# 楽譜管理ソフトウェア 内部設計書 v1

dir-en-gray

2023年12月9日

## 第1章

## システム概要

### 1.1 ユーザ側システム一覧

- 基本機能
  - ユーザ登録機能
  - ログイン機能
  - ログアウト機能
  - ユーザ情報編集機能
  - ユーザ削除機能
- 楽譜データに関する機能
  - 楽譜データ登録機能
  - 楽譜データ編集機能
  - 楽譜データ削除機能
  - 楽譜データ一覧閲覧機能
  - 楽譜データ詳細閲覧機能
- 楽譜検索機能
  - 作曲者, 編曲者からの検索機能
  - 曲名からの検索機能
  - グレード(難易度)からの検索機能
  - 使用楽器からの検索機能
- 楽譜ソート機能
  - 曲名から,五十音順にソートする機能
  - 作曲者, 編曲者を五十音順にソートする機能
  - グレード順にソートする機能
  - 演奏時間の短い順にソートする機能
- お問い合わせ画面表示機能
  - お問い合わせ機能
- 広告表示機能
  - 広告表示機能

## 1.2 管理者側システム一覧

- 基本機能
  - ログイン機能
  - ログアウト機能
  - 管理者情報編集機能
- ユーザに対する機能
  - ユーザデータ一覧閲覧機能
  - ユーザデータ詳細閲覧機能
  - ユーザ情報編集機能
  - ユーザ削除機能
  - 楽譜データ削除機能
- ユーザ検索機能
  - ユーザ名から検索する機能
  - ユーザ ID から検索する機能
- ユーザソート機能
  - ユーザ名を五十音順にソートする機能
- 楽譜検索機能
  - 作曲者, 編曲者からの検索機能
  - 曲名からの検索機能
  - グレード (難易度) からの検索機能
  - 使用楽器からの検索機能
- 楽譜ソート機能
  - 曲名から、五十音順にソートする機能
  - 作曲者、編曲者を五十音順にソートする機能
  - グレード順にソートする機能
  - 演奏時間の短い順にソートする機能
- 広告登録機能
  - 広告登録機能

# 第2章

# システム実装方法

## 動作環境

ユーザ、管理者共に Javascript 使用に対応したブラウザ

## 開発環境

OS	Windows, Mac, Linux
開発フレームワーク	Ruby on Rails バージョン7以降
Ruby バージョン	3.2.2
開発言語	Ruby, HTML, ERB, SCSS
DBMS	SQLite
サーバ	AWS
IDE	Visual Studio Code
バージョン管理	Git, GitHub

## 第3章

# 規約

## 3.1 コーディング規約

ここでは、コーディングに関する規約を定める.変数名、ファイル名については、3.2節で述べる.ここではコード整形についてのみ言及する.

コード整形には、rubocop というツールを用いる、rubocop が定めるルールに従う. rubocop の設定ファイルを以下に示す.

指摘された箇所は、全て確認、訂正する.

```
inherit_from: .rubocop_todo.yml
require:
- rubocop-rails
- rubocop-rspec
- rubocop-performance
NewCops: enable
SuggestExtensions: false
Max: 30
Metrics/PerceivedComplexity:
Max: 10
# ブロックの長さ制限
Metrics/BlockLength:
Max: 20
Exclude:
- 'config/environments/development.rb'
# メソッド内のながさ制限
Metrics/MethodLength:
Max: 20
# selfをつけるかどうか
Style/RedundantSelf:
Exclude:
- 'app/models/user.rb'
# validation をスキップする警告を無視
Rails/SkipsModelValidations:
Exclude:
- 'app/models/user.rb'
- 'test/helpers/sessions_helper_test.rb'
```

図 3.1: rubocop 設定ファイル

### 3.2 Rails フレームワーク規約

Rails のフレームワークに従って実装する. Rails は MVC モデルを採用しているため, それぞれに命名は Rails が定める規則に従う.

#### • Model

1. ファイル

モデルファイルは, app/model ディレクトリ内に配置する. モデルファイル名は, tableName.rb とする. tableName には作るデータテーブル名を単数形・小文字で書く. 例えば, User に関するデータテーブルを作りたい場合には, user.rb とする.

2. クラス

1つのファイルに、1つのクラスを記述する. クラス名は、ファイル名の先頭を大文字にしたものを使う. 例えば、ファイル名が user.rb である場合、クラス名は User である. 必ず、ApplicationRecord を継承する.

- 3. マイグレーションファイル
  - a. ファイル

マイグレーションファイルは、db/migrate ディレクトリ内に配置する.ファイル名は timestamp\_verb.rb とする. verb には動作を記入、ファイル名はスネークケースを用いて表現し、全て小文字とする. 例えば User データベース内の Email 列にインデックスを追加するマイグレーションファイル名は、timestamp\_add\_index\_to\_users\_email.rb である.

b. クラス

1つのファイルに、1つのクラスを記述する。クラス名は、ファイル名の timestamp と拡張子を除いてスネークケースで表現したものを用いる。上記の例に倣うと、クラス名は AddIndexToUsersEmail となる。

必ず, ActiveRecord::Migration[7.1]を継承する.

c. 関数名

関数名は change とする. 1つのクラスに1つの関数のみを定義する.

- Controller
  - 1. ファイル

コントローラファイルは app/controllers ディレクトリ内に配置する. コントローラのファイル名は, viewContents\_controller.rb とする. viewContents は表示したいものを複数形・小文字で書く. 例えば, User に関するコントローラの場合は, users\_controller.rb とする.

2. クラス

1つのファイルに、1つのクラスを記述する.クラス名はキャメルケースを用いて表現

する. 必ず, ApplicationController を継承する. 例えば, users\_controller.rb 内で定義されているクラス名は, UsersController である.

#### 3. アクション

コントローラ内のアクション名は、全て小文字で表現する.原則、以下のアクション名を用いる.これ以外に定義する場合、全体で相談する.

- index
- home
- new
- create
- edit
- show
- update
- destroy

#### • View

#### 1. ファイル

ビューファイルは、app/view ディレクトリ内に配置する.ディレクトリ内にサブディレクトリを"コントローラ名"で作り、その中に各アクションに対応するビューを作る.例えば、users\_controller.rb 内で定義されたアクション new に対応するビューを作るもち、そのファイル名は app/views/users/new.html.erb となる.

## 3.3 開発規約

ソースコードの差分管理には GitHub を用いる. 差分ログを GitHub 上へ反映することを「PUSH」と書く. また, CI を GitHub Actions で実行する. 内容は, rubocop と, rails test である.

#### 3.3.1 PUSH する前に

PUSH する前に、テストを走らせ、rubocop を実行する. 全て成功したら PUSH する.

- \$ bundle exec rubocop
- \$ bundle exec rails test

#### 3.3.2 ブランチ

- メインブラインチを develop とする.
- develop ブランチへ PUSH しないこと.必ずブランチを切って編集し、そのブランチ名で

PUSH する. 以下の項目が成功したら develop ヘマージする.

- 1. GitHub 上でプルリクエストを作成する.
- 2. CI が通っていることを確認する.
- 3. 溝口洸熙が Approve する.
- ブランチ名は以下のように定める.
  - docs/\*ドキュメントに関する更新
  - fix/\* バグの更新
  - o future/\*新規機能の追加
- 他ブランチ上で作業する時は、定期的に以下のコマンドを実行し、developの更新を取り入れる.

