Technical Documentation

Planvas

Last Updated: December 2024

Contents

1 Introduction

1.1 Purpose

Planvas is a web application that students can use as a calendar for all course work and events. It provides functionality to combine separate calendars, for example from Google and Canvas, into one central calendar.

1.2 Scope

Planvas supports basic functionality for adding assignments with their subjects, details, and due dates. Users can filter and sort their assignments by different criteria, such as by due date. It offers .ics import functionality to make the transfer of data from calendars like Canvas' calendar and Google Calendar simple for users.

1.3 Audience

Planvas is an excellent tool for students who may have deadlines spread out across multiple calendars. Users who value centralizing and organizing their responsibilities will enjoy Planvas' features to trivialize schoolwork management.

2 System Overview

2.1 Architecture

Planvas uses the Onion Architecture, with domain layer containing all the entities, the application layer with all the services of the web-app, the infrastructure layer with access to the database, and the presentation layer with the user interfaces and user interaction. Planvas also uses the Data Access Object (DAO) pattern, separating the client interface from the interactions with the database.

2.2 Technologies Used

Wrapper:

- Gradle 7.1.1

Framework:

- Angular
- Jersey/Guice

Runtime Environments:

- Node.js 14.16.1
- JDK 11

Languages:

- Java 11
- TypeScript
- HTML
- CSS

Database:

- HSQL

Web Server:

- Tomcat

2.3 Dependencies

Server dependencies can be found in

```
~planvas/assignment/server/build.gradle
```

For reference, this is a list of the included dependencies:

- Apache Commons Lang
- Google Guice
- Guice Bridge
- HSQLDB
- Jackson
- Jersey
- Joda-Time
- Logback
- SLF43
- Typesafe Config
- UrlRewriteFilter
- Servlet
- Mockito
- TestNG

3 Installation Guide

3.1 System Requirements

Windows, macOS, or Linux operating systems.

3.2 Installation Steps

Access the GitHub Repository here.

Method 1:

Open a new terminal session and run

```
git clone https://github.com/hoi-is-repeating/CS-370-Planvas
```

into the directory where you want Planvas to be installed.

Method 2:

1. Click "Code" on the repository page
2. Click "Download ZIP"
3. Extract the ZIP file into the directory where you want Planvas to be installed

4 Configuration Guide

4.1 Environment Setup

1. Download JDK 11
2. Download Node.js

From the terminal, navigate to:

```
~/planvas/assignment/client
```

and run:

1. nvm install v14.16.1
2. nvm use v14.16.1
3. npm install
4. npm install -g @angular/cli@13.3.4
5. npm install -g gulp
6. npm install ngx-bootstrap bootstrap

7. ng build
8. ng serve

The web application client should now be live, listening on port 4200 at `http://localhost:4200/`.

> IMPORTANT: The application's functions will not work yet without starting the server. Please see Deployment for more details on how to deploy the application.

5 Project Overview

5.1 Web Application

Planvas uses the Angular framework.

```
~/planvas/assignment/client/webapp/
```

This directory houses the components for the web application client.

The major functionalities are split into six sub-groups: domain, global, login, main-grid, service, and signup.

Other pertinent files within the app folder are `app.component` files (with an `app.module` and `app-routing.module`).

Domain holds two files: `assignment.d.ts` (an interface for assignments) and `config.d.ts` (defines the type format of the config object).

Global holds one file: constants.ts, which contains pertinent data on page sizes and configuration.

Login contains the necessary login component files to process user login requests/actions.

5.1.1 LoginComponent

The LoginComponent manages the login page functionality in an Angular application. It leverages the Router service to navigate to the calendar page (`/main-grid`) when the "Login" button is clicked.

- Selector: `<planvas-login>`
- Template and Styles:
  - `login.component.html`

  - `login.component.css`.

- Key Methods: `onLogin` - navigates to `/main-grid` upon user login action
- Lifecycle Hook: `ngOnInit` - placeholder for initialization logic, currently unused
- Routing: requires proper configuration in the app's routing module

This component provides a modular and reusable structure for login functionality with dynamic navigation to the calendar page.

The main-grid folder represents the overarching structure of the home-page, with sub components of calendar, modal, and sidebar. It holds the main-grid components that define the structure of the homepage.

The subfolder calendar within main grid houses the calendar component. The CalendarComponent displays and manages a dynamic calendar in an Angular application, integrating with an assignment service to populate calendar days with relevant assignments. It includes navigation between months and supports pagination for assignment data.

