@in.work/in.work

Open Source Corporate Agile Tool

- readme.md for @in.work/in.work
- changelog.md for @in.work/in.work

readme.md for @in.work/in.work

A modern Kanban project management system with bidirectional external sync capabilities. Built as a Jira alternative focused on simplicity, real-time collaboration, and seamless integration with external systems like Gitea, GitLab, and email.

Features

Core Kanban

- **Boards & Columns** - Flexible Kanban boards with customizable columns and WIP limits
- **Cards** - Rich work items with descriptions, priorities, labels, due dates, and time tracking
- **Drag & Drop** - Intuitive card movement between columns
- **Card Links** - Link cards with relationships (blocks, relates to, duplicates, parent/child)

Collaboration

- **Real-time Updates** - WebSocket-powered live updates across all connected clients
- **Presence Indicators** - See who's viewing the same board
- **Typing Indicators** - Know when teammates are commenting
- **Card History** - Full timeline of all changes and movements

External Sync

- **Gitea Integration** - Bidirectional sync with Gitea issues
- **Mirror to External** - Create cards locally, mirror them to external repos
- **Webhook Support** - Real-time updates from external systems
- **Privacy Filter** - AI-powered filtering of sensitive data during sync

Access Control

- **Organizations** - Multi-tenant with organization-level isolation
- **Projects** - Group related boards within organizations
- **Role-Based Access** - Hierarchical permissions (owner, admin, manager, contributor, viewer)
- **Board Sharing** - Share boards across organizations or via public links
- **OAuth/OIDC** - External authentication providers (Gitea, GitLab, etc.)

API & Automation

- **REST API** - Comprehensive API for all operations
- **API Tokens** - Scoped tokens for automation (`inw_` prefix)
- **Audit Logging** - Full audit trail for compliance

Tech Stack

- **Backend**: Deno + TypeScript
- **Database**: MongoDB via @push.rocks/smartdata
- **Frontend**: Angular 19 with signals
- **Real-time**: Native WebSocket
- **AI**: @push.rocks/smartai (configurable per instance)

Quick Start

Prerequisites

- Deno 2.x
- MongoDB 6.x+
- Node.js 20+ (for UI development)

Development Setup

1. Clone the repository:

```
git clone https://github.com/your-org/in.work.git
cd in.work
```

2. Create environment config:

```
mkdir -p .nogit
cat > .nogit/env.json << 'EOF'
{
  "MONGO_URL": "mongodb://localhost:27017",
  "MONGO_DB": "inwork",
  "JWT_SECRET": "your-secret-key-change-in-production"
}
EOF
```

3. Start the development server:

```
deno task dev
```

4. Access the application:

- Web UI: http://localhost:3000
- API Health: http://localhost:3000/api/v1/health
- WebSocket: ws://localhost:3000/ws/realtime

Default Admin

On first run, a platform admin is created:

- Email: `admin@in.work`
- Password: `admin123`

Change this immediately in production!

API Overview

Authentication

```
# Login
curl -X POST http://localhost:3000/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"admin@in.work","password":"admin123"}'

# Use token in subsequent requests
```

```
curl http://localhost:3000/api/v1/organizations \
  -H "Authorization: Bearer <access_token>"
```

Key Endpoints

Method	Endpoint	Description
POST	`/api/v1/auth/login`	Login
POST	`/api/v1/auth/register`	Register
GET	`/api/v1/organizations`	List organizations
POST	`/api/v1/organizations`	Create organization
GET	`/api/v1/projects`	List projects
GET	`/api/v1/boards/:id`	Get board
GET	`/api/v1/boards/:id/cards`	Get board cards
POST	`/api/v1/boards/:id/cards`	Create card
POST	`/api/v1/cards/:id/move`	Move card
POST	`/api/v1/connections`	Create external connection
POST	`/api/v1/connections/:id/sync`	Trigger sync

Real-time WebSocket

Connect to receive live updates:

```
const ws = new WebSocket('ws://localhost:3000/ws/realtime?token=' + accessToken);

ws.onmessage = (event) => {
  const data = JSON.parse(event.data);
  console.log('Event:', data.type, data.data);
};

// Subscribe to a board
ws.send(JSON.stringify({
  action: 'subscribe',
  roomId: 'Board:xxx',
  roomType: 'board'
```

```
}));
```

Event Types

- `card:created`, `card:updated`, `card:moved`, `card:deleted`
- `column:created`, `column:updated`, `column:deleted`
- `presence:join`, `presence:leave`, `presence:update`
- `typing:start`, `typing:stop`

Project Structure

```
in.work/
├── ts/
│   ├── index.ts              # Entry point
│   ├── cli.ts                # CLI commands
│   ├── inwork.ts             # Main application class
│   ├── plugins.ts            # Dependency imports
│   ├── interfaces/           # TypeScript interfaces
│   ├── models/               # SmartData entities
│   ├── services/             # Business logic
│   ├── api/                  # REST API router
│   └── sync/                 # External sync engine
├── ui/                       # Angular 19 frontend
├── test/                     # Test files
├── .nogit/                   # Local config (gitignored)
├── deno.json                 # Deno configuration
└── readme.md
```

Configuration

Environment Variables

Variable	Description	Default
`MONGO_URL`	MongoDB connection URL	`mongodb://localhost:27017`

Variable	Description	Default
`MONGO_DB`	Database name	`inwork`
`JWT_SECRET`	Secret for JWT signing	Auto-generated (dev)
`PORT`	Server port	`3000`
`SMTP_HOST`	SMTP server for emails	-
`SMTP_USER`	SMTP username	-
`SMTP_PASS`	SMTP password	-