\$ git merge develop

#### 3.3.3 タグ

リリース作業には git Tag を用いる. タグの命名は以下のように行う.

v[major-Version].[minor-Version]

開始番号を 0 とし、バグ修正を minor-Version Up, 新機能追加を major-Version Up とする. タグの更新を GitHub Actions で検知し、AWS へオートデプロイ(CD)する.

## 第4章

# モジュール設計

### 4.1 前提

我々はRuby on rails を用いて開発を行う. この言語の慣習に則り, これ以降,

- $\bullet$  index
- new
- show
- $\bullet$  edit
- create
- $\bullet$  destroy
- home

の7つは「アクション」と呼び、これ以外の関数を全て「メソッド」と呼ぶ.

アクションはコントローラが異なれば同名のものを使用するため, モジュール ID を「Controller 名. アクション名」とする. また, 1 アクション 1 機能を持つ.

## 4.2 モジュール詳細

以下に、各モジュールについて「定義書」「フロー図」の順で示す.

モジュール	/定義			
	システム名	楽譜管理ソフトウェア	バージョン	
管理情報	工程名	内部設計	作成日	2023/12/2
口工旧私	作成者	奥平舜理	更新日	2023/12/5
	概要	UsersController 内の inde		2023/12/3
	<u> </u>	UsersController	^/// =/	
基本情報	モジュール名	index		
	モジュール日 モジュールID	UsersController.index		
処理説明	C/I NID	O Set S Contitioner. Index		
【処理内容	z]			
	►』 -覧を表示する.			
【処理手順				
		fい,管理者でない場合は Sco	procControllor homo を呼	バルナ
		い,管理者でない場合は 300 と用いてデータベースからユ-		
	ンへの all スフットを に格納する.	日用いてナーダベースからユー	- 9 0 一見で取付し,12	ヘメノヘ <b>を</b> 奴
	に合かりする. sers/index.html.erb	も世面まる		
	isers/index.ntmi.erb	で抽画する.		
【補足】				
İ				
入力值説明	1			
なし				
/4 U				
出力值説明	1			
. 10	_			
ユーザー賢	Ī,			
他クラス・	関数との関係			
Applicatio	nController クラスを	継承する.		

図 4.1: UsersController.index 定義書

所属クラス	UsersController	作成日	2023/12/08
機能名	ユーザー覧表示	更新日	2023/12/08
モジュールID	UsersController.index	作成者	奥平舜理
使用モジュールID	UsersController.home		

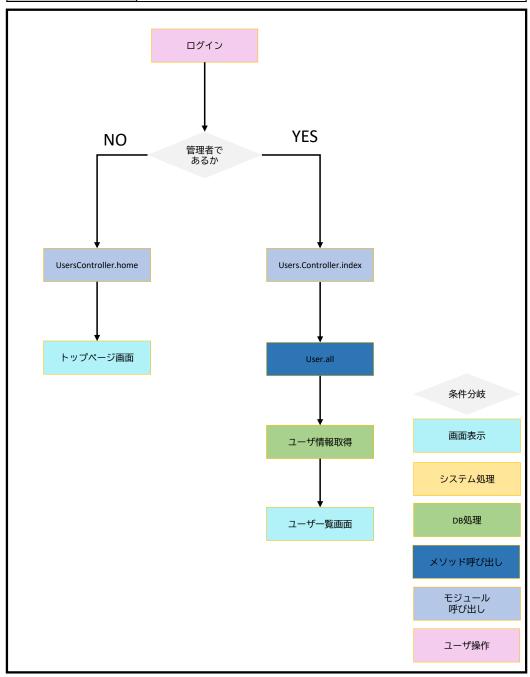


図 4.2: UsersController.index フロー図

モジューノ	レ定義			
	システム名	楽譜管理ソフトウェア	バージョン	
管理情報	工程名	内部設計	作成日	2023/12/2
	作成者	奥平舜理	更新日	2023/12/3
	概要	UsersController 内の home	e アクション	
++ 1 1++0	所属クラス	UsersController		
基本情報	モジュール名	home		
	モジュールID	UsersController.home		
処理説明				
		して,楽譜データ一覧を生成す erb を描画する.	<b>వ</b> .	
入力値説明	月			
ユーザid				
出力值説明	月			
楽譜データ	女の一覧			
他クラス・	・関数との関係			
Applicatio	nController クラス	を継承する.		

図 4.3: UsersController.home 定義書

所属クラス	UsersController	作成日	2023/12/08
機能名	楽譜データ一覧表示	更新日	2023/12/08
モジュールID	UsersController.home	作成者	奥平舜理
使用モジュールID	UsersController.index		

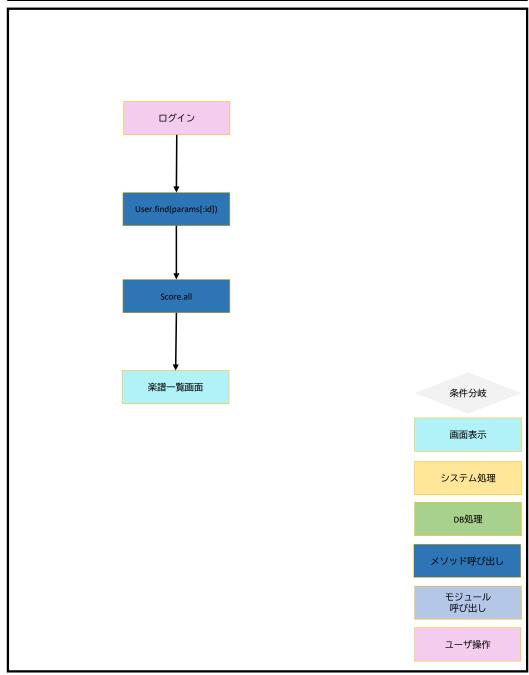


図 4.4: UsersController.home フロー図

モジュール	レ定義			
	システム名	楽譜管理ソフトウェア	バージョン	
管理情報	工程名	内部設計	作成日	2023/12/1
	作成者	中村祐貴	更新日	2023/12/6
	概要	ScoresController 内の sho		2020, 12, 0
	所属クラス	ScoresController		
基本情報	モジュール名	show		
	モジュールID	ScoresController.show		
処理説明				
【処理内容	z1			
	☆ 田画面を表示する.			
【処理手順				
		ドを用いて,インスタンスを多	5数 @agara I-枚姉士ス	
			<b>之奴 @SCUTE に伯利 9 る.</b>	
	ノッドの引数は para scores/show.html.e			
	scores/snow.ntmi.e	rD を抽凹する.		
【補足】				
入力值説明	Ħ			
score_id				
30016_Iu				
出力值説明				
なし				
61				
他クラス・	・関数との関係			
Applicatio	nController を継承 <sup>-</sup>	<b>ま</b> る		
Applicatio	incontroller を秘承	૪ ૾.		