- Selector: `<planvas-calendar>`
- Template and Styles:
  - `calendar.component.html`
  - `calendar.component.css`
- Key Properties:
  - `currentDate`: Tracks the currently displayed month.
  - `calendarDays`: Holds the days to be displayed, including assignment data.
  - `assignments`: Stores loaded assignments for the current page.
  - `page/pageSize`: Tracks pagination settings for assignment data.
  - `calendarTitle`: Displays the current month and year.
- Key Methods:
  - `populateCalendar`: Generates the calendar grid for a specific month, filling in assignments and handling days from adjacent months.
  - `loadAssignments`: Fetches paginated assignment data and the total count using observables.
  - `setAssignments`: Updates the assignment list and adjusts pagination details.
  - `setPage`: Changes the current page and reloads assignments.

  - `prevMonth/nextMonth`: Navigate to the previous or next month and refresh the calendar view.

- Dependencies:
  - Services: AssignmentsService for fetching assignment data.
  - Libraries: Font Awesome icons, RxJS for managing observables, and moment for date formatting.
  - Modals: Integrates modals for additional assignment-related actions like adding or removing assignments.

- Lifecycle: `ngOnInit` - Initializes the calendar for the current month and loads the first page of assignments.

This component is designed to provide a responsive and feature-rich calendar, seamlessly integrating assignment management with calendar navigation.

The subfolder modal within main-grid contains components for adding, removing, and completing assignments, loading upcoming assignments, and loading contacts.

5.1.2 AddNewAssignmentComponent

This Angular component is used within a modal to create new assignments. It incorporates a reactive form, validation logic, and datepicker configuration, providing a user-friendly interface for entering assignment details.

Key features:

1. Reactive form for assignments:
   - Includes fields: date, time, subject, and details
   - Built with Angular's FormBuilder for structured and dynamic control
2. Custom validation:
   - Subject validators:
     - Maximum length validation
     - Prevents double spaces
   - Details validator:
     - Ensures text length does not exceed a specified maximum
   - Error messages are displayed in a modal `errorModal`
3. Datepicker integration:
   - Configures `BsDatepickerConfig` for a user-friendly date selection

   - Uses a customizable date format (`DD-MMM-YYYY`)

4. Time options:
   - Generates selectable time intervals (e.g. every 30 minutes)
   - Configurable based on app settings (`meetingIntervalMinutes`)

5. Modal management:
   - Save: Emits a new assignment as an observable (`submit$`) with:
     - Combined date and time as a timestamp
     - Validated subject and details
   - Cancel: Closes the modal without emitting data

6. Utility methods:
   - Error handling: displays error modals when validation fails
   - Validation functions: encapsulate logic for reusable checks
   - Time list generator: dynamically creates time options for dropdowns

7. Icons: uses Font Awesome (faWarning) for styling and UX.

8. Dependency injection:
   - Injected services:
     - ConfigService for app-specific configurations
     - BsModalRef for managing modal instances

5.1.3 CompleteAssignmentsComponent

This Angular component is designed for confirming the deletion or completion of assignments via a modal dialog. It uses reactive programming principles to handle user actions and interacts with the parent component through observables.

Key features:

1. Modal purpose:
   - Used to confirm actions on a list of assignments (e.g., marking as complete or deleting)

2. Input data:
   - Subject validators:
     - Accepts a list of `Assignment` objects from the parent component via the `@Input()` decorator

3. Observables for communication:
    - Utilizes an RxJS Subject (`submit$`) to emit a boolean value (true for confirmation)
4. Date-time format:
    - Configures and applies a consistent date-time format using a value from ConfigService
5. Modal management:
    - Confirm: emits true to indicate the action was confirmed, closes the modal, and completes the observable
    - Cancel: closes the modal without emitting any value and completes the observable
6. Dependency injection:
    - Injected services:
        - ConfigService for app-specific configurations (e.g., date-time format)
        - BsModalRef for managing the modal instance
7. Minimalist UI logic: no additional logic or validation, focusing solely on confirming or canceling the user action

5.1.4 LoadContactsComponent

This Angular component is designed for loading and handling contact information in a modal. It does not require a selector since it is referenced by its class name when opened within a modal.

