摘要

当今时代，互联网无处不在，它极大的便利了人们的日常生活和工作效率，为人类社会的发展做出了巨大贡献。互联网技术以其高效办公效率和精密计算，在银行等财汇业务领域占据着重要地位，没有哪一家银行可以脱离它而运行。

基于传统的web2.0技术所开发的第一代银行管理系统，已经不能满足用户日益增长的业务需要和银行系统不断迭代升级的需求，为此基于web3.0技术的飞速发展和浏览器、服务器等软件和硬件的不断优化升级的形势下，开发一套第二代银行系统，无论对于浏览器端客户的体验感，还是对于管理银行的办公效率提升、业务扩充、系统优化升级等方面，都意义重大。

为了实现本系统，调研了当今世界一些比较成熟的技术与解决方案，决定采用浏览器-服务器-数据库的架构模式；浏览器端采用了MVVM框架下主流的Vue技术，其性能优势尤为突出，被广泛使用在当今的各大型网站开发中；服务器端采用Node.js服务器，其单线程的异步回调模式、IO之间的高效切换能力的性能尤为优越，以及可以直接在本地（localhost）运行系统，更是方便了开发和减少了网络传输的时延；数据库采用MYSQL数据库，其体积小、速度快、成本低，开源等优势及大了方便了开发；这些技术都将是实现本系统的核心技术，他们将贯穿整个系统的开发过程。

通过开发本系统，极大的锻炼了自己抽象思维分析和解决众多业务需求问题、阅读国内外文献（以技术官网为主）、运用专业手段及科学方法获取相关信息的能力和文字表达能力等。对自己综合部分大学所学知识和技术，是一个很好的展现。

关键词：银行系统；Vue；Node.js；web开发；

ABSTRUCT

目录

1绪论

本章是第一张，主要对本系统的研究背景及问题、研究内容和技术路线进行说明。首先指出过去使用传统技术开发的网站在当今社会中出现的问题，以及提出新技术的优势与应用在系统中的价值和意义所在，然后对本系统的业务逻辑和功能做一个具体的阐述，让读者在一开始就对本系统有一个清晰直观的认知。

1.1研究背景

随着计算机技术和Internet技术的飞速发展，Web发展越来越快，应用范围越来越广，已经深入到各行各业中。众多知名的银行（如中国银行、中国建设银行等）由于当时JavaScript技术发展受限而使用传统的Jquery技术开发的网站，已经在当今的运行和维护中出现了许许多多的问题（如Jquery直接操做原生DOM元素、不断跳转刷新页面使得浏览器不断地请求和获取服务器等对浏览器性能有较大的消耗），众所周知，一个系统网站良好运行是离不开长期的维护的，而使用落后的技术将极大增加维护的难度和成本，以及不便于业务的扩充和系统的升级；此类问题举不胜数。

随着2015年ES6模块化标准由ECMA-262（欧洲计算机制造商协会中专门制定ECMAScript标准规范的一个分支机构）发布，使得JavaScript快速发展起来，很多开发者、技术厂商早已预见到JavaScript的无穷潜力，一些解决复杂问题的方案技术如Vue、React等MVVM前端框架技术层出不穷、开发大型应用所使用的各种第三方库（npm、yarn）和构建工具（webpack、gulp）、各种后端框架（Express、Koa）等使得JavaScript飞速发展和广泛应用进入有史以来从未有过的繁盛阶段，它们使浏览器运行更加高效快速，将用户体验感提升到了一个新的水平（如无刷新地跳转、单页面应用极大地降低了服务器端渲染压力），这些为本系统的成功实现奠定了坚实地技术基础。

1.2本文工作

本系统将综合自己在校期间所学知识和技术，完成一个从需求分析、UI 设计到开发、测试的全栈项目；实现一个银行系统的基本功能：用户（存款用户、取款用户、系统普通管理员、银行高级管理员）登录、注册、存取款、贷还款、抽奖、修改部分个人信息，管理员管理用户（例如设置用户的权限（冻结用户、设置大额贷款限制））、管理银行（修改删除用户信息、增改利率、设置银行本金等操做）；通过本系统，将极大的锻炼自己技术水平和解决问题的能力，对技术在业务中的应用有一个深层次的认知和理解。

2 相关技术、开发工具

本章将介绍一些项目开发过程中使用到的目前主流技术和相关开发工具。前端（浏览器端）主要使用基于Javascript语言的Vue技术，服务器端使用Node.js，数据库采用MYSQL。开发工具使用微软的VScode，数据库工具DBeaver。

2.1 浏览器端技术

HTML

Hyper Text Markup Language，超文本标记语言

它包括一系列标签．通过这些标签可以将网络上的文档格式统一，使分散的Internet资源连接为一个逻辑整体。HTML文本是由HTML命令组成的描述性文本，HTML命令可以说明文字，图形、动画、声音、表格、链接等。(摘自：百度百科)

CSS

Cascading Style Sheets，层叠样式表

CSS为HTML标记语言提供了一种样式描述，定义了html元素的显示方式和效果。

主要特点有：丰富的样式定义、易于使用和修改、多页面应用、层叠、不区分大小写。

Less

官网：https://lesscss.org/

Less 是一门 CSS 预处理语言，它扩展了 CSS 语言，增加了变量、Mixin、函数等特性，使 CSS 更易维护和扩展。

JavaScript

JavaScript是一种编译型的脚本语言，被广泛使用在浏览器和非浏览器的(Node.js服务器等)，支持面向对象、命令式、声明式、函数式编程范式。

Ajax

Asynchronous Javascript And XML

Ajax 在浏览器与 Web 服务器之间使用异步数据传输（HTTP 请求），这样就可使网页从服务器请求少量的信息，而不是整个页面。Ajax可使因特网应用程序更小、更快，更友好。

在 JavaScript 语言刚刚诞生的时候，它仅仅用于实现页面中的一些小效果，那个时候，一个页面所用到的 JS 可能只有区区几百行的代码。在这种情况下，语言本身所存在的一些缺陷往往被大家有意的忽略，因为程序的规模实在太小，只要开发人员小心谨慎，往往不会造成什么问题。

直到1998年ajax的出现，逐渐改变了 JavaScript 在浏览器中扮演的角色。现在，它不仅可以实现小的效果，还可以和服务器之间进行交互，以更好的体验来改变数据，JS代码的数量开始逐渐增长，从最初的几百行，到后来的几万行，前端程序逐渐变得复杂，传统的技术已不足以应对这些复杂的场景，随之出现了一些列解决方案和新技术的出现。