図 4.5: ScoresController.show 定義書

所属クラス	ScoresController	作成日	2023/12/7
機能名	楽譜情報詳細	更新日	2023/12/8
モジュールID	ScoresController.show	作成者	中村祐貴
使用モジュールID			

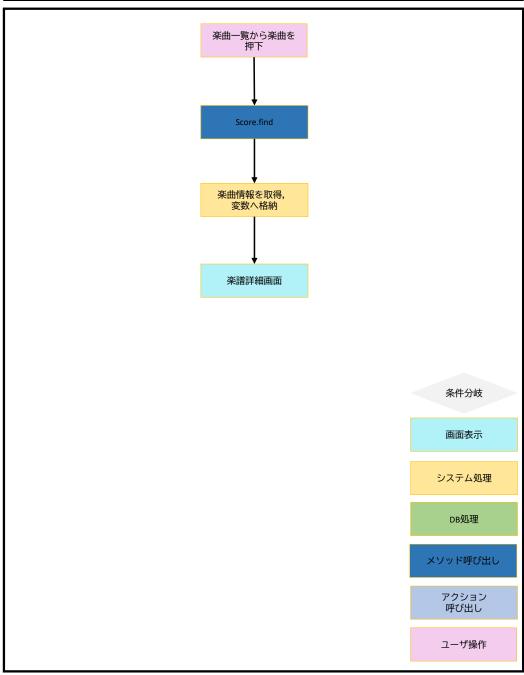


図 4.6: ScoresController.show フロー図

モジューハ	定義					
	システム名	楽譜管理ソフトウェア	バージョン			
管理情報	工程名	内部設計	作成日	2023/12/2		
	作成者	中村祐貴	更新日	2023/12/		
	概要	UsersController 内の show	アクション			
基本情報	所属クラス	UsersController	UsersController			
至午旧刊	モジュール名	show				
	モジュールID	UsersController.show				
処理説明						
【処理内容	<b>F</b> ]					
ユーザの訓	詳細画面を表示する					
【処理手順	頁】					
L KEYE J M	· <del>=</del>					
	- _user が admin か	user か判定する.current_user	が uesr なら自身の use	er_id しか取得		
	_	user か判定する.current_user	が uesr なら自身の use	er_id しか取得		
1. current できない	`.	user か判定する.current_user 画面,user なら自身のユーザ情幸				
1. current できない (admin	- ゝ. ならユーザの詳細i		W詳細画面が表示される			
1. current できない (admin 2. User ク	- ゝ. ならユーザの詳細i	画面,user なら自身のユーザ情幸 ドを用いて,インスタンスを変	W詳細画面が表示される			
1. current できない (admin : 2. User ク find メン	- ゝ. ならユーザの詳細區 ラスの find メソッ	画面,user なら自身のユーザ情幸 ドを用いて,インスタンスを変詞 ams[:id] とする.	W詳細画面が表示される			
1. current できない (admin : 2. User ク find メン	- 、 ならユーザの詳細區 ラスの find メソッ ノッドの引数は par	画面,user なら自身のユーザ情幸 ドを用いて,インスタンスを変詞 ams[:id] とする.	W詳細画面が表示される			
1. current できない (admin: 2. User ク find メン 3. views/レ 【補足】	・、 ならユーザの詳細証 ラスの find メソッ ノッドの引数は par ısers/show.html.er	画面,user なら自身のユーザ情幸 ドを用いて,インスタンスを変詞 ams[:id] とする.	仮詳細画面が表示される 数 @user に格納する.	. )		
1. current できない (admin: 2. User ク find メン 3. views/レ 【補足】	・、 ならユーザの詳細証 ラスの find メソッ ノッドの引数は par ısers/show.html.er	画面,user なら自身のユーザ情報 ドを用いて,インスタンスを変態 ams[:id] とする. b を描画する.	仮詳細画面が表示される 数 @user に格納する.	. )		
1. current できない (admin: 2. User ク find メン 3. views/レ 【補足】 ユーザがし	・、 ならユーザの詳細証 ラスの find メソッ ノッドの引数は par ısers/show.html.er	画面,user なら自身のユーザ情報 ドを用いて,インスタンスを変態 ams[:id] とする. b を描画する.	仮詳細画面が表示される 数 @user に格納する.	. )		
1. current できない (admin: 2. User ク find メン 3. views/レ 【補足】 ユーザがし	・、 ならユーザの詳細証 ラスの find メソッ ノッドの引数は par ısers/show.html.er	画面,user なら自身のユーザ情報 ドを用いて,インスタンスを変態 ams[:id] とする. b を描画する.	仮詳細画面が表示される 数 @user に格納する.	. )		
1. current できない (admin 2. User ク find メン 3. views/レ 【補足】 ユーザがし	・、 ならユーザの詳細証 ラスの find メソッ ノッドの引数は par ısers/show.html.er	画面,user なら自身のユーザ情報 ドを用いて,インスタンスを変態 ams[:id] とする. b を描画する.	仮詳細画面が表示される 数 @user に格納する.	. )		
1. current できない (admin 2. User ク find メン 3. views/レ 【補足】 ユーザがし	・、 ならユーザの詳細証 ラスの find メソッ ノッドの引数は par ısers/show.html.er	画面,user なら自身のユーザ情報 ドを用いて,インスタンスを変態 ams[:id] とする. b を描画する.	仮詳細画面が表示される 数 @user に格納する.	. )		
1. current できない (admin: 2. User ク find メン 3. views/レ 【補足】 ユーザがし	・、 ならユーザの詳細証 ラスの find メソッ ノッドの引数は par ısers/show.html.er	画面,user なら自身のユーザ情報 ドを用いて,インスタンスを変態 ams[:id] とする. b を描画する.	仮詳細画面が表示される 数 @user に格納する.	. )		

user\_id

#### 出力值説明

なし

#### 他クラス・関数との関係

ApplicationController を継承する.

所属クラス	UsersController	作成日	2023/12/7
機能名	ユーザ情報詳細	更新日	2023/12/8
モジュールID	UsersController.show	作成者	中村祐貴
使用モジュールID			

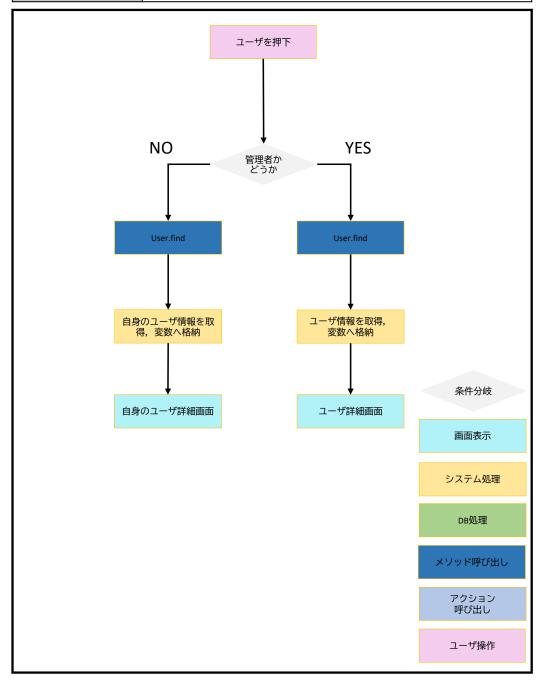


図 4.8: UsersController.show フロー図

	人民			
	システム名	楽譜管理ソフトウェア	バージョン	
管理情報	工程名	内部設計	作成日	2023/12/4
	作成者	山本祥弘	更新日	2023/12/5
	概要	ScoresController 内の new	アクション	
甘土桂起	所属クラス	ScoresController		
基本情報	モジュール名	new		
	モジュールID	ScoresController.new		
処理説明				
【処理内容	<b>F</b> ]			
新しい楽譜	<b>普データを作成するへ</b>	ページを表示するアクション.		
【処理手順	頁】			
1. views/s	cores/new.html.erb	を描画する.		
【補足】				
入力値説明				
なし				
	-			
出力値説明	3			
なし				
0.0				
他クラス・	関数との関係			
Applicatio	nController クラスを	と継承する.		