Key features:

1. Input data:
    - Accepts a list of contact names (strings) via the `@Input()` decorator, passed from the parent component
2. Observables for communication:
    - Utilizes an RxJS Subject (`submit$`) to pass a boolean value indicating the user's action (e.g. confirming or canceling the operation)
3. Modal management:
    - Cancel: closes the modal without emitting any value and completes the observable to unsubscribe from it
4. Dependency injection:

- BsModalRef for managing the modal instance (open/close)

5. Minimalist UI logic: focuses on modal interaction, with no additional functionality beyond showing contacts and handling the modal lifecycle

5.1.5 RemoveAssignmentsComponent

This Angular component is used within a modal for confirming the deletion of assignments. It does not require a selector since it is referenced by its class name when the modal is shown.

Key features:

1. Input data:
    - Accepts a list of assignments to be removed, passed from the parent component via the `@Input()` decorator
2. Observables for communication:
    - Utilizes an RxJS Subject (`submit$`) to emit a boolean value indicating the user's action (e.g. confirming or canceling the deletion)
3. Font Awesome Integration: displays a warning icon (faWarning) to alert the user about the deletion action
4. Modal management:
    - Confirm: emits true and closes the modal, signaling that the deletion action is confirmed
    - Cancel: completes the observable and closes the modal without emitting any value, indicating cancellation of the action
5. Dependency injection:
    - ConfigService to fetch configuration values (such as date format)
    - BsModalRef to control the modal instance
6. Minimalist UI logic: handles the modal's confirm and cancel actions, with minimal additional functionality

The sub-folder `main-grid/sidebar/` handles important functionality for the to-do on the left-hand side of the web application.

5.1.6 SidebarComponent

This SidebarComponent in Angular manages the UI and functionality of an assignment sidebar. It includes features for displaying, managing, and interacting with assignments, as well as importing assignments from .ics files.

Core functionalities:

1. Assignment management
    - Load assignments: fetches assignments and the total count for pagination
        - Method: `loadAssignments(page, pageSize)`
        - Uses RxJS `combineLatestWith` for simultaneous data fetching
    - Set assignments: updates the view with fetched assignments
        - Method: `setAssignments(assignments, total)`
2. Pagination
    - Manages current page, total pages, and page size
    - Methods: `setPage(newPage), setLastPage(), onFirstPage(), onLastPage()`
    - Options: Configured using constants like `pageSizeOptions`
3. CRUD Operations
    - Add assignments: opens a modal for creating assignments
        - Method: `addNewAssignment()`
    - Edit assignments: updates an existing assignment
        - Method: `saveEditedAssignment(editedAssignment)`
    - Delete assignments: removes selected assignments after confirmation
        - Method: `removeAssignments()`
4. Sidebar Features
    - Minimize sidebar: toggle state for a compact UI
        - Property: `isMinimized`
        - Method: `toggleMinimize()`

Advanced features:

1. File import
    - Supports .ics files for importing calendar data as assignments
        - Open dialog: `openFileDialog()`
        - Read file: `readICSFile(file)`
        - Parse file: `parseICS(icsContent)`
2. Upcoming and contacts modal methods
    - `loadUpcomingAssignments()`: displays assignments for the next 24 hours

- `loadContacts()`: retrieves unique names from all assignments

3. Icons and styling:
    - Font Awesome icons: used for navigation (e.g., faAngleLeft, faAngleRight)
    - Styling linked via `sidebar.component.css`
4. RxJS and observables:
    - Used extensively for data fetching (`assignmentsService`) and modal interactions.
    - Operators: `combineLatestWith, pipe, take`
5. Bootstrap modals:
    - Integration with ngx-bootstrap for modal dialogs:
        - Components: AddNewAssignmentComponent, LoadContactsComponent, RemoveAssignmentsComponent
6. Dependency injection
    - Services injected in the constructor:
        - AssignmentsService: manages backend assignment data
        - ConfigService: fetches global configurations (e.g. dateFormat)

The service folder has a service that handles API requests made to the server, and contains another config file that does the following:

The ConfigService is an Angular service that retrieves configuration values stored in a global object planvas. The configuration values are initialized via a JSP file (`config.jsp`) and made accessible globally through window or globalThis.
Key functionalities:

1. Dependency injection: registered with Angular's `@Injectable` system, ensuring it can be used as a singleton service across the application
2. Type-safe configuration retrieval: The `get<T>` method allows fetching configuration values by specifying a dot-notated path. Type inference or explicit typecasting ensures flexibility and safety.

The signup folder houses necessary components for user account creation (sign ups).