下面就介绍一下这些新的技术和解决方案：

**npm**

官网：https://www.npmjs.com/

npm 全称为 node package manager，即 node 包管理器，它运行在 node(服务器端) 环境中，让开发者可以用简单的方式完成包的查找、安装、更新、卸载、上传等操作。

**webpack**

官网：<https://webpack.js.org/>

随着**开发时态**(项目开发的过程，主要侧重于：模块划分更细、支持多种模块化标准、支持npm或其他包管理器下载的模块、能够解决其他工程化的问题)和**运行时态**(项目代码上线运行时，主要侧重于：文件越少越有利、文件体积越小越有利代码内容越乱越有利、所有浏览器都要兼容、能够解决其他运行时的问题，主要是执行效率问题)两者之间的巨大差异不断加大,我们需要有一个工具，这个工具能够让开发者专心的在开发时态写代码，然后利用这个工具将开发时态编写的代码转换为运行时态需要的东西，这样的工具，叫做**构建工具**。

webpack是基于模块化的打包(构建)工具，它把一切视为**模块**，它通过一个开发时态的入口模块为起点，分析出所有的依赖关系，然后经过一系列的过程（压缩、合并），最终生成运行时态的文件。

Vue

官网：https://vuejs.org/

Vue 是一套用于构建用户界面的渐进式JavaScript框架，它将以上所述的各种技术集成在了一起，便于开发者使用，使其开发更加高效和便捷，编写出的代码可以实现可维护、可复用、可扩展等。

Vue主要有以下的几点重大创新：

1. 引入vue-cli脚手架工具（https://cli.vuejs.org/),使得开发更为便捷和高效，
2. 组件化开发：降低整体复杂度，提升代码的可读性和可维护性，提升局部代码的可复用性。
3. 单页面应用：有良好的交互体验、前后端分离、减轻服务器的压力、公用一套后端程序代码。
4. 虚拟DOM：由于真实JS操做DOM时：创建、更新、插入等操作会带来大量的浏览器性能性能损耗，从而就会极大的降低渲染效率。因此，vue在渲染时，使用虚拟DOM来替代真实DOM，主要解决渲染效率的问题。
5. 数据共享技术 vuex（https://vuex.vuejs.org/），该技术主要解决了几个难题：如何保证数据的唯一性、某个组件改动数据后，如何让其他用到该数据的组件知道数据变化了两个问题，为此我们可以设置一个独立的数据仓库，数据只有一份且是响应式的，这样上面的两个问题就迎刃而解了。
6. 路由技术vue-router（https://router.vuejs.org/），该技术主要解决了：根据地址中的路径选择不同的组件、把选择的组件放到相应的位置、无刷新的切换组件三个问题，实现了良好的路由跳转。

由于该技术性能优越，被广泛的使用在当今web开发领域中，本系统也是采用了该技术。

2.2 服务器端技术

Node.js

官网：<https://nodejs.org/en/>

Node.js 是一个开源、跨平台、基于 Chrome V8 引擎的 JavaScript 运行时环境。

Node.js 应用程序运行于单个进程中，无需为每个请求创建新的线程。 Node.js 在其标准库中提供了一组异步的 I/O 原生功能（用以防止 JavaScript 代码被阻塞），并且 Node.js 中的库通常是使用非阻塞的范式编写的（从而使阻塞行为成为例外而不是规范）。

当 Node.js 执行 I/O 操作时（例如从网络读取、访问数据库或文件系统），Node.js 会在响应返回时恢复操作，而不是阻塞线程并浪费 CPU 循环等待。

这使 Node.js 可以在一台服务器上处理数千个并发连接，而无需引入管理线程并发的负担（这可能是重大 bug 的来源）。

Node.js 具有独特的优势，因为为浏览器编写 JavaScript 的数百万前端开发者现在除了客户端代码之外还可以编写服务器端代码，而无需学习完全不同的语言。

在 Node.js 中，可以 毫无问题地使用新的 ECMAScript 标准 ，因为不必等待所有用户更新其浏览器，你可以通过更改 Node.js 版本来决定要使用的 ECMAScript 版本，并且还可以通过运行带有标志的 Node.js 来启用特定的实验中的特性。

(摘自：node.js官网)

Node服务器相较于传统的Java服务器

1. 单线程的异步回调模式， 没有线程之间的竞争，防止了很多因其产生的问题，例如线程之间的切换很消耗资源；

2. IO之间的切换很高效，比如单纯拿取ajax请求，node远远超过Java；

3. node不适合做巨大的运算量，Java是能开启多个线程做并行运算的，而且是编译性语言，远超过解释性的node；

Node的主要应用场景

1. 开发桌面应用程序，例如vscode ，以及QQ微信等；

2. 开发服务器应用程序，大型站点和转发模式；

Node.js 框架和工具

Node.js 是一个底层的平台。 为了使开发者做事变得容易又来劲，社区在 Node.js 上构建了数千个库。其中Express尤为盛行。

Express（<http://expressjs.com/>）提供了创建 Web 服务器的最简单但功能最强大的方法之一，它的极简主义方法，专注于服务器的核心功能，是其盛行的关键。

2.3 数据库技术

MYSQL

为了能够使数据库和node服务器进行交互，我们采用mysql2驱动；另外，直接连接数据库有很多弊端（例如sql注入、连接池被长时间占用导致效率低下等问题），为此，我们可以使用ORM来帮助我们开发。

ORM

Object Relational Mapping 对象关系模型

通过ORM框架，可以自动把程序中的对象和数据库关联起来；ORM框架会隐藏具体的数据库底层的衔接，让开发者使用同样的数据操做接口，完成对不同的数据库的操做，如图2-3。