図 4.9: ScoresController.new 定義書

所属クラス	ScoresController	作成日	2023/12/07
機能名	ユーザ削除	更新日	2023/12/07
モジュールID	ScoresController.new	作成者	山本祥弘
使用モジュールID			

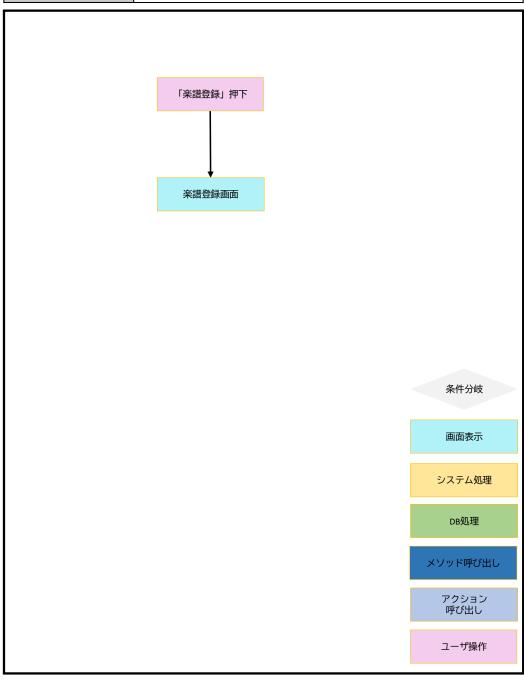


図 4.10: ScoresController.new フロー図

モジュール	定義			
	システム名	楽譜管理ソフトウェア	バージョン	
管理情報	工程名	内部設計	作成日	2023/12/4
	作成者	山本祥弘	更新日	2023/12/5
	概要	UsersController 内の new フ	アクション	
++ ++ += +D	所属クラス	UsersController		
基本情報	モジュール名	new		
	モジュールID	UsersController.new		
処理説明				
【処理内容	<b>F</b> ]			
サインアッ	,プ画面を表示するア	クション.		
【処理手順	<b>〔</b> 】			
1. views/u	ısers/new.html.erb を	描画する.		
【補足】				
入力值説明	1			
なし				
14 C				
出力值説明	1			
なし				
/lk 4 = =	BB*b b o BB /5			
他クラス・	関数との関係			
Applicatio	nController クラスを	継承する		

図 4.11: UsersController.new 定義書

所属クラス	UsersController	作成日	2023/12/07
機能名	サインアップ画面表示	更新日	2023/12/08
モジュールID	UsersController.new	作成者	山本祥弘
使用モジュールID			

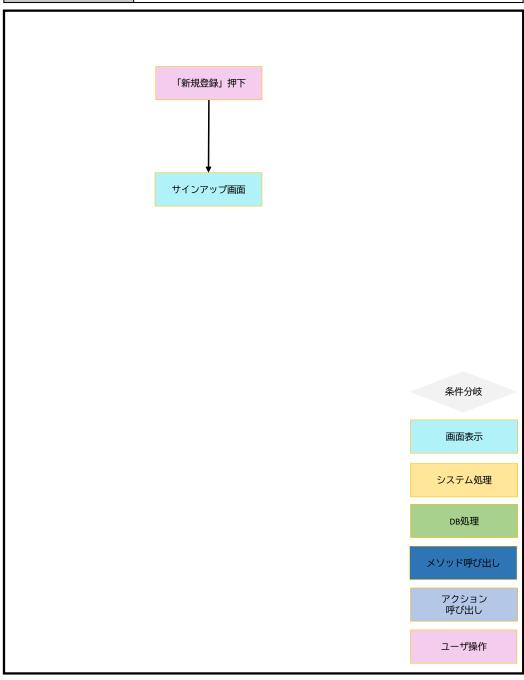


図 4.12: UsersController.new フロー図

モジュール						
	システム名	楽譜管理ソフトウェア	バージョン			
管理情報	工程名	内部設計	作成日	2023/12/3		
1 117 16	作成者	山田滉希	更新日	2023/12/5		
	概要	ScoresController 内の edit アク		2020, 12, 0		
	所属クラス	ScoresController				
基本情報	モジュール名	edit				
	モジュールID	ScoresController.edit				
処理説明	1 - 1 - 1 - 1					
【処理手順 1. current_ 2. 一致して find メソッ	・ ! について編集ページを [] _user_id と編集する対	象の user_id が一致しているかの の find メソッドを用いて,インス id] とする.)		core に格納する.(		
入力値説明						
score_id,	user_id, current_use	or_id				
出力値説明	]					
なし						
他クラス・	関数との関係					
Applicatio	nController クラスを紛	迷承する.				

図 4.13: ScoresController.edit 定義書

モジューハ	レ定義			
	システム名	楽譜管理ソフトウェア	バージョン	
管理情報	工程名	内部設計	作成日	2023/12/
	作成者	山田滉希	更新日	2023/12/
	概要	UsersController 内の edit :	アクション	
基本情報	所属クラス	UsersController		
至午旧刊	モジュール名	edit		
	モジュールID	UsersController.edit		
処理説明				
【処理手順 1. current current_us 2. 一致して 2. User ク 数は paran	_user_id と admin_ ser_id に紐づけられ こいるなら変数 @us	id が一致しているか判定し, - れた user_id しか格納できない. ser には任意の user_id を格納て ドを用いて,インスタンスを変	できる.	
入力値説明				
user_id, ‹	current_user_id			
出力値説明	月 一			
なし				
他クラス・	・関数との関係			
Applicatio	nController クラス	を継承する.		_

図 4.14: UsersController.edit 定義書

モジュール	モジュール定義				
	システム名	楽譜管理ソフトウェア	バージョン	1.0.2	
管理情報	工程名	内部設計	作成日	2023/12/03	
	作成者	三上柊	更新日	2023/12/08	
	概要	ScoresController 内の create アクション			
基本情報	所属クラス	ScoresController			
<b>基</b> 华 旧 和	モジュール名	create			
	モジュールID	ScoresController.create			

#### 【処理内容】

新たに作成された楽譜データをDBに登録する.

#### 【処理手順】

- 1. score\_params メソッドにて入力を取得する
- 2. Score クラスの new メソッドに score\_params の戻り値を引数として渡し、 @score 変数に結果を格納する.
- 3. save メソッドを用いてDBに @score の中身を保存する.
- 4. 保存が成功しているかを判定する.
- 5. 成功していれば成功を意味する flash を含めて ScoresController.index を呼び出す.
- 6. 失敗していればエラーを意味する flash を含めて ScoresController.new を呼び出す.

#### 【補足】

- ・score\_params メソッドは private である. new での入力内容を取得する.
- ・flashとは、リダイレクト時に、ページ上に一度だけメッセージを表示するメソッドである。
- ・saveとは、Railsに標準搭載されたDBにインスタンスを追加するメソッドである.

#### 入力值説明

:composer:作曲者,:arranger:編曲者,

:name:曲名, :grade:難易度, :time:演奏時間, 使用楽器::piccolo:ピッコロ,:c\_flute:フルート等

#### 出力值説明

#### 他クラス・関数との関係

ApplicationController クラスを継承する.

score\_params メソッドを作成,利用する.このメソッドは private である.

