What is new in Editoria *Monemvasia*?

Infrastructure and component level (more details on below)

- 100% rebuilt application!
- Upgraded Wax for web based word processing!
- Upgraded XSweet for conversion!
- Extended Paged.j for export!!

Community-led features

WAX - Color code tracked changes by role *(UCP)*
EDITORIA - Configure book builder to omit blue buttons *(Book Sprints)*
EDITORIA - Indenting chapters in book builder *(Book Sprints)*
EDITORIA - Error message when uploading incorrect file format *(Longleaf Services)*
WAX - Author name style *(punctum)*
WAX - Provide a larger text box for inputting figure captions *(Longleaf Services)*
EDITORIA - enable "read only" chapter mode *(Book Sprints)*
EDITORIA - Autocomplete for adding book team roles *(punctum)*
EDITORIA - Always show chapter name *(Longleaf Services)*
EDITORIA - Book-level (and perhaps chapter-level) metadata *(UCP)*
EDITORIA - Configurable archive options for completed / abandoned books *(CDL)*
WAX - Toolbar button to change case *(UCP)*
PUBSWEET CORE - Usernames allowed with special characters *(punctum)*
PUBSWEET CORE - Ask for first name and surname on sign up *(punctum)*
XSWEET - Kill automatic numbering in numbered list style *(UCP)*
EDITORIA - Allow components to move from body to front- or backmatter and vice versa (*Longleaf Services)*
EDITORIA - Add more information to the “Books” dashboard and make it sortable *(Longleaf Services)*

Substance/Wax and Editoria Upgrades

Editoria was running into several performance issues and many bugs we could not resolve without undertaking an upgrade of the underlying technologies - PubSweet and Substance. This has resulted in a complete rebuild of Editoria from the ground up and a significant upgrade of the Wax Editor.

Substance/Wax Upgrade

The Substance upgrade was necessary because copy and paste was not functioning as expected in the current Wax Editor. In addition, we could not develop the features requested. We were able to complete this upgrade before the end of 2018. This allowed Christos to focus on developing some extra features for the community (see below) before moving on to developing Wax 2. The Substance upgrade, authored by Christos, includes:

- Support for nested lists
- Support for tables
- A significant improvement to copy/paste functionality. Formatting is now preserved and documents can be copy/pasted directly from .doc(x) and Google Docs
- Access to the Substance Clipboard, which enables copy/paste functionality with Track Changes on
- Toolbars, overlays, and context menus that are now configured explicitly via config.addToolPane, which enables configurable menus
- Improved rendering performance. This can been seen especially in Firefox

- An upgraded editing surface that now allows us to override keyboard event handlers via its KeyboardManager. This has made it easier to do custom keyboard operations. For example, with Find and Replace it is now possible to go to the next match by hitting the Enter key
- The ability to add custom commands as keyboard shortcuts
- An entirely refactored MarkersManager. This component is essential to features like the Spell Checker, as well as any feature where the user might want to mark something in the document without creating an annotation
- Several improved commands including a new option disableCollapsedCursor and the ability to disable the cursor on an InlineNode
- A new surface event for the Tab key. We can now easily catch the tab event and build custom functionality around it. For example, we can now make sure 3 spaces are added on a dialogue node when Tab is pressed
- The ability to customize commands for each editing surface, controlled by a whitelist/blacklist
- A new and improved editing API
- Lots of additional bug fixes
- Caption Formatting
- Contextual Style Menus

Christos went far beyond the brief for Caption Formatting, which included italics and superscript, however Christos anticipated that the team might also like Track Changes, Comments, Spellcheck, and Find and Replace to work with image captions. This required significant work for each feature. Nothing is simple in the world of editing softwares!

Editoria Upgrade

The upgrade of Editoria to the latest PubSweet was necessary from both a functional point of view and to reduce technical debt. The 'old' PubSweet suffered from the following issues:

- The four core database entities (User, Team, Collection, Fragment) were not enough to support future development imagined for Editoria. The application would be unable to scale across many users and books given this structure.
- Editoria was not utilizing the power of a relational database. Everything was stored as a json entity, which impacted performance.
- It was an outdated server-side and client-side stack, which meant a) we were unable to take advantage of upgrades and improvements to the libraries we were utilizing and b) we could not take advantage of new approaches to the issues that PubSweet was trying to solve.
- Too much business logic was implemented on the client-side. Due to the database's unsophisticated storage model, the client was tasked with too much. This created unnecessary complexity, bugs, and performance issues.
- Unstable behaviour on real-time operations. For example, sometimes concurrent operations with multiple users produced inconsistent results in the Book Builder.

The new PubSweet addresses all of the issues outlined, above, and also now provides the additional functionality:

- We are now able to take full advantage of PostgreSQL's latest capabilities. This allows us to design database schemas custom-tailored for each application. We can now build a custom server with Editoria business logic.
- It is based on GraphQL and the Apollo framework, providing powerful real-time operations through subscriptions, as well as endless customizability.
- A new data layer that guarantees atomicity in write operations.

In summary, the above represents a complete overhaul of PubSweet, as well as replacing the storage and server-client layers. To achieve this upgrade, Alexis (later joined by Giannis) rewrote every component and the authorization (Authsome) modes. Editoria was rebuilt, top-to-bottom in just over 4 months. Not surprisingly, this took longer than the 3 months estimated. In effect it took 8 developer months. We recognised it was going to take longer at the end of last year and doubled up the team early in the new year. As a result, we reduced the total time to between 4 and 5 calendar months.

In total, Alexis and Giannis made 29,228 additions to the code. The top-level overview includes the following:

- A new, customized Editoria data model (https://gitlab.coko.foundation/editoria/editoria/issues/225).
- New database entities that focus on the needs of the publishing industry like Collection, Books, Translations, Metadata, Files, Book Divisions, Book Components (Parts, Chapters, Generic Components).
- A new server application implemented from the ground up (every operation of Editoria was rewritten).
- Concurrent operations that are now stable and reliable.
- Hardening of data storage.
- A reimplementation of real-time operations to improve concurrent user events, such as drag and drop of chapters and parts in the Book Builder. All concurrent users are now receiving updates from each other in real time ensuring everyone is seeing and working on the same thing.
- A complete redesign and implementation of a new authorization and permissions system using Authsome (https://gitlab.coko.foundation/pubsweet/pubsweet/issues/442).
- A ground up re-implementation of the Editoria UI, so that we could take advantage of the more modern Styled Components framework. This opens the doors for theming the application on a per-organization or per-publisher basis.
- Reusable and robust popup/modal components.
- Locking book components based now on web sockets, which improves stability and performance.
- A significant authorization performance improvement: we can now pre-calculate all users' rules server-side, which frees up resources in the browser. This provides users with a much faster and responsive experience.
- Upgrades to configuration components such as Babel and Webpack.
- The re-implementation of server side components from the PubSweet repo, such as the User and Team models. These are now structured in a more convenient way that gives greater flexibility, as well as reducing maintenance moving forward.
- Bonus Feature: On-the-fly transformation of component types in the Book Builder.