其优势还在于：开发者不用关心数据库，仅需关心对象，可轻易完成数据库的移植、 无需拼接复杂的sql语句即可完成精确查询等。

Sequelize（<https://sequelize.org/>），是Node中较为成熟的一款ORM框架，我们采用其帮助我们进行高效便捷的进行开发。

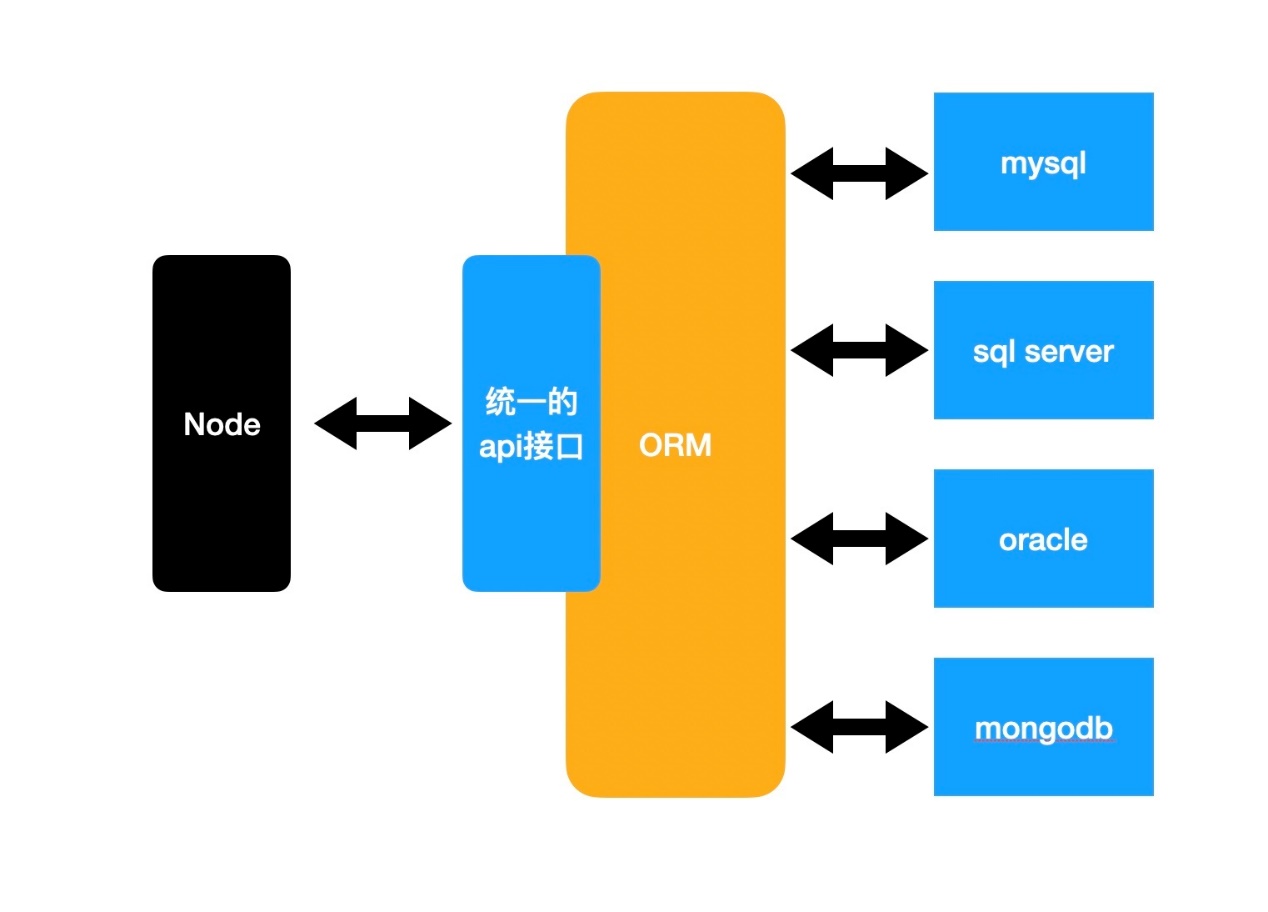


图2-3：ORM原理图

2.4 运行环境

本系统开发的运行环境在window和MAC系统下均可运行，要求是在操作系统上安装有MYSQL数据库、Node.js运行环境、VSCode开发工具、浏览器(推荐使用Google chrome浏览器)。

3 系统分析与设计

3.1 需求分析

本设计主要为了实现用户的业务办理和银行工作人员的高效工作。

系统的主要组成角色: 管理员、存取款用户、贷还款用户。

1. 管理员

管理员分为高级管理员和普通管理员，这样做的目的是权责分明，各行其是，有序高效的完成对银行的管理；

其权责和操做主要有：

(1) 管理员共有的操做：登录和退出登录。

(2) 高级管理员：查看银行帐户信息、管理银行资金（本金的增减）、修改利率信息（修改和增删利率）。

(3) 普通管理员：管理用户信息(存取款用户、贷还款用户)，主要分为：删除和修改用户信息，冻结/解冻用户，开启/关闭贷还款用户的大额贷款限制。另外，高级管理员含有这些操做的权限。

2. 存款用户

同一存款用户帐户可以分别办理取款和存款业务，以及抽奖操做。

主要操做有：用户注册、登录和退出、查看个人帐户信息、存款和取款、抽奖操做。

3. 贷款用户

同一贷款用户帐户可以分别办理贷款和取款业务。

主要包括：用户注册、登录和退出、查看个人帐户信息、贷款和还款操做。

3.2 系统的功能设计

系统结构如图：3-1所示；银行系统主要分为管理员子系统、存款用户子系统、贷款用户子系统。每个系统下面分为多个模块，管理员子系统模块有登录、查看和管理用户信息、管理利率和银行信息模块；存款用户系统主要有登录与注册、办理存款取款业务、抽奖、查看个人帐户信息模块；贷款用户子系统有登录和注册、办理贷款取款业务、查看个人信息模块。



图3-1：中国银行系统结构图

3.3 功能模块设计

3.3.1 用户登录、退出登录、注册模块

同一时间只能允许一位用户登录。

用户（管理员和存款用户、贷款用户）可通过顶部导航栏中的【请先登录】点击进入登录页面，输入账号，选择对应的身份，点击【登录】进行身份验证。

用户退出时，可以点击界面右上角的【退出】按钮，清除该用户的所有信息，便于其他用户登录。

存款用户和贷款用户还可以注册新用户，点击登录界面下面的【注册】，进入到用户注册页面，选择并填入对应账号类型的基本的信息；当用户填入的账号在数据库中存在时，系统会及时提示用户该账号已存在，并清空用户输入的账号信息，以便用户重新填入；当用户注册完成后，系统提示用户是否进入该账号的个人中心界面（直接跳过登录流程，提升用户体验），如果用户选择不进入，则返回首页。