図 4.15: ScoresController.create 定義書

所属クラス	ScoresController	作成日	2023/12/06
機能名	楽曲登録	更新日	2023/12/08
モジュールID	ScoresController.create	作成者	三上柊
使用モジュールID	ScoresController.index, UsersController.home		

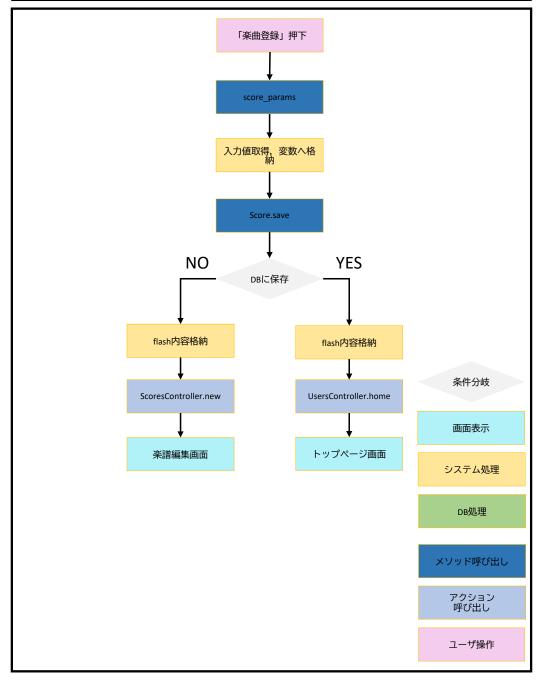


図 4.16: ScoresController.create フロー図

モジュール	モジュール定義					
	システム名	楽譜管理ソフトウェア	バージョン	1.0.2		
管理情報	工程名	内部設計	作成日	2023/12/03		
	作成者	三上柊	更新日	2023/12/08		
	概要	UsersController 内の create アクション				
基本情報	所属クラス	UsersController				
<b></b>	モジュール名	create				
	モジュールID	UsersController.create				

#### 【処理内容】

新規のユーザをDBに登録する.

#### 【処理手順】

- 1. user\_params メソッドを用いて new ページでの入力内容を取得する.
- 2. User クラスの new メソッドを用いて,入力をインスタンス変数 @user に格納する.
- 3. save メソッドを用いて @user をDBに保存する.
- 4. 保存が成功したか判定する.
- 5. 成功していればログイン処理を行い,成功を意味する flash を含めて,ScoresController.index を呼び出す.
- 6. 失敗していたらエラーを意味する flash を含めて UsersController.new を呼び出す.

#### 【補足】

- ・このアクションは new ページにて「登録」が押された際に呼び出される.
- ・「ログイン処理」は SessionsController.create を参照.
- ・user\_params メソッドは private である. new での入力内容を取得する.
- ・flashとは、リダイレクト時に、一度だけページ上にメッセージを表示させるメソッドである.
- ・saveとは、Railsに標準搭載された、DBにインスタンスを追加するメソッドである.

#### 入力值説明

:name:ユーザーネーム,:email:メールアドレス,

:pasword:パスワード, :password\_confirmation:パスワード再確認

#### 出力值説明

#### 他クラス・関数との関係

ApplicationController クラスを継承する.

user\_params メソッドを使用する.

図 4.17: UsersController.create 定義書

所属クラス	UsrsController	作成日	2023/12/06
機能名	ユーザ新規登録	更新日	2023/12/08
モジュールID	UsersController.create	作成者	三上柊
使用モジュールID	UsersController.new, UsersContoroller.home		

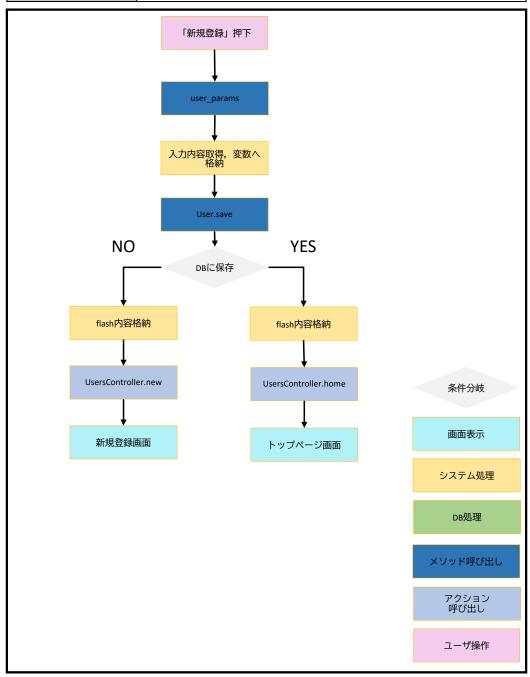


図 4.18: UsersController.create フロー図

モジュール	·定義 				
	システム名	楽譜管理ソフトウェア	バージョン		
管理情報	工程名	内部設計	作成日	2023/12/4	
	作成者	山本祥弘	更新日	2023/12/8	
	概要	ScoresController 内の destr	oy アクション		
基本情報	所属クラス	ScoresController			
空午旧刊	モジュール名	destroy			
	モジュールID	ScoresController.destroy			
処理説明					
【処理内容	7				
楽譜削除ホ	ドタンが押されたときに	「呼び出す.楽譜データを削除	する.		
【処理手順	Ī				
1. current	_user.id が管理者もし	くはログインしたユーザである	るかを判定する.		
2. current	_user.id がログインし	たユーザである場合,ユーザ削	削除確認ボックスを表示	する.	
current_us	current_user.id が管理者であった場合,パスワード認証を含むユーザ削除ボックスを表示する.				
3. 「戻る」ボタンが押されると UsersController.home を呼び出し処理を終了する.「削除」ボタンが押さ					
れると、 Score.find(id).destroy を実行し、楽譜データを削除する。楽譜データを削除する際,					
current_us	current_user.id が管理者である場合にはパスワード認証を通過している必要がある.				
4. UsersController.home を呼び出し,削除完了の flash を表示する.					
【補足】					

### 入力值説明

:score\_id : スコアID,:user\_id : ユーザID

## 出力值説明

なし

#### 他クラス・関数との関係

ApplicationController クラスを継承する.

所属クラス	ScoresController	作成日	2023/12/07
機能名	楽譜データ削除	更新日	2023/12/08
モジュールID	ScoresController.destroy	作成者	山本祥弘
使用モジュールID	UsersController.home		

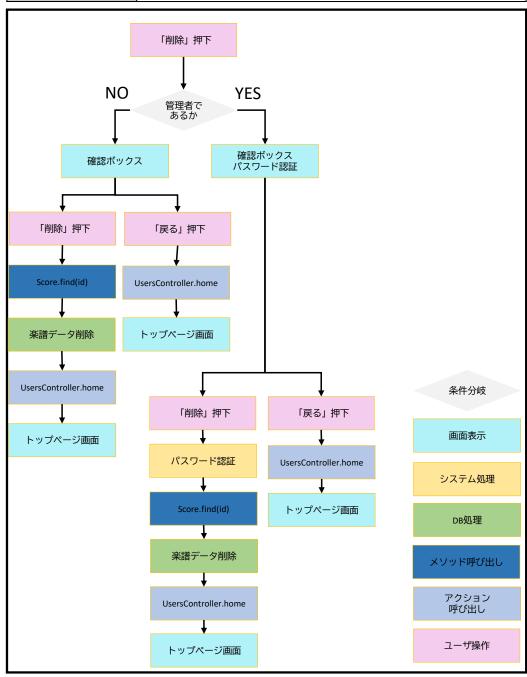


図 4.20: ScoresController.destroy フロー図

モジュール	モジュール定義				
	システム名	楽譜管理ソフトウェア	バージョン		
管理情報	工程名	内部設計	作成日	2023/12/4	
	作成者	山本祥弘	更新日	2023/12/5	
	概要	UsersController 内の destroy アクション			
基本情報	所属クラス	UsersController			
至午旧刊	モジュール名	destroy			
	モジュールID	UsersController.destroy		_	

#### 【処理内容】

ユーザ削除ボタンが押されたときに呼び出す.データベースに登録されているユーザを削除する.

