

# 如何快速开发一款小游戏（一）

By 毅然

# 目录

- 目标
- 游戏逻辑
- 数据库设计
- 代码开发
  - 安装 LeanCloud
  - 登录
  - 单人
  - 挑战

# 目标

GitHub: <https://github.com/leancloud/answer-game>

LeanCloud

登录

单人

挑战

> 返回

0

甲壳虫乐队共有几个人组成?

90

3

0

4

5

6

8s

# 游戏逻辑

- 可选模式

- 单人：从题库中随机抽取 3 道题，玩家答完后记录结果供他人挑战。
- 挑战：从所有玩家的单人历史闯关中随机抽取一条记录供当前玩家挑战。这条记录包括历史题目及顺序、玩家选择及得分。

- 得分逻辑

- 玩家选对得分，选错不得分
- 得分计算方式：每一题分数 \*  $(\text{答题剩余时间} / \text{答题要求时间 } 10 \text{ 秒})$

# 数据库设计

# 数据库设计

- 登录：需要用户表记录用户信息
- 单人：单人题库表，题目从题库表中抽取。
- 挑战：单人模式结束后，会将单人模式答过的题及答题顺序记录下来供他人挑战。
  - 挑战题库表：存储可供他人挑战的题目。
  - 挑战分数表：存储挑战的分数。

# 用户表

- 使用 LeanCloud 内置 \_User 表。

字段名	字段类型	描述
username	String	用户名
authData	Object	第三方平台的登录信息

# 题库表

- Question 表

字段名	字段类型	描述
title	String	问题的标题
options	Array	问题的所有选项
answerIndex	Number	正确答案在 options 中的索引位置
qid	Number	自增 ID，用于随机获取数据



# 挑战题库表

- ChallengeQuestion 表

字段名	字段类型	描述
questions	Array	单人模式中答过的 3 道题
creator	Pointer	产生这条记录的玩家, 指向 _User 表

# 挑战分数表

- ChallengeScore 表

字段名	字段类型	描述
challenge	Pointer	指向 ChallengeQuestion 表
user	Pointer	产生这条记录的玩家，指向 _User 表
userOptions	Array	单人闯关中玩家选择的 3 个答案位置
userOptionScores	Array	单人闯关中每道题的得分
cid	Number	自增 ID，用于随机获取数据

# 数据库设计

## 请注意

- Array 如果超过了 50 个对象，建议换用其他的设计方式。
- <https://leancloud.cn/docs/relation-guide.html>

代码开发

# 代码开发

## 客户端

- 游戏开发引擎：CocosCreator
- 语言：JavaScript

## 服务端

- 数据库：LeanCloud 数据存储
- 代码部署：LeanCloud 云引擎
- 语言：Node.js

# 安装 LeanCloud

<https://leancloud.cn/docs/start.html>

# 登录

- 匿名登录
- 微信小程序登录
- 第三方账号登录

# 匿名登录

```
User.loginWithAuthData({
    id: uuid
}, 'anonymous')
.then(function (currentUser) {
    // 将匿名用户的随机字符串用户名修改为可读的用户名
    var Random = Mock.Random;
    var username = Random.cname();
    currentUser.setUsername(username);
    return currentUser.save();
})
.then(function () {
    cc.director.loadScene('menu');
})
.catch(console.error);
```



# 游戏

- 准备题库
- 获取数据
- 界面展示

# 单人 — 获取数据

- 流程

- 客户端调用云函数
- 云函数访问数据库获取题目
- 返回题目给客户端

- 随机逻辑

- 获得表中最大的 qid。
- 在 1 和最大的 qid 之间取 3 个不重复的数字
- 获取 qid 是这三个数字的数据

# 单人 — 云函数

```
AV.Cloud.define('getSingleGameData',  
function(request) {  
  var qidQuery = new AV.Query('Question');  
  // 取随机数的边界  
  qidQuery.descending('qid');  
  qidQuery.limit(1);  
  return qidQuery.first().then((question) => {  
    var qid = question.get('qid');  
    var randomNumbers = getRandomNumbers(qid, 3);  
    var questionQuery = new AV.Query('Question');  
    questionQuery.containedIn('qid', randomNumbers);  
    return questionQuery.find();  
  });  
});
```

# 单人—客户端调用

```
AV.Cloud.rpc('getSingleGameData')
  .then(function(questions) {
    // [AVObject, ...]
    self.questions = questions;
    cc.director.loadScene('single-game-play');
  })
  .catch(function(err) {
    console.error(err);
  });
```

# 单人 — 存储结果

```
// 存储挑战题库
var ChallengeQuestion = AV.Object.extend('ChallengeQuestion');
var challengeQuestion = new ChallengeQuestion();
challengeQuestion.set('questions', this.questions);
var currentUser = AV.User.current();
challengeQuestion.set('creator', currentUser);
return challengeQuestion.save().then((challengeQuestion) => {
  // 存储当前用户得分
  var ChallengeScore = AV.Object.extend('ChallengeScore');
  var challengeScore = new ChallengeScore();
  // 存储题目
  challengeScore.set('challenge', challengeQuestion);
  // 是谁答题
  challengeScore.set('user', currentUser);
  // 选了哪些答案
  challengeScore.set('userOptions', this.userOptions);
  // 每个答案多少分
  challengeScore.set('userOptionScores', this.userOptionScores);
  // 保存
  return challengeScore.save();
});
```

# 挑战 — 云函数

```
AV.Cloud.define('getChallengeGameData', function(request) {  
  var cidQuery = new AV.Query('ChallengeScore');  
  // 取随机数的边界  
  cidQuery.descending('cid');  
  cidQuery.limit(1);  
  return cidQuery.first().then((challengeScore) => {  
    var cid = challengeScore.get('cid');  
    var randomNumbers = getRandomNumbers(cid, 1);  
    cid = randomNumbers[0];  
    console.log('随机取到的 1 个 cid 为 ' + cid);  
    var challengeQuery = new AV.Query('ChallengeScore');  
    challengeQuery.equalTo('cid', cid);  
    challengeQuery.include(['user', 'challenge.questions']);  
    challengeQuery.select(['user.username', 'challenge',  
      'userOptionScores', 'userOptions']);  
    return challengeQuery.first();  
  });  
});
```

# 挑战 — 客户端调用

```
AV.Cloud.rpc('getChallengeGameData')
  .then(function(challenge) {
    self.questions =
challenge.get('challenge').get('questions');
    self.rivalOptions =
challenge.get('userOptions');
    self.rivalOptionScores =
challenge.get('userOptionScores');
    self.rivalUsername =
challenge.get('user').get('username');
    return challenge;
  })
```

# 挑战 — 获取数据

## 请注意

- 如果 Array 中超过了 10 个 Pointer 对象，不要直接 Include，建议分批 Fetch 数据。



# 自己尝试一下

- 微信小游戏登录
- 定期清理数据

# 深入知识点总结

- 数据模型设计
- 数据安全
- 索引

# 预告

- 二期课程： 排行榜
- 三期课程： 多人实时在线对战