Node.jsを用いた
出席管理システムの設計

足利大学 工学部 創生工学科 情報システムデザイン学系
学籍番号 315156 研究者名 續橋 涼
指導教員 平石 広典

はじめに

Webアプリケーション
出欠席管理システム

基本的機能
・ルーティング
・ページネーション
・etc..

システム構成

学習コスト

- Node.js ⇨ JavaScript

生産性

- Express ⇨ 自動生成

運用性

- SQLite3 ⇨ 単一ファイル

Node.js

- JavaScriptのランタイム環境
- シングルスレッド
- サーバー機能
- ノンブロッキングI/O
- イベントループ
- npm

npm

Node Package Manager
Node.jsのパッケージを管理するツール

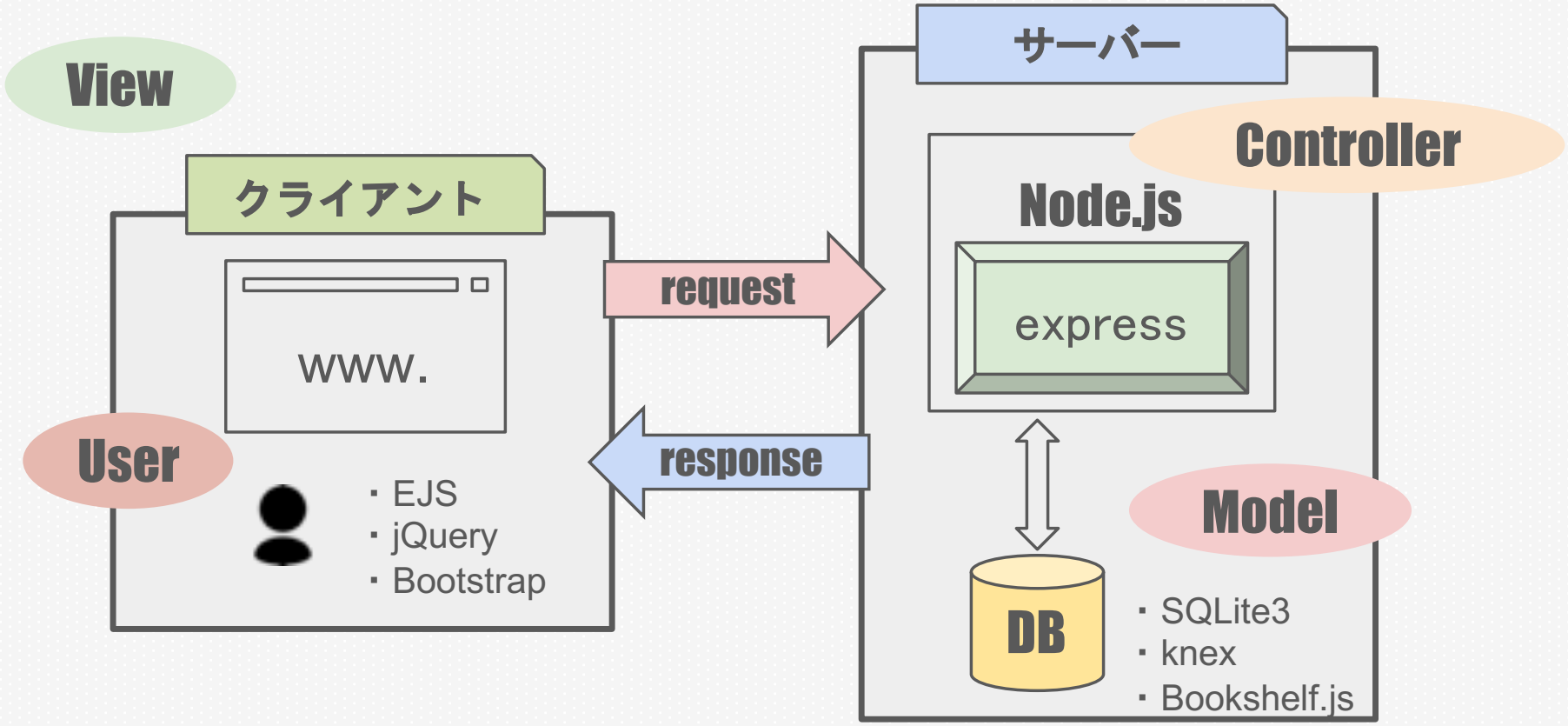
expressなど
開発効率を上げることが可能

express

- Node.js上で動作する
  Webアプリケーションフレームワーク
- MVCモデル
- 導入事例
  Netflix

express

Model View Controller



express

- Express generator
  expressのよく使う機能をまとめ，ファイルを分割

- Express session
  クライアントごとに値を保持する仕組み

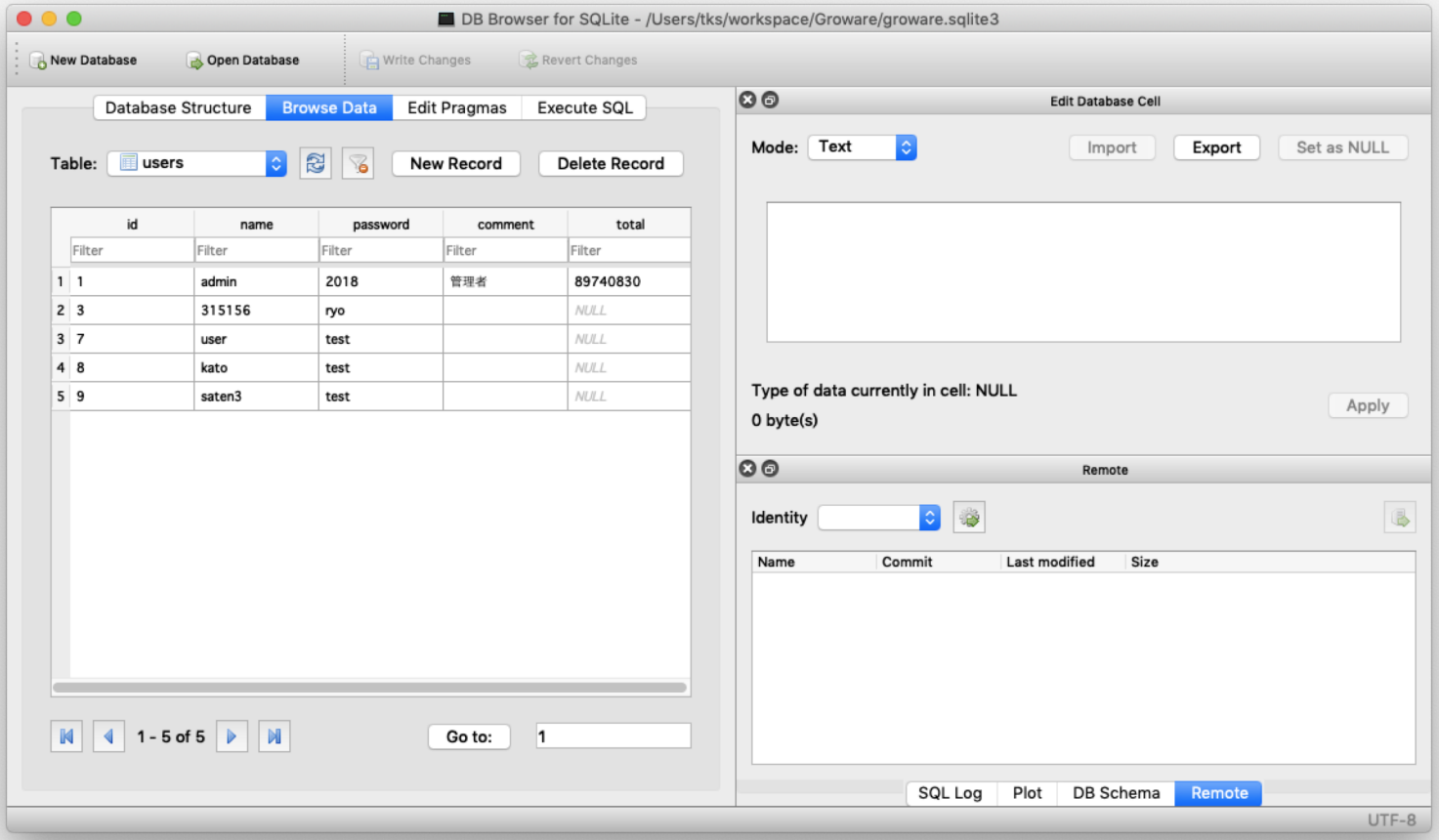
- Express validator
  入力された値をチェック

SQLite3

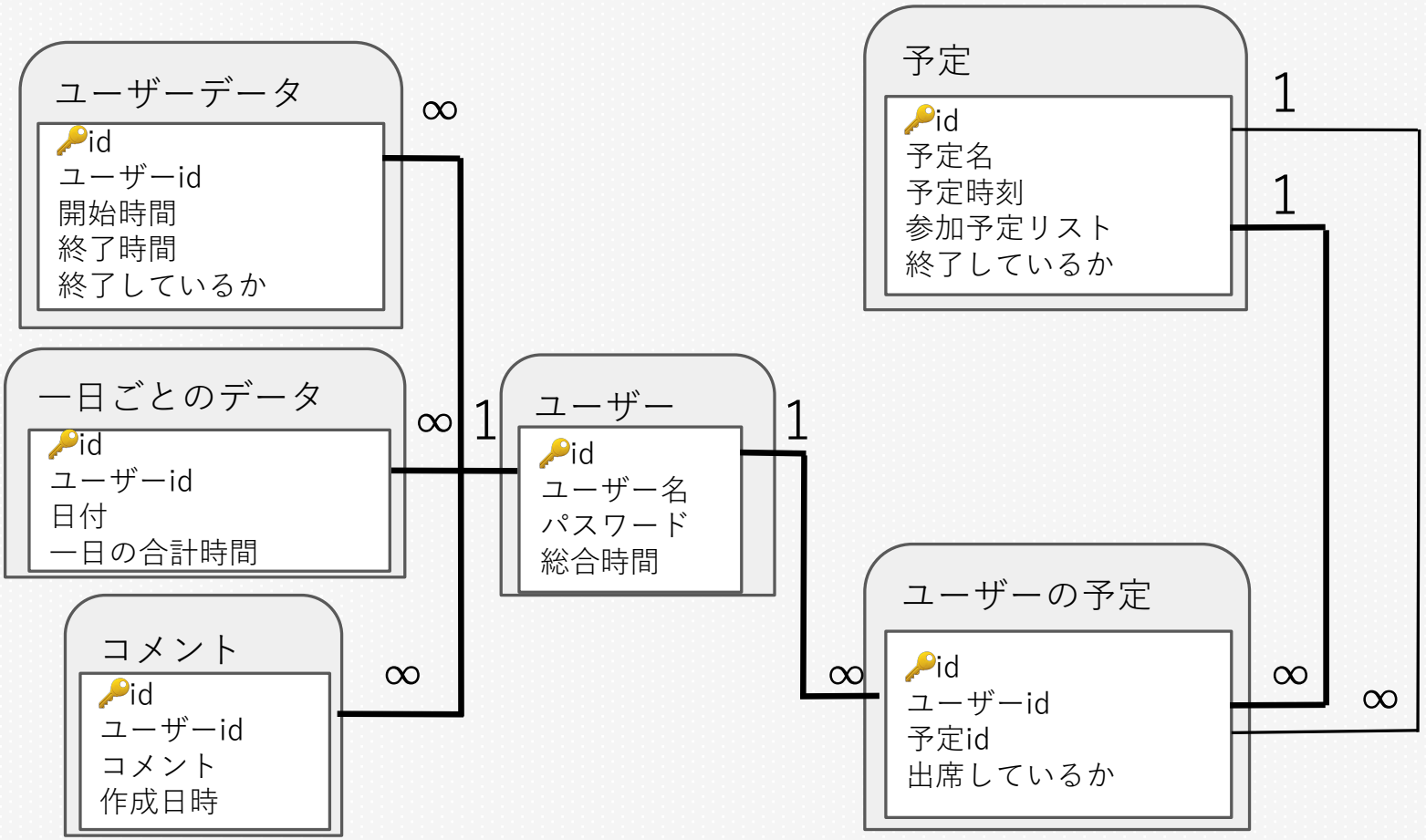
- 関係データベース管理システム(RDBMS)
- 組み込み型データベースエンジン
- データの保存は単一ファイルのみ

データ型名	データの内容
NULL	NULL型
INTEGER	符号付整数
REAL	浮動小数点数
TEXT	テキスト
BLOB	バイナリ

DB Browser for SQLite



テーブル


ユーザーデータ
id
ユーザーid
開始時間
終了時間
終了しているか
一日ごとのデータ
id
ユーザーid
日付
一日の合計時間
コメント
id
ユーザーid
コメント
作成日時
ユーザー
id
ユーザー名
パスワード
総合時間
予定
id
予定名
予定時刻
参加予定リスト
終了しているか
ユーザーの予定
id
ユーザーid
予定id
出席しているか
∞
1
∞
1
∞
1
∞
1
∞
∞

