

個人開発した図書管理アプリを 学校に導入した話と失敗談

Presented by ゆっきー☆

Introduction

自己紹介

開発者: ゆっきー☆

年齢: 18歳(高校生)

趣味: プログラミング, 戦略ゲームetc...

生息地: 大阪府

概要

週5日の通信制高校(?)に通う高校3年生

フロントエンドエンジニアのバイトに受かったので実質現役エンジニアです

What is Kashidasu ?

Kashidasuとは？

学校で使うことを想定した図書管理webアプリ

Consept

“Simple is the best”

Features

- ▷ 必要最低限の機能
- ▷ 小規模
- ▷ 簡単に使い始めることができる

Why developed it ?

Kashidasuを開発した理由

- ▷ 以前運用されていた図書管理アプリが使いづらいという声があった
→俺やったらそんな思いさせへんのに。じゃ、作るで～
- ▷ 推薦入試の手土産的な
→ちゃんと大学合格しました(総合型選抜)
- ▷ 技術スキルの獲得
→バックエンドの知識を獲得

What is using?

技術スタック

言語: JavaScript(Node.js)

テンプレートエンジン: EJS

使用ライブラリ: Express, Quagga.js(バーコード読み取りライブラリ)等

データベース: MySQL

▷ バックエンドとフロントエンドはあえて分離しない構成
(詳細は後述)

▷ 学校で習うことができるJavascriptを使用することで、
友人や後輩もいじることができる

Architecture

アーキテクチャ

▷ あえてフロントエンドとバックエンドを分離しない小規模な構成
→ MVCモデルで開発

理由

「学校」で使ってもらうんだったら「小規模」のほうが楽でいいよね

Operating

現在の運用状況

- ▷学校のオンプレミス環境下で稼働中
(非公開)
- ▷Raspberry Pi 5でホスティング
→ 経費で落ちた

経費で落ちたRaspberry Pi 5



技術的失敗談

Database design

誤ったテーブル設計

当時の貸し出ししている本のテーブル

BOOK_ID	USER_ID	LENDING_DAY
ISBNコード	学籍番号	貸出日

▷ 「返却日」が含まれていない

→ 「返却日は貸出日から一定日数後だから、プログラム側で計算すればよくね？」 ← **アホ**

Database design

誤ったテーブル設計

なぜダメ？

- ▷ 計算ロジックが分散する
 - 赤枠に囲まれたところで分散している

実際の実装（v1.2.0時点）

```
src
├── model
│   └── lendBook.js
│       └── 本の貸出日をDBに保存する処理
├── public
│   ├── js
│   │   └── BookList.js
│   │       └── 貸出日を元に返却日を計算し、画面表示用に加工
│   └── views
│       └── BookList.ejs
│           └── 貸出状態を描画
```

Database design

誤ったテーブル設計

なぜそうしたのか

「シンプルを追求した」からです

▷ カラムを必要最低限にして「シンプル」にしたかった
→ ロジックが分散しかえって「複雑」になった

__人人人人人人人人__
> 本末転倒である <
__Y^Y^Y^Y^Y^Y^Y^Y__

Database design

誤ったテーブル設計

どうすれば良いのか？

- ▷ 返却日のカラムを含める
- ▷ 貸出・返却の責務を一元化する

実際の実装（v1.2.0時点）

Before

```
src
├── model
│   └── lendBook.js
│       └── 本の貸出日をDBに保存する処理
├── public
│   ├── js
│   │   └── BookList.js
│   │       └── 貸出日を元に返却日を計算し、画面表示用に加工
│   └── views
│       └── BookList.ejs
│           └── 貸出状態を描画
```

After

```
src
├── model
│   └── lendBook.js
│       └── 貸出日・返却日をDBに保存
├── public
│   ├── js
│   │   ├── BookList.js
│   │   │   └── DBから取得した値をそのまま表示
│   └── views
│       └── BookList.ejs
│           └── 貸出状態を描画
```

Results

得た学び

必要最低限を追求しすぎない

▷「必要最低限」ではなく「いらないものを省く」考え方であるべき

設計してから実装すべき

▷.mdで軽く設計してから実装したら失敗が大きく減ることに気づいた

Roadmap

今後の展望

一度リライトする

- ▷ 人様に見せられるコードに書き変える
 - このままだと、保守開発する後輩が可哀想

READMEを充実させる

- ▷ このままだと情報が少ないようにも思えるので充実化を図る

Thank you !!

ご清聴ありがとうございました

Kashidasu関連リンク

Github <https://github.com/yukki072417/Kashidasu>

Qiita <https://qiita.com/yukki072417/items/48743564ef1b6a19b1c4>

Presented by ゆっきー☆