5.1.7 SignupComponent

The SignupComponent handles the signup page function in an Angular application. It uses the Router service to navigate to the calendar page `/main-grid/` when the "Sign Up" button is clicked.

- Selector: `<planvas-signup>`
- Template and styles:
  - `signup.component.html`
  - `signup.component.css`
- Key Method: `onSignup` - navigates to `/main-grid/` on user interaction
- Routing: requires proper configuration in the app's routing module
- Enhancements: add input validation, error handling, and unit tests for navigation and component creation

This component is modular, reusable, and provides dynamic navigation as part of the signup process.

5.2 Database Schema

The schema is based primarily around the `Assignments` table, with the following attributes:

1. `id` **(KEY)**: An integer, auto-incremented (using IDENTITY keyword) to uniquely identify each assignment.
2. `date`: A timestamp attribute that stores the date the user selects for the assignment.
3. `subject`: A variable character attribute that holds the subject of the assignment, with a max length of 30.
4. `details`: A variable character attribute that holds additional details or a description of the assignment, with a max length of 100.
5. `active`: A character attribute that indicates if the assignment is active or not. The default value is set to Y, meaning active.

6 Testing

6.1 General Advice

If you run into issues with multiple test cases, try isolating them and running them one-by-one. Sometimes, errors in previous test cases end up interfering with later test cases.

Test the server with the provided test cases or with your own by running:

```
./gradlew test
```

from the root project directory.

6.2 Test Cases

This is a list of the test cases used during development.

- testGetAssignments()
- testGetUpcomingAssignments()
- testGetContacts()
- testgetTotalAssignments()
- testDeleteAssignments()
- testDeletePageAssignments()
- testAddingAssignmentWithAllValidFields()
- testAddingAssignmentNullSubject()
- testAddingAssignmentEmptySubject()
- testAddingAssignmentRestrictedCharacterSubject()
- testAddingAssignmentSubjectMaxLength()
- testAddingAssignmentAllWhitespace()
- testAddAssignmentMissingDate()
- testAddAssignmentMissingDetails()
- testAddingAssignmentDetailsMaxLength()
- testAddAssignmentInThePast()

7 Deployment

7.1 Deployment Process

1. Download Tomcat
2. From the terminal, navigate to the project root directory `planvas`
3. Build the project with

   ```
   ./gradlew build -x npmLint
   ```

4. Locate the generated WAR file in the project at the `build/libs/` directory
5. Copy the WAR file into the Tomcat `webapps` directory
6. Navigate to the Tomcat `bin` directory
7. Run `startup.sh`
   - If necessary, give permission to `startup.sh` by running

     ```
     chmod +x startup.sh
     ```

   - On Windows or non-Unix systems, run `startup.bat`

   The application should now be live, listening on port 8080 at `http://localhost:8080/assignment/`.

7.2 Known Issues and Limitations

- Does not support “periods” of time
- Calendar needs a refresh for assignments to populate it

8 Troubleshooting Guide

Building Client

- Ensure you have all necessary libraries installed, as outlined in the Configuration Guide.

Building Project

- If `./gradlew build -x npmLint` does not work, try running

  ```
  ./gradlew clean build -x npmLint
  ```

  Sometimes, the application has issues that persist from previous builds. This will create a clean build of the project.

9 Version History

RELEASE 1.0

- released Planvas

0.5.0

- removed Spring Boot
- added HSQL support
- improved UI/UX

0.4.2

- added Spring Boot “user” and “assignment” controller files
- fixes to Spring Boot dependencies

0.4.1

- changed user function on the client

0.4.0

- added draft .ics import function
- added draft syllabus parsing function

0.3.3

- changed “task” to “assignment” in backend logic

0.3.2

- fixes to “user” and “task” to work with framework changes

0.3.1

- added Spring Boot dependencies
- added Spring Boot REST structure to server files
- restructured Spring Boot to work with Gradle

0.3.0

- removed Spring Boot framework
- added Angular framework
- updated UI/UX for the client

0.2.2

- added “task” feature support

0.2.1

- added “user” feature support
- added user creation services

0.2.0

- added Spring Boot framework with Maven
- added PostgreSQL support

0.1.3

- added draft Java files for log in, sign up, and task functionality
- added draft log in HTML and CSS files for testing

0.1.2

- adjustments to SQL script

0.1.1

- added draft SQL script for database schema initialization
- added draft HTML for calendar, sidebar, and buttons

0.1.0

- added README