另外，注册新的存款用户后，可以获得一次抽奖的机会作为奖励，奖励金额会自动计入该用户的存款中。

3.3.2 管理员模块

管理员根据权限和职能分为高级管理员和普通管理员，其权限和操做主要为：

高级管理员：查看银行帐户信息(银行资金运行情况、银行用户情况、以及将其使用图标来直观的进行展示，便于管理员清晰的了解银行运行情况以及做出决策)、管理银行资金（对银行本金进行增减操做）、修改利率信息（修改和增删利率）。

普通管理员：管理用户信息(存取款用户、贷还款用户)，主要分为：删除和修改用户信息，冻结/解冻用户，开启/关闭贷款用户的大额贷款限制。另外，高级管理员也应当含有这些操做的权限。

3.3.3 用户办理业务模块

本模块主要分为两部分：存款用户的存款业务和取款业务以及抽奖操做、贷款用户的贷款业务和存款业务；

存款用户可以办理存款（或取款）业务，用户点击选择办理业务的类型(存款或取款)，（系统为了增加用户存款的欲望，默认显示的业务办理区域是存款类型，当用户用户点击【取款】按钮后，系统自动切换到取款业务办理界面）；如果用户办理存款业务，用户根据系统流程，选择与输入与之相匹配的存款类型、存款时间、存款金额、利率，如果用户输入合理，则系统自动计算相应的利息并提示用户办理成功，反之，办理失败，清空用户输入，让用户重新输入；如果用户办理取款业务，则输入取款金额，如果合理（取款金额不大于存款金额与利息之和）则提示用户办理成功，反之，办理失败。另外，存款用户办理存款业务后，可以获得一次抽奖的机会作为奖励，奖励金额会自动计入该用户的存款中。

贷款用户可以办理贷款（或还款）业务，用户点击选择用户点击选择办理业务的类型(贷款或还款)，（系统为了增加用户贷款的欲望，默认显示的业务办理区域是贷款类型，当用户用户点击【还款】按钮后，系统自动切换到还款业务办理界面）；如果用户办理贷款业务，用户根据系统流程，选择与输入与之相匹配的贷款类型、贷款时间、贷款金额，（如果用户有大额贷款限制，则判断用户贷款累计是否达到限制的金额标准），如果用户输入合理，则系统自动计算相应的利息并提示用户办理成功，反之，办理失败，清空用户输入，让用户重新输入；如果用户办理还款业务，则输入还款金额，如果合理（还款金额不大于存款金额与利息之和）则提示用户办理成功，反之，办理失败，提示用户重新出入。

3.3.4 系统功能增强模块

1.多语言切换功能

为了使业务拓宽到全世界，这里可以采用系统多语言提示功能，这里暂时为中英文，用户点击导航栏顶部的【切换语言】即可完成系统语言的切换。

2.查看银行动态信息、利率、日期功能

用户打开网站时，首页可以展示一些关于银行的最新动态信息，例如利率和业务的调整，以及当前的系统日历等。

3.进入网站数据加载功能

用户初次在浏览器端打开该网站时，由于浏览器会去后台服务器中获取数据，由于网络的原因可能有一点延迟，此时为了提升用户体验，可以设置一个中国银行加载中的界面提示。

4 系统数据库设计

数据库的设计在整个银行系统设计过程中占有重要地位，一个数据库是否设计的合理直接影响到整个系统。而本系统涉及管理员对银行的资金管理，以及广泛的各种用户群体，其数据信息一定要设计的合理和精炼，以下来详细的介绍数据库的设计。

4.1 数据库E-R图

E-R图（实体-联系图），是一种以图形界面的形式向人们展现该设计所需要涉及到的模型关系，它的特点是图形化表示，直观简洁，不容易产生歧义，使用特别广泛。其组成元素分为实体、属性、联系。“实体”采用长方形来表示，“属性”采用椭圆来表示，“联系”采用菱形来表示。

1.管理员实体

姓名(name)、账号(account)、登录ID(loginId)、登录密码(loginPwd)、收益(income)、本金(corpus)、级别（rank）、身份类别（Flag）,如图4-1：



图4-1 管理员实体图

2.贷款用户实体

姓名(name)、账号(account)、登录ID(loginId)、登录密码(loginPwd)、性别(gender)、生日(birthday)、年龄（age）、电话号码（mobile）、工作(job)、公司名称(company)、大额贷款限制(limited)、是否冻结(isFreezed)、贷款用户身份（ident）、借款原因(cause)、借款资金(累计)(loan)、借款利息(累计)(interest)、身份类别（Flag），如图4-2：



图4-2 贷款用户实体图

3. 存款用户实体

姓名(name)、账号(account)、登录ID(loginId)、登录密码(loginPwd)、性别(gender)、生日(birthday)、年龄（age）、电话号码（mobile）、工作(job)、是否冻结(isFreezed)、存款资金(累计)(deposit)、存款利息(累计)(interest)、身份类别（Flag），如图4-3：



图4-3：存款用户实体图

4.利率实体

ID，类型、Flag，如图4-4：



图4-4：利率实体图

5. 银行信息实体

银行名称（name）、银行ID（bankId）、银行本金（corpus）、身份类别（Flag），如图4-5：



图4-5：银行信息实体图

高级管理员可以管理利率和银行信息。如图4-6：



图4-6：高级管理员和利率、银行信息关系E-R图

普通（高级）管理员可以管理贷款用户和存款用户信息。如图4-7：



图4-7：普通（高级）管理员和存款用户、贷款用户关系E-R图

贷款用户、存款用户可以对应相应的利率、银行信息来办理相应的业务。如图4-8：



图4-8：贷款用户、存款用户与利率、银行信息关系图E-R图

4.2 数据库相关表

本系统有4张表，表与表之间的耦合性较低，便于扩展；分别是管理员表、贷款用户表、存款用户表、利率表、银行信息表。数据库相关表如下：

1.管理员表

该表存储管理员的相关信息，管理员分为高级管理员和普通管理员，主要的区分为权限不同。详细结构见表4-1

表4-1 管理员表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 能否为空 | 描述 | 是否为主键 |
| Id | VATCHAR | 255 | N | 数据库自动分配 | N |
| Name | VATCHAR | 255 | N | 管理员名称 | N |
| loginID | VATCHAR | 255 | N | 登录ID | Y |
| loginPwd | VATCHAR | 255 | N | 登录密码 | N |
| rank | VATCHAR | 255 | N | 管理员级别 | N |
| flag | VATCHAR | 255 | N | 管理员表的标志，值为admin | N |