#### 【処理手順】

- 1. current\_user.id が管理者もしくはログインしたユーザであるかを判定する.
- 2. current\_user.id がログインしたユーザである場合,ユーザ削除確認ボックスを表示して手順3-1に移行する.current\_user.id が管理者であった場合,パスワード認証を含むユーザ削除ボックスを表示して手順4-1に移行する.
- 3-1. 「戻る」ボタンが押されると UsersController.show を呼び出し処理を終了する.「削除」ボタンが押されると, current\_user.id と削除するユーザが同じであるかを判定し,同じである場合のみ User.find(id).destroy を実行してユーザを削除する.
- 3-2. UsersController.new を呼び出し、削除完了の flash を表示して処理を終了する.
- 4-1. 「戻る」ボタンが押されると UsersController.index を呼び出し処理を終了する.パスワード認証を通過し、かつ「削除」ボタンが押されると、 User.find(id).destroy を実行し、ユーザを削除する.
- 4-2. UsersController.index を呼び出し、削除完了の flash を表示する.

#### 【補足】

命令 has\_many :scores, dependent: :destroy により、削除されるユーザに関連づけられた楽譜データも同時に削除される

時に削除される.
1 +/ <del>///                                   </del>
入力值説明
:user_id : ユーザID
出力值説明
なし
他クラス・関数との関係

#### 他クラス・関数との関係

ApplicationController クラスを継承する.

所属クラス	UsersController	作成日	2023/12/07
機能名	ユーザ削除	更新日	2023/12/07
モジュールID	UsersController.destroy	作成者	山本祥弘
使用モジュールID	UsersController.show, UsersController.new, UsersController.index		

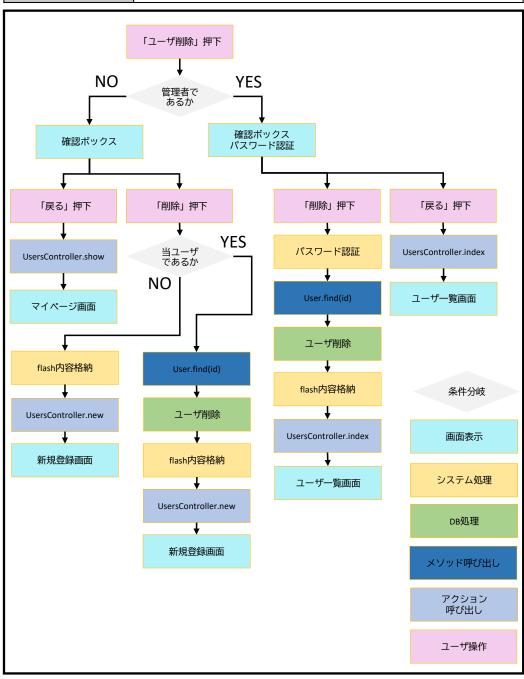


図 4.22: UsersController.destroy フロー図

レンユーハ	/ 上我						
	システム名	楽譜管理ソフトウェア	バージョン				
管理情報	工程名	内部設計	作成日	2023/12/05			
	作成者	溝口洸熙	更新日	2023/12/05			
	概要	SessionsController 内の new メソッド					
基本情報	所属クラス	SessionsController					
<b>全</b> 个旧拟	モジュール名	new					
	モジュールID	SessionsController.new					
処理説明							
【処理内容	\$]						
セッション	ンを作成する画面(ロ <sub>:</sub>	グイン画面)を表示するアクショ	ン.				
【処理手順	頁】						
1. views/s	essions/new.html.er	b を描画する.					
【補足】							
入力值説明							
なし							
出力値説明	<u> </u>						
なし							
なし							
他クラス・	関数との関係						
Appliant's	nControllor t 如云十	7					
Applicatio	nController を継承する	ଚ.					
1							

図 4.23: SessionsController.new 定義書

所属クラス	SessionsController	作成日	2023/12/08
アクション名	new	更新日	2023/12/08
モジュールID	SessionsController.new	作成者	田中諒
使用モジュールID			

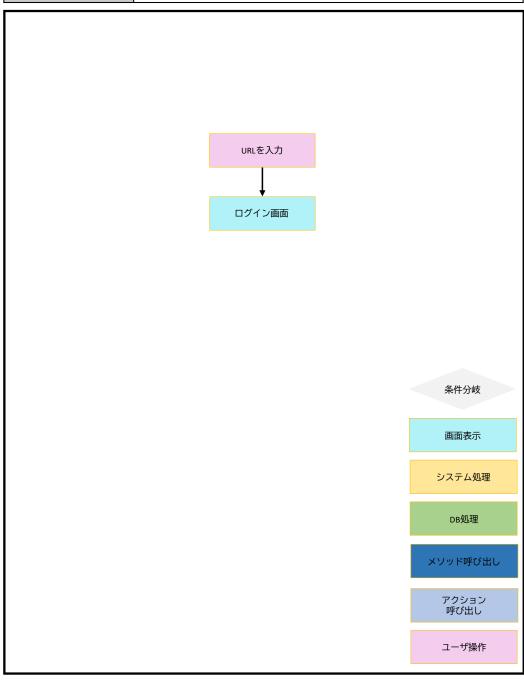


図 4.24: SessionsController.new フロー図

モジュール定義						
	システム名	楽譜管理ソフトウェア	バージョン			
管理情報	工程名	内部設計	作成日	2023/12/04		
	作成者	溝口洸熙	更新日	2023/12/04		
	概要	SessionsController 内の create メソッド				
基本情報	所属クラス	SessionsController				
坐个旧拟	モジュール名	create				
	モジュールID	SessionsController.create				

#### 【処理内容】

セッションを作成する画面(ログイン画面)を処理する.

#### 【処理手順】

- 1. User クラスの find\_by 関数を用いて,入力にある :email と一致するユーザインスタンスを変数名 user に格納する.
- 2. 取得した:password を用いて認証する.
- 3. 認証に通過した場合,ログイン処理をする.ログイン処理は,session の :user\_id に, user\_id を 渡す処理をする.ここで,ログイン時刻を users テーブルに格納する.
- 4. 認証に失敗した場合は,HTTPステータスコードを「422 Unprocessable Entity」として応答し,認証失敗のflashを表示し,SessionsController.new を呼び出す.

#### 【補足】

#### 入力值説明

:email: ログイン情報のEメールのアドレス :password: ログイン情報のパスワード

#### 出力值説明

なし

#### 他クラス・関数との関係

ApplicationController を継承する.