Bookshelf.js

- Node.jsで動作するORM(Object Relational Mapping)
- Knexというクエリビルダーを元に実装
- JavaScriptでデータベースを操作可能
  =>SQLの習得が不要

- SQLite3, PostgreSQL, MySQL 対応

EJS (Embedded JavaScript Templates)

テンプレートエンジン
HTMLの中にコードを埋め込む
他のテンプレートエンジンに比べ
理解しやすい構造

EJS

テンプレート × データ = HTML

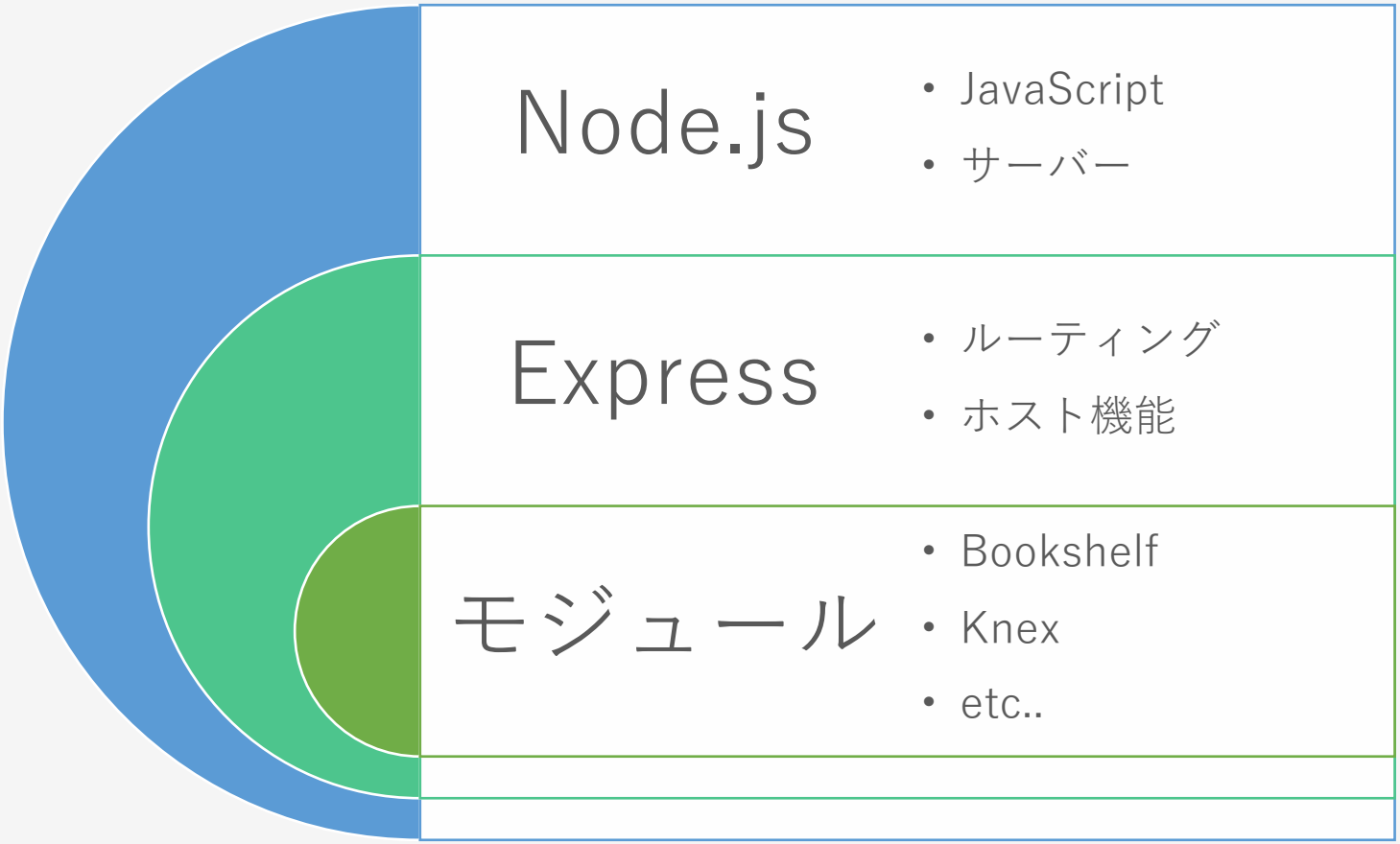
```
<h1><%= title %></h1>

<p><%= date %></p>

<p>NAME :
<%= name %> </p>
```

```
data = {
    title : '出席管理',
    date : now,
    name : session.user
};
```

出席管理

2018/12/02
NAME : USER1

```
new EJS({url : 'index.ejs' }).render(data)
```


Node.js
• JavaScript
• サーバー
Express
• ルーティング
• ホスト機能
モジュール
• Bookshelf
• Knex
• etc..