2.贷款用户表

该表主要存储贷款用户的个人和贷款等信息，主要由帐户、贷款金额和利息、贷款身份等组成；贷款用户根据自己的账号和密码进行登录（或注册新用户），查看自己的帐户贷款信息，以及办理贷还款业务等操做。详细结构见表4-2

表4-2 贷款用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 能否为空 | 描述 | 是否为主键 |
| Id | VATCHAR | 255 | N | 数据库自动分配 | N |
| account | VATCHAR | 255 | N | 贷款用户账号 | N |
| name | VATCHAR | 255 | N | 贷款用户姓名 | N |
| loginID | VATCHAR | 255 | N | 登录ID | Y |
| loginPwd | VATCHAR | 255 | N | 登录密码 | N |
| sex | BOOLEAN |  | N | 用户性别(男true女false) | N |
| birthday | DATETIME |  | N | 用户生日(时间戳) | N |
| Age | DATETIME |  | N | 用户年龄，通过生日计算所得 | N |
| mobile | INT | 11 | N | 用户电话 | N |
| job | VATCHAR | 255 | Y | 用户工作名称 | N |
| limited | BOOLEAN |  | N | 贷款用户是否有大额贷款限制 | N |
| isFreezed | BOOLEAN |  | N | 帐户是否被冻结 | N |
| ident | INT | 1 | Y | 贷款身份类别，取值为：1为个人，2为企业 | N |
| loan | BIGDECIMAL | 18 | Y | 贷款金额 | N |
| interest | BIGDECIMAL | 18 | Y | 利息金额 | N |
| company | VATCHAR | 255 | Y | 用户单位名称 | N |
| cause | VATCHAR | 255 | Y | 贷款原因 | N |
| flag | VATCHAR | 255 | N | 贷款用户表的标志，值为useri | N |

3.存款用户表

该表主要存储存款用户的个人和存款等信息，主要由帐户、存款金额和利息等组成；存款用户根据自己的账号和密码进行登录（或注册新用户），查看自己的帐户存款信息，以及办理存取款业务和抽奖等操做。详细结构见表4-3

表4-3存款用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 能否为空 | 描述 | 是否为主键 |
| Id | VATCHAR | 255 | N | 数据库自动分配 | N |
| account | VATCHAR | 255 | N | 存款用户账号 | N |
| name | VATCHAR | 255 | N | 存款用户名称 | N |
| loginID | VATCHAR | 255 | N | 登录ID | Y |
| loginPwd | VATCHAR | 255 | N | 登录密码 | N |
| sex | BOOLEAN |  | N | 用户性别(男true女false) | N |
| birthday | DATETIME |  | N | 用户生日（时间戳） | N |
| Age | DATETIME |  | N | 根据生日计算所得 | N |
| mobile | INT | 11 | N | 用户电话号码 | N |
| job | VATCHAR | 255 | Y | 用户工作名称 | N |
| isFreezed | BOOLEAN |  | N | 户是否被冻结 | N |
| deposit | BIGDECIMAL | 18 | Y | 存款金额 | N |
| interest | BIGDECIMAL | 18 | Y | 利息金额 | N |
| flag | VATCHAR |  | N | 存款用户表的标志，值为userii | N |

4.利率表

该表主要存储银行的利率和存贷款限制等信息，主要由存贷款利率类型、存款起薪等组成；高级管理员可以完成对利率的修改和增删等操做。详细结构见表4-4

表4-4 利率表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 能否为空 | 描述 | 是否为主键 |
| Id | VATCHAR | 255 | N | 数据库自动分配 | Y |
| A | INT | 11 | N | 大额存款起存金额 | N |
| B | INT | 11 | N | 中额存款起存金额 | N |
| C | INT | 11 | N | 大额贷款限制金额 | N |
| A\_1 | DECIMAL(18,6) |  | N | 大额贷款短期利率 | N |
| A\_3 | DECIMAL(18,6) |  | N | 大额贷款中长期利率 | N |
| B\_1 | DECIMAL(18,6) |  | N | 中额存款短期利率 | N |
| B\_3 | DECIMAL(18,6) |  | N | 中额存款中长期利率 | N |
| L\_s | DECIMAL(18,6) |  | N | 短期贷款利率 | N |
| L\_m | DECIMAL(18,6) |  | N | 中期贷款利率 | N |
| L\_l | DECIMAL(18,6) |  | N | 长期贷款利率 | N |
| flag | VATCHAR | 255 | N | 利率表的标志,值为rate | N |

5.银行信息表

该表主要存储银行的基本信息，主要由银行名称和银行ID，以及本金等字段，可以实现银行分支结构的增减；高级管理员可以完成对银行信息的修改和增删等操做。详细结构见表4-5

表4-5 银行信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 能否为空 | 描述 | 是否为主键 |
| Id | VARCHAR | 255 | N | 数据库自动分配 | N |
| Name | VATCHAR | 255 | N | 银行名称 | N |
| bankId | VATCHAR | 255 | N | 银行ID | Y |
| corpus | BIGDECIMAL | 18 | Y | 本金 | N |
| flag | VATCHAR | 255 | N | 银行信息表的标志，值为bank | N |

5 详细设计

本章的主要目标：设计和开发一个简易但囊括了基本银行业务功能的银行系统，为用户和管理员提供高效便捷的服务，使银行系统能够良好的运作和发展。

本系统设计主要内容包括：用户登录和注册模块、管理员管理银行和用户信息模块、用户办理业务和抽奖模块、中国银行简介模块、首页信息展示模块、局部功能增强模块（网站顶部功能控制导航、多语言切换、快速回到页面顶部、进入网站初始化界面）等模块。

5.1 用户登录和注册模块

5.1.1 用户登录

用户登录界面如图5-1-1(1)所示，用户登录时设置了三种角色，分别为管理员、贷款用户、存款用户，使用一个登录页面可以满足三种角色的登录需要；首先用户输入自己的账号(管理员输入loginID)和密码信息，在下拉框中选择对应的登录身份，如果账号、密码、身份三者在数据库中相匹配，则会登录进入相应的页面，如果不匹配，则提示用户登录失败。用户输入信息如图5-1-1(2)。



图5-1-1(1)：用户登录初始界面



图5-1-1(2)：用户输入信息

本系统中的用户登录功能的实现，流程为：用户在浏览器端输入数据，浏览器将此数据发送给服务器，服务器与数据库交互，查询该用户是否存在，如果存在，则返回该用户的信息，浏览器将信息存储到数据仓库中，网站中所需用户信息时，均在数据仓库中读取即可；如果不存在，返回空信息到浏览器端，浏览器提示用户输入信息有误。