図 4.25: SessionsController.create 定義書

所属クラス	SessionsController	作成日	2023/12/06
アクション名	create	更新日	2023/12/08
モジュールID	SessionsController.create	作成者	三上柊
使用モジュールID	UsersController.home		

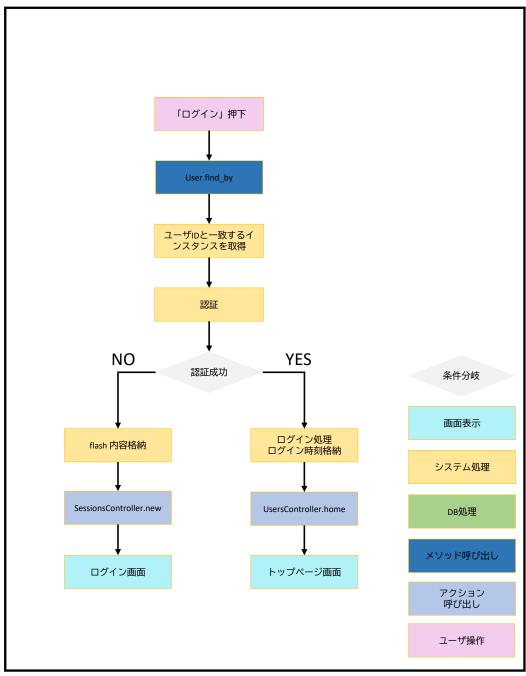


図 4.26: SessionsController.create フロー図

モジュール	レ定義							
	システム名	楽譜管理ソフトウェア	バージョン					
管理情報	工程名	内部設計	作成日	2023/12/05				
	作成者	溝口洸熙	更新日	2023/12/05				
	概要	SessionsController 内の d		2020, 12, 00				
	所属クラス SessionsController							
基本情報	モジュール名	destroy						
	モジュールID	SessionsController.destro	)V					
			,					
【処理内容	· · · · · · · · · · · · · · · · · · ·							
2. current <sub>_</sub> 3. Sessior 【補足】	ウト処理をする。 re _user に nil を代入す nsController.new を		ails標準のものである.					
入力值説明	月							
なし								
出力值説明	月							
なし								
他クラス・	・関数との関係							
Applicatio	nController を継承す							

図 4.27: SessionsController.destroy 定義書

所属クラス	SessionsController	作成日	2023/12/06
アクション名	destroy	更新日	2023/12/08
モジュールID	SessionsController.destroy	作成者	田中諒
使用モジュールID	ScoresController.new		

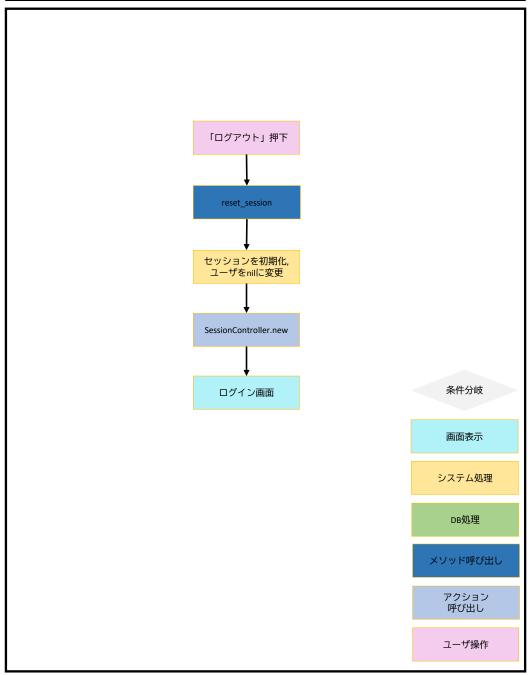


図 4.28: SessionsController.destroy フロー図

# 第5章

DB

DB 構造を以下に示す.