前期の課題点

- 管理者はデータベースを直接手を加える必要があった.
- 毎週実行の予定を作成することができなかった.
- 大学内でのLAN認証

新機能

✓ユーザー名変更

✓パスワード変更

✓ユーザーの削除

新機能


133.106.123.114
User一覧
adminさんおつかれさまです。
id
name
1
admin
password
変更
9
315156
name
変更
password
変更
削除
12
315123
name
変更
password
変更
削除
13
315025
name
変更
password
変更
削除
14
315158
name
変更
password
変更
削除
15
315189
name
変更
password
変更
削除
16
315194
name
変更
password
変更
削除

Users > all.ejs

```
<table class="table table-hover" >
<tr>
    <th>id</th>
    <td>name</td>
</tr>

 <% for(var i in collection){ %>
    <tr>
        <th><%=collection[i].attributes.id %></th>
        <td>
            <form action="/home/<%= collection[i].attributes.id %>/1" method="POST" style="float: left; margin-right: 10px;">
                <input class="btn " type="submit" value="<%=collection[i].attributes.name %>">
            </form>
        </td>
        <td>
            <form action="/users/all/<%= collection[i].attributes.id %>/name" method="POST"
                onsubmit='return confirm("<%= collection[i].attributes.name %>の名前を変更しますか？");'>
            <% if(collection[i].attributes.name != "admin") {%>
            <input type="text" name="name" placeholder="name" value="<% form.name %>">
            <input type="submit" value="変更">
            <% } %>
            </form>
        </td>
            <td>
            <form action="/users/all/<%= collection[i].attributes.id %>/password" method="POST"
                onsubmit='return confirm(" <%= collection[i].attributes.name %>のパスワードを変更しますか？");'>
                <input type="text" name="password" placeholder="password" value="<% form.password %>">
                <input type="submit" value="変更">
            </form>
        </td>
        <% if(collection[i].attributes.name != "admin") {%>
```