登录功能的浏览器端核心代码为：

 async toLogin() {

      if (

        !this.loginFormData.loginId ||

        !this.loginFormData.loginPwd ||

        !this.loginFormData.flag

      ) {

        alert("值缺失！");

        return;

      }

      const userObj = {

        loginId: this.loginFormData.loginId,

        loginPwd: this.loginFormData.loginPwd,

        flag: this.loginFormData.flag,

      };

      const user = await this.$store.dispatch("login", userObj);

      if (user) {

        // log in

        this.$router.push(`/${userObj.flag}`);

        console.log("user info :", user); // 从数据库中拿到了该用户的所有信息

        try {

          this.$store.commit("setUserData", user);

        } catch (error) {

          new Error("store write fail .");

        }

      } else {

        alert("账号密码、类型不匹配");

      }

    }

5.1.2 用户注册

用户注册界面如图5-1-2(1)(2)所示，用户注册界面主要完成贷款用户和存款用户的注册；为了时用户体验更加友好，当用户注册成功之后，可以直接进入对应账号个人中心页面，无需登录。由于两种用户身份的大部分输入信息一致，可以通过一个通用的登录页面实现，区别的地方根据用户选择注册身份来区别展示即可，区别部分：（1）左侧的界面提示图片不同，为了用户友好的交互体验而设计；（2）贷款用户注册时，还需填入所在单位；

功能实现和实现流程：（1）用户输入账号后和选择身份类型后，系统会自动将其在数据库中查重，如果重复，提示用户“该账户已被注册，请重新填入账号”；（2）密码确认，如果用户两次输入的密码不一致，提示用户重新输入；（3）电弧号码验证，这里由于电话号码类型众多且条件有限，没有采用发送验证码验证的功能，采用正则表达式，验证用户1开头，且长度达到11位数即可通过验证。

该部分的主要代码如下：

// vue src/views/Register/index.vue

 // 去数据库中验证 账号是否可用

    async accountIsUseable() {

      console.log("is repeat ? to db valiable .");

      const userr = await this.$store.dispatch("queryUser", {

        account: this.userInfoTemp.account,

        flag: this.userInfo.flag,

      });

      if (userr) {

        this.tipMsg("warn", "该账号已被注册，请重新填写！");

        this.userInfoTemp.account = "";

        this.userInfoTemp.ps1 = "";

        this.userInfoTemp.ps2 = "";

        return;

      }

      // 验证通过，赋值

      this.userInfo.account = this.userInfoTemp.account;

    },

    //   密码确认

    rules\_passwordConfirme() {

      if (this.userInfoTemp.ps1 !== this.userInfoTemp.ps2) {

        this.tipMsg("warn", "密码不一致,请重新填写！");

        this.userInfoTemp.ps1 = "";

        this.userInfoTemp.ps2 = "";

        return;

      }

      this.userInfo.loginPwd = this.userInfoTemp.ps2;

    },

    //   联系电话 验证(规则:1开头，11位数)

    rules\_mobile() {

      const result = /^1\d{10}$/.test(this.userInfoTemp.mobile);

      if (result) {

        this.userInfo.mobile = this.userInfoTemp.mobile;

      } else {

        this.tipMsg("warn", "电话号码有问题，请重新填写！");

        this.userInfoTemp.mobile = "";

        return;

      }

    }



图5-1-2 (1)：贷款用户注册页面



图5-1-2 (2)：存款用户注册页面

5.2管理员管理银行和用户信息模块

管理员页面如图5-2(1)(2)所示；根据权限等级可分为高级管理员和普通管理员；权限不同，对应的操做和展示内容随即不同，高级管理员含有所有权限，其主要权限侧重于对银行的管理；普通管理员只能拥有少部分权限，其主要侧重于对用户信息的管理；

高级管理员可以查看和操做银行的所有信息，例如查看银行资金运行情况、银行用户情况，查看和修改银行的本金、利率等信息；采用图表、数字等形式展示，更加有利于管理员直观的对银行运行情况的了解以及做出正确的决策和操做。

高级管理员由于也拥有普通管理员的权限和操做界面，但其主要侧重不在此，可以在顶部设置一个显示/影藏普通管理员操做区域展示的按钮，便于界面的简洁和方便。

普通管理员只有查看和修改用户信息的权限和操做，管理员可以查看银行存贷款最多的三位用户、输入需要查询用户的账号和选择身份类型来查找指定的用户，以及对其删除和修改部分用户信息（例如对于存款用户有修改密码、电话号码、工作、冻结/解冻，对于贷款用户有修改密码、电话号码、工作、单位、设置/取消大额贷款限制、冻结/解冻），如图5-2(3)所示。



图5-2(1)：管理员页面



图5-2(2)：管理员页面



图5-2(3)：管理员查询和修改指定用户

主要开发技术和实现过程：

银行资金运行和用户信息情况,如图5-2(1)中的下半部分；该部分大部分的数据值采用Vue中的计算属性(computed)来获得，这样的好处是，可以汇总每个用户的信息来动态的实时获取最新的数据，展示银行的资金运行情况；右侧的图标采用ECharts图标工具库。

高级管理员修改银行利率、修改银行本金的情况，如图5-2(2)中上半部分，该部分为纯粹的操作数据库中的数据部分，流程为在浏览器端输入相应的数据，然后提交发送给服务器中，服务器将其数据写入数据库中，并返回成功的状态值给浏览器端，提示修改完成，即可完成数据的修改。

普通管理员管理用户信息，如图5-2(2)中中间“查找指定用户”部分；管理员输入待查询的用户账号，选择其对应的身份信息，即可查询，流程为：浏览器将管理眼输入的用户（账号、身份类型）信息发送给服务器中，服务器将其在数据库中进行查找，如果该用户存在，将其用户所属的全部信息返回给浏览器中，浏览器将其写入数据仓库UserQuery状态中存储，该界面并从该该状态中拿取数据渲染到页面中，如图5-2(3)所示；如果不存在该用户信息，返回空给浏览器端，浏览器提示“该用户不存在！”。

展示存贷款用户最多的前三位用户，如图5-2(2)中下半部分所示；该部分流程为：服务器拿取数据库库中所有用户信息，并按照升序排列，拿取最下面的三位用户即可，将其返回给浏览器，写入数据仓库的userData状态中，该状态为一个二维数组，分别存放存款用户和贷款用户的三位用户信息，该界面拿取该数据渲染在浏览器中展示。