External Sync

Gitea Setup

1. Create OAuth application in Gitea
2. Add connection via API:

```
curl -X POST http://localhost:3000/api/v1/connections \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "My Gitea",
    "systemType": "gitea",
    "config": {
      "baseUrl": "https://gitea.example.com",
      "token": "<gitea-token>"
    },
    "syncSettings": {
      "syncDirection": "bidirectional",
      "autoSync": true,
      "syncIntervalMinutes": 15
    }
  }'
```

3. Link a board to a repository and sync will begin automatically

License

MIT

changelog.md for @in.work/in.work

2025-12-07 - 1.2.0 - feat(comments)

Add comments subsystem: backend models & routes, realtime events, UI components, and tooling updates

- Backend: new Comment model with CRUD helpers, soft-delete/restore, mention and card-link parsing, and card comment count updates
- API: added comment routes (list/create/update/delete) and autocomplete endpoints for member/card suggestions; router registers comment routes
- Realtime: added comment events (comment:created, comment:updated, comment:deleted) and broadcast helpers in RealtimeService
- UI: added comment components (CommentList, CommentItem, CommentInput) and integrated comments into CardDetail; ApiService client methods for comments and suggestions
- UI library integration: migrated many form controls and buttons to @design.estate/dees-catalog components and added CUSTOM_ELEMENTS_SCHEMA where required
- Styling & build: added Tailwind (tailwind.config.js), PostCSS config, updated main.scss to include Tailwind base/components/utilities and dees-catalog theme overrides
- Dependencies & tooling: bumped @design.estate/dees-catalog, added clsx, tailwind-merge, tailwindcss, postcss, autoprefixer and updated pnpm lockfile; added "up" script to root package.json for updating UI deps
- Models index and exports updated to include Comment; interfaces extended with IComment and related types

2025-12-06 - 1.1.0 - feat(api)

Add full REST API, core models, auth, realtime and sync services

- Add API router and modular route handlers for authentication, organizations, projects, boards, columns, cards, connections and tokens
- Implement AuthService with JWT access/refresh tokens, session management, registration and OAuth flows
- Add core models: User, Session, ApiToken, Board, Card, Organization, ExternalConnection and related model helpers
- Introduce RealtimeService (WebSocket) for presence, typing indicators and broadcast events
- Add PermissionService for RBAC resolution and NotificationService (email templates / placeholders)
- Implement connection/webhook endpoints and ExternalConnection model to support external sync providers
- Seed default admin in DB initialization and provide DB connection singleton (db.ts)
- Expose shared TypeScript interfaces under ts/shared and update UI API client mapping for getOrganization

2025-12-06 - 1.0.0 - BREAKING CHANGE(auth)

Auto-login on registration and align frontend API responses

- Backend: after successful user registration the server now attempts to auto-login the user and returns accessToken, refreshToken and sessionId along with an enriched user object (avatarUrl, isPlatformAdmin). The registration flow captures IP and User-Agent for session creation. This changes the auth response shape (no more success/expiresIn).
- Frontend (UI): auth.service.ts updated to expect accessToken/refreshToken/sessionId, removed reliance on response.success/expiresIn and now schedules token refresh with a fixed 15-minute interval. Login/register flows updated to set auth from the new token response.
- Frontend (API client): organizations endpoints updated to map the API's organization payload shape (displayName -> name, plan typing), and createOrganization now posts displayName+slug fields and maps the returned organization object to the frontend model.
- Config: Angular CLI analytics disabled in ui/angular.json.
- Tests / tooling: Playwright screenshots (test artifacts) were added under .playwright-mcp.
- Compatibility note: The auth and some API response shapes changed — clients relying on the old fields (e.g. response.success or expiresIn, or older organization JSON shapes) will need updates.

2025-12-06 - 0.2.0 - feat(ui)

Add full Angular frontend, embed built UI into backend, and implement core realtime/sync/services

- Add a complete Angular 19 frontend (ui/) with routes and many standalone components: login, register, dashboard, organization, project, board, card detail, settings, sidebar, toasts and shared layout/styles.
- Introduce client-side services and utilities: ApiService, AuthService, RealtimeService, ToastService, and HTTP auth interceptor for the UI.
- Add build/dev orchestration: top-level package.json (concurrently), ui/package.json, angular.json, tsconfig, proxy.conf.json and deno.json script updates for build, watch and bundle tasks.
- Add scripts/bundle-ui.ts to embed compiled Angular UI files into a generated TypeScript module (ts/embedded-ui.generated.ts) and a watch mode for live rebundling.
- Serve embedded UI from the backend: ts/inwork.ts now uses getEmbeddedFile/hasEmbeddedFiles, serves embedded assets with SPA fallback, and retains a dev console fallback page (with WebSocket hot-reload UI).
- Large backend additions and enhancements: expanded ApiRouter routes, AuthService, RealtimeService, PermissionService, NotificationService, SyncEngine and Gitea provider, and many SmartData models (users, sessions, boards, columns, cards, memberships, card links, history, tokens, auth providers, oauth state, board shares, etc.).
- Database init now seeds a default admin user when needed (db init), and models include rich public info/CRUD helpers and lifecycle hooks.
- UI styling and theme: add ui/src/styles/main.scss with design system, utilities and animations.

2025-12-06 - unknown - initial

Initial commit: project scaffold and first files.

- Repository initialized.
- Initial project structure and placeholder files added.