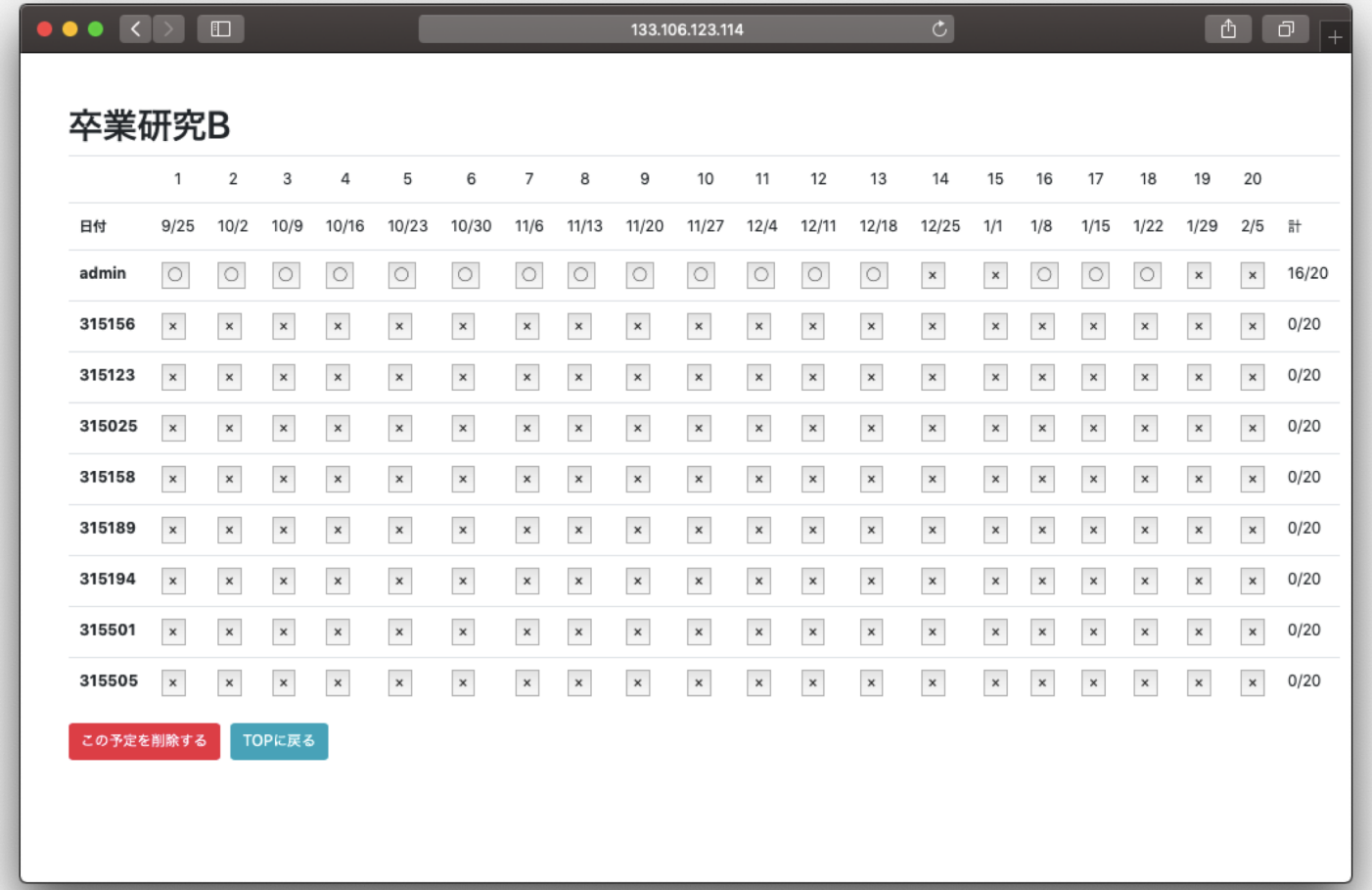
users.js

```
123   //ユーザー削除
124   router.post('/all/delete/:id', (req,res,next) =>{
125     //ユーザーの削除
126     new User({id: req.params.id}).destroy().then((model)=>{});
127
128     res.redirect('/users/all');
129
130
131   })
132
133   //パスワード変更
134   router.post('/all/:id/password', (req,res,next)=>{
135
136     new User(req.body)
137       .where('id', '=', req.params.id)
138       .save({}, { method: 'update' })
139       .then((model) => {
140     });
141     res.redirect('/users/all');
142   })
143
144   //ユーザー名編集
```

新機能

- ✓出欠席の編集
- ✓予定の削除
- ✓毎週行う予定の自動生成

新機能


133.106.123.114
卒業研究B
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
日付 9/25 10/2 10/9 10/16 10/23 10/30 11/6 11/13 11/20 11/27 12/4 12/11 12/18 12/25 1/1 1/8 1/15 1/22 1/29 2/5 計
admin ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ × × ○ ○ ○ × × 16/20
315156 × 0/20
315123 × 0/20
315025 × 0/20
315158 × 0/20
315189 × 0/20
315194 × 0/20
315501 × 0/20
315505 × 0/20
この予定を削除する
TOPに戻る

学内LAN自動ログインツール

Python	ブラウザ
• Selenium • Platform • Schedule	• Chrome • ChromeDriver

学内LAN自動ログインツール

```
52
53      # ユーザID
54      Username = driver.find_element_by_css_selector("#userId")
55      Username.send_keys(u_text)
56
57      time.sleep(1)
58      # パスワード
59      Password = driver.find_element_by_css_selector("#userPass")
60      Password.send_keys(p_text)
61
62      time.sleep(1)
63
64      PolicyCheck = driver.find_element_by_css_selector("#policyCheck")
65      PolicyCheck.click()
66
67      time.sleep(1)
68
69      SubmitButton = driver.find_element_by_css_selector("#submitButton")
70      SubmitButton.click()
71
72
73      print("ログイン成功")
```

学内LAN自動ログインツール

・24時間稼働

・AM4:00 にログアウト
・AM5:00 にログイン

if ログイン == False
・AM5:30 に再ログイン

プロジェクト管理

- GitHub
- Gitを利用したプロジェクト管理ツール
- 変更履歴が全て保存される

- 開発効率を上げる
- 問題の管理がしやすい

まとめ

- JavaScript + HTML + CSSのみで実装可能
  => 学習コスト削減
- npmを利用して様々なモジュールが使える
  => 生産性向上
- SQLite3
  => 運用性向上

課題点

✓データベースの正規化

✓リファクタリング
　　冗長性の排除

✓パスワードなどの暗号化

✓脆弱性をチェックするため
　　ペネトレーションテストを行う(skipfishなど)