int user_id PK ユーザID text email メールアドレス text name 組織名など text password パスワード boolean admin 管理者であるか date login 最終ログイン日 timestamp created_at 作成日時 timestamp updated_at 更新日時  1つのユーザはO以上のスコアを持つ  ****  *** *** ** ** ** ** ** ** ** **		User						
text name 組織名など text password パスワード boolean admin 管理者であるか date login 最終ログイン日 timestamp created_at 作成日時 timestamp updated_at 更新日時  1つのユーザはO以上のスコアを持つ  ****  ****  ***  ***  ***  ***  **		int		user_id	PK	ユーザ		fID
text password パスワード boolean admin 管理者であるか date login 最終ログイン日 timestamp created_at 作成日時 timestamp updated_at 更新日時 コラウスコアを持つ 中ででででででででででででででででいます。 ロード は score_id FK 参照(ユーザ) を		text		email		メールアドレス		アドレス
boolean   admin   管理者であるか   date   login   最終ログイン日   timestamp   created_at   作成日時   timestamp   updated_at   更新日時   ToのユーザはO以上のスコアを持つ   ToのユーザはO以上のスコアを持つ   Teferences   user_id   FK 参照(ユーザ)   text   name   楽曲名   text   composer   作曲者名   text   arranger   編曲者名   text   grade   グレード   int   piccolo   int   c_flute   int   oboe   int   english_horn   int   e_clarinet   int   b_bass_clarinet   int   b_bass_clarinet   int   b_tenor_saxophone   int   b_trumpet   int   b_trumpet   int   f_horn   int   trombone   int   tuba   int   string_bass   int   piano   int   timpani   int   drums   int   percussion   timestamp   created_at   作成日時		text		name		組	織名	aなど
date   login		text		password		バ	゚スワ	7 — ド
timestamp created_at 作成日時 timestamp updated_at 更新日時  1つのユーザはO以上のスコアを持つ  Score  int score_id PK スコアID references user_id FK 参照(ユーザ) text name 楽曲名 text composer 作曲者名 text grade グレード int time 演奏時間(秒) int c_flute int oboe int english_horn int e_clarinet int b_bass_clarinet int b_bass_clarinet int b_bass_clarinet int b_bass_clarinet int b_bass_clarinet int b_baritone_saxophone int b_trumpet int f_horn int trombone int tuba int string_bass int piano int drums int drums int percussion timestamp created_at 作成日時		boolean		admin		管	理者	であるか
Timestamp   updated_at   更新日時   Updated_at   更新日時   Updated_at   更新日時   Updated_at   Updated		date		login		最	終口	1グイン日
Score int score_id PK スコアID references user_id FK 参照(ユーザ) text name 楽曲名 text composer 作曲者名 text grade グレード int piccolo int c_flute int oboe int english_horn int e_clarinet int b_bass_clarinet int b_bass_clarinet int b_bass_clarinet int b_bass_clarinet int b_trumpet int b_trumpet int trombone int tuba int string_bass int piano int harp int drums int drums int percussion time perference, idea PK スコアID を		timesta	mp	created_at		作	成日	]時
Score  int score_id PK スコアID references user_id FK 参照(ユーザ) text name 楽曲名 text composer 作曲者名 text grade グレード int time 演奏時間(秒) int piccolo int c_flute int oboe int english_horn int e_clarinet int b_bass_clarinet int b_bass_clarinet int b_to b_ton_saxophone int b_trumpet int b_trumpet int trombone int tuba int string_bass int piano int harp int drums int drums int percussion int percussion itext PK スコアID を察院(ユーザ) を認定している。 を		timesta	mp	updated_at		更	新日	]時
int score_id PK スコアID references user_id FK 参照(ユーザ) text name		100	カユ			スコ	ア	を持つ
references user_id FK 参照(ユーザ) text name 楽曲名 text composer 信曲者名 text grade グレード int time 演奏時間(秒) int piccolo int c_flute int oboe int english_horn int e_clarinet int b_bass_clarinet int b_bass_clarinet int b_tompet int b_trumpet int b_trumpet int trombone int tuba int string_bass int piano int drums int drums int drums int drums int drums int preded, after in the fixed between the fixed					e	_		
text name 楽曲名 text composer 作曲者名 text arranger 編曲者名 text grade グレード int time 演奏時間(秒) int piccolo int c_flute int oboe int english_horn int e_clarinet int b_bass_clarinet int b_bass_clarinet int b_to bass_clarinet int b_to b_ton_saxophone int b_ton_saxophone int b_ton jaxophone int b_trumpet int trombone int trombone int tuba int string_bass int piano int drums int drums int drums int percussion timestamp created_at	_		_			_		
text composer 作曲者名 text arranger 編曲者名 text grade グレード int time 演奏時間(秒) int piccolo int c_flute int oboe int english_horn int e_clarinet int b_bass_clarinet int b_sasson int e_alto_saxophone int b_trumpet int b_trumpet int trombone int tuba int string_bass int piano int harp int drums int praceded  #################################	_		_			4	FK	
text arranger 編曲者名 text grade グレード int time 演奏時間(秒) int piccolo int c_flute int oboe int english_horn int e_clarinet int b_bass_clarinet int b_sassoon int e_alto_saxophone int b_trumpet int b_trumpet int trombone int tuba int string_bass int piano int harp int drums int drums int drums int drums int piccolo int drums int drums int drums int drums int piccolo int drums int piccolo int drums int piccolo int drums int precussion int time int int precussion int time int precussion int time int piccolo int piccolo int piccolo int drums int precussion int friccolo int piccolo int piccolo int drums int precussion int friccolo int piccolo int piccolo int piccolo int piccolo int friccolo int	tex	t	nar	ne		4		
text grade グレード int time 演奏時間(秒) int piccolo int c_flute int oboe int english_horn int e_clarinet int b_bass_clarinet int b_bass_clarinet int b_tenor_saxophone int b_tenor_saxophone int b_trumpet int trombone int trombone int tuba int string_bass int piano int drums int drums int drums int drums int drums int druschild int drums int drums int piccolo int drucklession int drums int precussion int time precussion int piacolo int precussion						4		
int time 演奏時間(秒) int piccolo int c_flute int oboe int english_horn int e_clarinet int b_clarinet int b_bass_clarinet int bassoon int e_alto_sxophone int b_tenor_saxophone int b_trumpet int f_horn int trombone int baritone int tuba int string_bass int piano int drums int drums int drums int drums int drussion int percussion int piccolo   int piccolo   int percussion int piccolo   int piccolo	-		_			4		
int piccolo int c_flute int oboe int english_horn int e_clarinet int b_clarinet int b_bass_clarinet int bassoon int e_alto_saxophone int b_trumpet int b_trumpet int trombone int tuba int string_bass int piano int drums int drums int drums int drums int drusselint int drusselint int drusselint int drums int drums int percussion int piano int drums int percussion timestamp created_at #frombone	-		-			4		
int c_flute int oboe int english_horn int e_clarinet int b_clarinet int b_bass_clarinet int bassoon int e_alto_saxophone int b_tenor_saxophone int b_trumpet int f_horn int trombone int baritone int tuba int string_bass int piano int drums int drums int drums int drums int drums int drums int percussion timestamp reated_at #fxxl=fxxl=fxxl=fxxl=fxxl=fxxl=fxxl=fxxl						_		演奏時間(秒)
int oboe int english_horn int e_clarinet int b_clarinet int b_bass_clarinet int bassoon int e_alto_saxophone int b_tenor_saxophone int b_trumpet int f_horn int trombone int baritone int tuba int string_bass int piano int harp int drums int drums int percussion time string_bass int piano int harp int drums int drums int percussion timestamp created_at 作成日時						4		
int english_horn int e_clarinet int b_clarinet int b_bass_clarinet int bassoon int e_alto_saxophone int b_tenor_saxophone int b_trumpet int f_horn int trombone int baritone int tuba int string_bass int piano int harp int drums int drums int drums int percussion timestamp reated_at 作成日時	_		_			4		
int e_clarinet int b_clarinet int b_bass_clarinet int bassoon int e_alto_saxophone int b_tenor_saxophone int b_trumpet int f_horn int trombone int baritone int tuba int string_bass int piano int harp int timpani int drums int drums int drums int percussion timestamp created_at 作成日時						4		
int b_clarinet int b_bass_clarinet int bassoon int e_alto_saxophone int b_tenor_saxophone int b_trumpet int f_horn int trombone int baritone int tuba int string_bass int piano int harp int timpani int drums int percussion ttimestamp created_at 作成日時	_		<u> </u>			+		
int b_bass_clarinet int bassoon int e_alto_saxophone int b_tenor_saxophone int b_baritone_saxophone int f_horn int trombone int baritone int tuba int string_bass int piano int harp int timpani int drums int percussion ttimestamp created_at 作成日時	-					+		
int bassoon int e_alto_saxophone int b_tenor_saxophone int b_baritone_saxophone int b_trumpet int f_horn int trombone int baritone int tuba int string_bass int piano int harp int timpani int drums int drums int percussion ttimestamp created_at 作成日時						+		
int e_alto_saxophone int b_tenor_saxophone int b_baritone_saxophone int b_trumpet int f_horn int trombone int baritone int tuba int string_bass int piano int harp int timpani int drums int percussion ttimestamp created_at 作成日時	_					+		
int b_tenor_saxophone int b_baritone_saxophone int b_trumpet int f_horn int trombone int baritone int string_bass int piano int timpani int drums int drums int percussion ttimestamp created_at 作成日時						+		
int b_baritone_saxophone int b_trumpet int f_horn int trombone int baritone int string_bass int piano int timpani int drums int drums int percussion ttimestamp reated_at 作成日時						+		
int b_trumpet int f_horn int trombone int baritone int tuba int string_bass int piano int harp int timpani int drums int percussion ttimestamp created_at 作成日時						0		
int f_horn int trombone int baritone int tuba int string_bass int piano int timpani int drums int drums int percussion timestamp created_at 作成日時								
int trombone int baritone int tuba int string_bass int piano int harp int timpani int drums int percussion timestamp created_at 作成日時	_		-			+		
int baritone int tuba int string_bass int piano int harp int timpani int drums int percussion timestamp created_at 作成日時	-	-						
int tuba int string_bass int piano int harp int timpani int drums int percussion timestamp created_at 作成日時								
int piano int harp int timpani int drums int percussion timestamp created_at 作成日時	int	int						
int piano int harp int timpani int drums int percussion timestamp created_at 作成日時	int	int :		string_bass		+		
int timpani int drums int percussion timestamp created_at 作成日時	int	int pi		piano		T		
int drums int percussion timestamp created_at 作成日時	int	int ha		harp		1		
int percussion timestamp created_at 作成日時	int	int t		timpani		1		
timestamp created_at 作成日時	int	int						
· -	int	int		percussion				
timestamp updated_at 更新日時	tim	timestamp		reated_at				作成日時
	tim	timestamp		odated_at				更新日時

図 5.1: ER 図

# 第6章

# 貢献内容

システム提案書第2版の作成における、各メンバの貢献内容は以下の通り.

学籍番号	氏名	貢献内容・担当箇所		
1250297	奥平 舜理	モジュールの内容,システム実装方法		
		モジュール担当箇所:UC.index, UC.home		
1250341	田中 諒	モジュールの内容、テンプレート作成		
1200041		モジュール担当箇所:UC.update, SC.update		
1250352	中村 祐貴	モジュールの作成,規約(T <sub>E</sub> X)		
1200002		モジュール担当箇所:UC.show, SC.show		
1250272	三上 柊	モジュールの内容,テンプレート作成・更新,進捗管理,TEX 清書		
1200312		モジュール担当箇所:UC.create, SC.create		
1250373	溝口 洸熙	モジュールの内容,規約		
1200010		モジュール担当箇所:SsC.create, SsC.new, Ssc.destroy		
1250382	山田 滉希	モジュールの内容		
1200002		モジュール担当箇所:UC.edit, SC.edit		
1250385	山本 祥弘	モジュールの内容,テンプレート更新		
		モジュール担当箇所:UC.destroy, SC.destroy, UC.destroy, SC.destroy		

UC: UsersControllerSC: ScoresControllerSsC: SessionsController