该部分的主要代码实现如下：

// node src/servers/userii\_Service,js

const Sequelize = require('sequelize');

exports.getAllUser = async function () {

    /\*\*

     \* 找出存款数最多的前三位用户

     \*/

    const users = await User.findAll({

        order: Sequelize.literal('deposit') // 将存款用户按照默认的升序排列

    });

    const result = JSON.parse(JSON.stringify(users));

    const usersTop =[];

    usersTop.push(result[result.length-1]); // max

    usersTop.push(result[result.length-2]); // max-1(前一位)

    usersTop.push(result[result.length-3]); // max-2

   return usersTop;

};

5.3用户办理业务和抽奖模块

本系统中的用户分为贷款用户和存款用户，其分别对应不同的业务类型，如存款用户可以办理存款、取款业务以及抽奖操做，贷款用户只能办理贷款、还款业务；本节将详细的介绍一下实现过程。

5.3.1 存款用户办理业务和抽奖模块

存款用户办理存取款页面如图5-3-1(1)(2)所示，存款用户进入个人中心页面后，呈现会用户的是个人信息和帐户资金情况（存款和利息）、还有抽奖按钮（抽奖必须是用户有相应的抽奖机会，注册新用户或者办理存款业务可以获得）、用户办理业务区域（默认展示存款业务）；用户可以根据流程选择相应的利率类型和输入相应要求的金额与年限，即可办理存款业务（存款起存金额100元起），如果用户填写有误时，系统自动及时提示用户（以消息弹框的形式）；用户也可以点击跳转到办理取款业务界面；用户业务办理完成后，界面进行数据更新，显示最新的账户信息。

该模块的主要实现流程：当用户选择利率类型和输入的金额时间相匹配，通过验证后，浏览器将计算其存款和利息，并将其（账号、存款、利息），返回给服务器，服务器将其写入数据库中，完成后服务器返回给浏览器状态码，浏览器弹出成功消息，此次存款业务办理完成；取款业务则则判断用户输入的取款金额是否不大于存款与利息之和，否则弹出提示消息，提醒用户重新输入。

如果用户帐户被冻结时，用户无法办理存取款业务，需要联系管理员解冻。如图

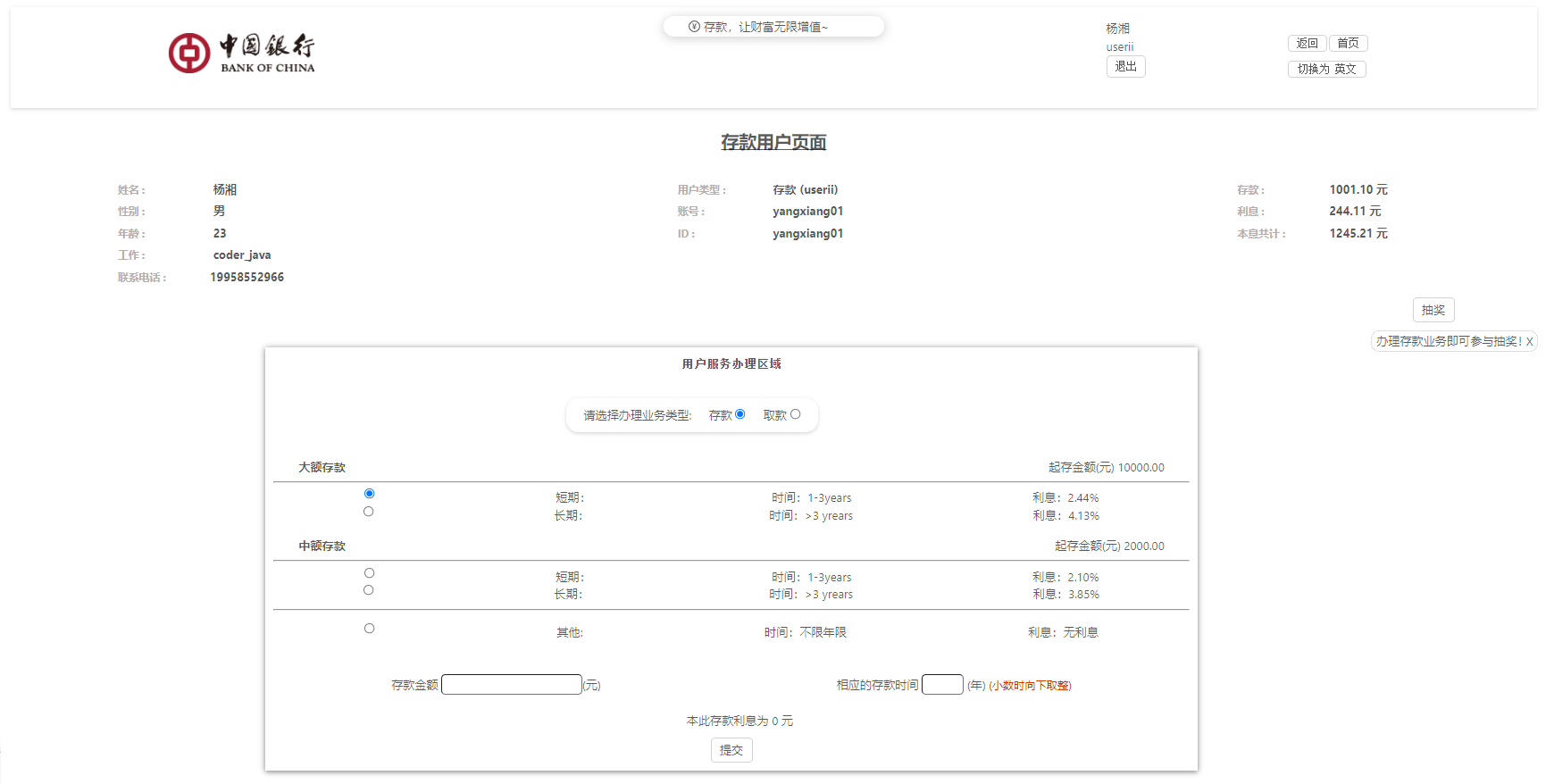


图5-3-1(1)：存款用户个人中心页面（办理存款业务部分）



图5-3-1(2)：存款用户办理取款业务部分

存款用户在办理存款业务后，可以进行一次抽奖，如果用户没有办理存款业务（或使用完一次机会后），则提示用户如图5-3-1(3)所示；用户点击按钮进入抽奖界面，如图5-3-1(4)所示，点击【start】进行，此时按钮随即影藏，用户一次机会使用完；抽奖结果如图5-3-1(5)所示，并提示用户，将所得奖金存储到其存款中，并返回用户页面。

