size rule is set to A4 landscape in CSS.

6. CI/CD Deployment Pipeline

6.1 Overview

This project uses GitHub Actions to automatically build, test, deploy to GitHub Pages, and create releases on every commit to the `main` branch.

6.2 Workflow Jobs

6.2.1 1. Build & Lint

Runs initial checks and builds the application:

1. Checks out code with full history
2. Sets up pnpm 9.x and Node.js 20 with dependency caching
3. Installs dependencies with `pnpm install --frozen-lockfile`
4. Runs markdown linting: `pnpm dlx markdownlint-cli2`
5. Runs code linting: `pnpm lint`
6. Builds site: `pnpm build`
7. Builds documentation with `ghcr.io/tiogars/mkdocs-docker-image` into `site_output/`, then copies it to `dist/docs/`
8. Automatically bumps PATCH version
9. Commits and pushes version changes to `main`

Outputs:

- `version`: Semantic version string (e.g., `0.1.2`)
- `version-tag`: Git tag (e.g., `v0.1.2`)

6.2.2 2. Deploy

Deploys built site to GitHub Pages:

1. Downloads build artifacts
2. Configures GitHub Pages settings
3. Uploads artifacts to Pages
4. Deploys using `actions/deploy-pages@v4`

Environment: GitHub Pages with OIDC token authentication

Deployment URL: Available as `steps.deployment.outputs.page_url`

6.2.3 3. Release

Creates GitHub Release with versioned archive:

1. Downloads build artifacts
2. Creates ZIP archive of built site
3. Creates GitHub Release with tag `vX.Y.Z`
4. Uploads ZIP file as release asset
5. Generates automated release notes

Release Contains:

- Version tag `vX.Y.Z`
- Automated changelog
- `calendar-X.Y.Z.zip` archive

6.3 Version Management

6.3.1 Semantic Versioning

The project uses semantic versioning: `MAJOR.MINOR.PATCH`

Strategy:

- Each commit to `main` automatically increments PATCH version
- Manual version bumps can be made by editing the `VERSION` file
- `package.json` is automatically kept in sync

Version File: `./VERSION` (root of repository)

6.3.2 How Versioning Works

1. Workflow reads current version from `VERSION` file
2. Increments PATCH component (e.g., `0.0.0` → `0.0.1`)
3. Updates both `VERSION` and `package.json`
4. Commits changes with automated message: `chore: bump version to X.Y.Z`
5. Pushes commits to `main` (will trigger another workflow run)

6.3.3 Manual Version Control

To bump MAJOR or MINOR version:

1. Edit `VERSION` file directly:

```
0.1.0  # was 0.0.20
```

1. Update `package.json`:

```
{
  "version": "0.1.0"
}
```

1. Commit: `git commit -am "chore: bump to 0.1.0"`
2. Push: `git push origin main`

Next automated build will increment PATCH: `0.1.1`

6.4 GitHub Pages Configuration

6.4.1 Prerequisites

1. **Enable GitHub Pages**:
2. Repository Settings → Pages
3. Source: Deploy from a branch
4. Branch: `gh-pages`

5. Folder: `/ (root)`
6. **Set Base Path** (if needed):

For project site (e.g., `https://username.github.io/calendar/`):

Update `vite.config.ts`:

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  base: '/calendar/',
  plugins: [react()],
})
```

For user/org site (e.g., `https://username.github.io/`):

```
export default defineConfig({
  base: '/',
  plugins: [react()],
})
```

6.4.2 Deployment

- Automatically deploys on every push to `main`
- Application deployed to the GitHub Pages root URL
- Documentation deployed to the `/docs/` subfolder of the same Pages site
- Deployment URL shown in Actions workflow logs

6.4.3 Documentation URL

The MkDocs documentation is accessible at `<pages-url>/docs/` (e.g., `https://tiogars.github.io/calendar/docs/`).

It is built with the `ghcr.io/tiogars/mkdocs-docker-image` Docker image using the `mkdocs.yml` configuration at the root of the repository.

6.5 GitHub Runner & Environment

- **OS**: Ubuntu latest (`ubuntu-latest`)
- **Node.js**: Version 20
- **pnpm**: Version 9.x
- **Cache**: Enabled for pnpm dependencies
- **Concurrency**: Maximum 1 deployment at a time

6.6 Permissions & Security

The workflow requires GitHub Actions permissions:

- `contents: write` - Commit version bumps and create releases
- `pages: write` - Deploy to GitHub Pages
- `id-token: write` - Secure OIDC token for Pages deployment

All automated commits use `github-actions[bot]` account.

6.7 Manual Workflow Trigger

Manually run the workflow from GitHub UI:

1. Actions tab → "Build, Deploy & Release"
2. Click "Run workflow"
3. Select branch: `main`
4. Click "Run workflow"

Useful for testing or deploying without code changes.

6.8 Artifacts & Outputs

6.8.1 Build Artifacts

- **dist/**: Primary build directory (Vite output)
- Uploaded as `dist` artifact with 1-day retention
- Downloaded by deploy and release jobs

6.8.2 GitHub Pages

- Deployed to configured GitHub Pages URL
- Contains complete built site
- Accessible immediately after deployment

6.8.3 GitHub Releases

- Created with semantic version tag (`vX.Y.Z`)
- Release notes auto-generated from commit history
- Includes `calendar-X.Y.Z.zip` archive
- Downloadable from GitHub Releases page

6.9 Troubleshooting

6.9.1 Workflow Fails on Lint

Markdown linting errors:

```
pnpm dlx markdownlint-cli2 "README.md" "docs/**/*.md" \
  ".github/**/*.md"
```

Fix issues locally, then commit.

ESLint errors:

```
pnpm lint --fix
```

6.9.2 Pages Not Updating

1. Check repository Settings → Pages is enabled
2. Verify `gh-pages` branch exists
3. Check workflow logs in Actions tab
4. Manually trigger workflow to retry

6.9.3 Version Commits Not Pushing

Verify GitHub Actions repository permissions:

- Settings → Actions → General
- Workflow permissions: "Read and write permissions"
- Allow GitHub Actions to create and approve pull requests

6.9.4 Release Creation Failed

Ensure repository allows:

- GitHub Actions to write releases
- No conflicting branch protections
- Sufficient storage quota for ZIP archives

6.10 Best Practices

1. **Always test locally** before pushing:

```
pnpm lint
pnpm build
```

1. **Keep VERSION file in sync** with package.json
2. **Review automated commits** in git history (version bumps)
3. **Monitor workflow runs** in Actions tab after each push
4. **Use release archives** for easy rollback and distribution