实现的主要流程为：从区间[1,30]中随机生成一个数字（向下取整），单位为元，将其存储到用户存款中，主要代码如下：

 methods: {

    getRandom(min, max) {

      return Math.floor(Math.random() \* (max - min) + min);

    },

var index = this.getRandom(0, this.colors.length);

      return this.colors[index];

    },

    async startHandle() {

      this.lottery.auth = false; // 使用一次机会使用完毕后关闭

      await this.preRandom();

      this.lottery.value = this.randomArrs[this.randomArrs.length - 1];

      alert(`抽奖结果为 ${this.lottery.value}元 将会自动进入您的存款中!`);

      // write store

      const newValue = +this.userData.deposit + +this.lottery.value;

      this.userData.deposit = newValue;

      //  back

      this.$router.go(-1);

    }

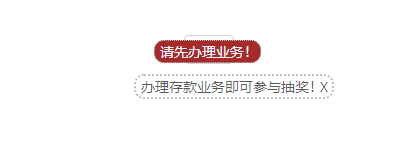


图5-3-1(3)：存款用户进行抽奖操做（提示信息）

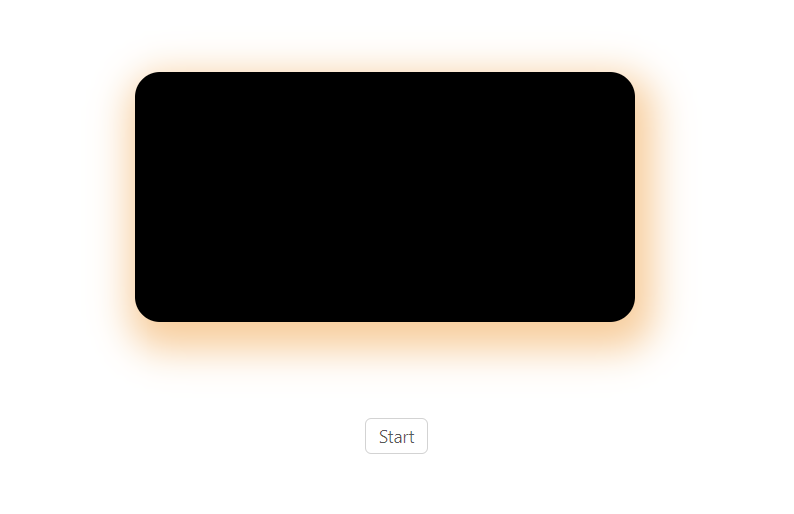


图5-3-1(4)：存款用户进行抽奖操做（开始前）

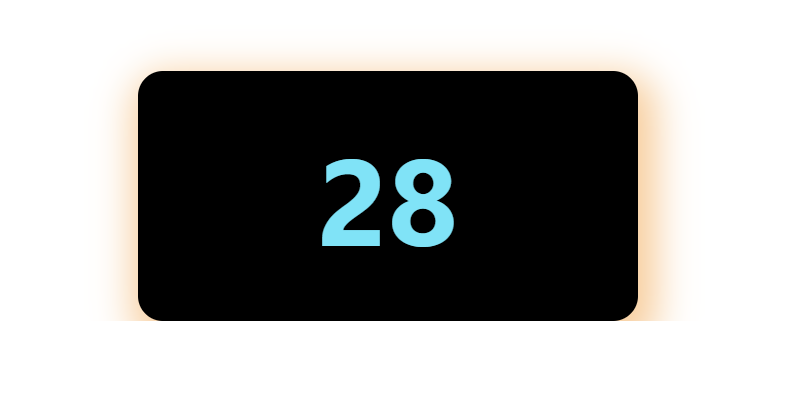


图5-3-1(5)：存款用户进行抽奖操做（完成）

5.3.2 贷款用户办理业务模块

贷款用户页面如图5-3-2(1)(2)所示；贷款用户办理贷款业务时，首先选择贷款身份，分为个人贷款和企业贷款（如果用户选择企业贷款，界面自动显示出企业名称输入框需要，须用户填入）；用户输入贷款原因时，不得小于5个字符；然后选择贷款利率类型与输入金额、时间，如果用户累计贷款达到银行设定的用户大额贷款限制时，银行将认为有借贷分险，需要用户联系银行管理员来为其开启权限（关闭大额贷款限制）；大体流程和存款用户相同，这里不再赘述。



图5-3-2(1)：贷款用户个人中心页面（办理贷款业务部分）



图5-3-2(2)：贷款用户办理还款业务部分

5.4中国银行简介模块

该模块为关于中国银行的基本介绍部分，如图5-4(1)(2)(3)所示；该部分以浏览器端纯静态页面展示为主，内容为中国银行的基本信息、历史由来、热门活动、网站页尾信息。



图5-4(1)：关于中国银行的基本信息



图5-4(2)：悠久的历史



图5-4(3)：热门活动与网站页尾信息

5.5首页信息展示模块

该模块是中国银行的首页部分，如图5-1(1)和图5-4(3)；该部分是中国银行的形象部分，也是用户打开中国银行门户网站时首先呈现的部分，主要组成部分有轮播图展示、重要公告和便捷服务、利率展示、日历、部分热门活动和网站页尾信息等；如下是各部分的基本实现。

轮播图组件和日历组件采用了组件库AntDesign（<https://www.antdv.com/docs/vue/introduce-cn/>）；

重要公告和便捷服务部分采用纯静态展示，点击链接即可完成跳转到对应的链接中；

利率展示部分为一个独立的组件，首页直接引入该组件展示到相应的位置上即可；在网站初始化数据加载数据时，浏览器向服务器发送请求，服务器从数据库中拿到数据并返回给浏览器中，浏览器将其写入数据仓库rateData状态中，利率展示组件随即读取数据仓库中的值渲染到界面上，即可完成展示。

首页底部为热门活动与网站页尾信息，如图5-4(3)，该部分也是纯静态展示，作为一个独立的组件便用在中国银行介绍页面和首页中，提高了组件的复用性。

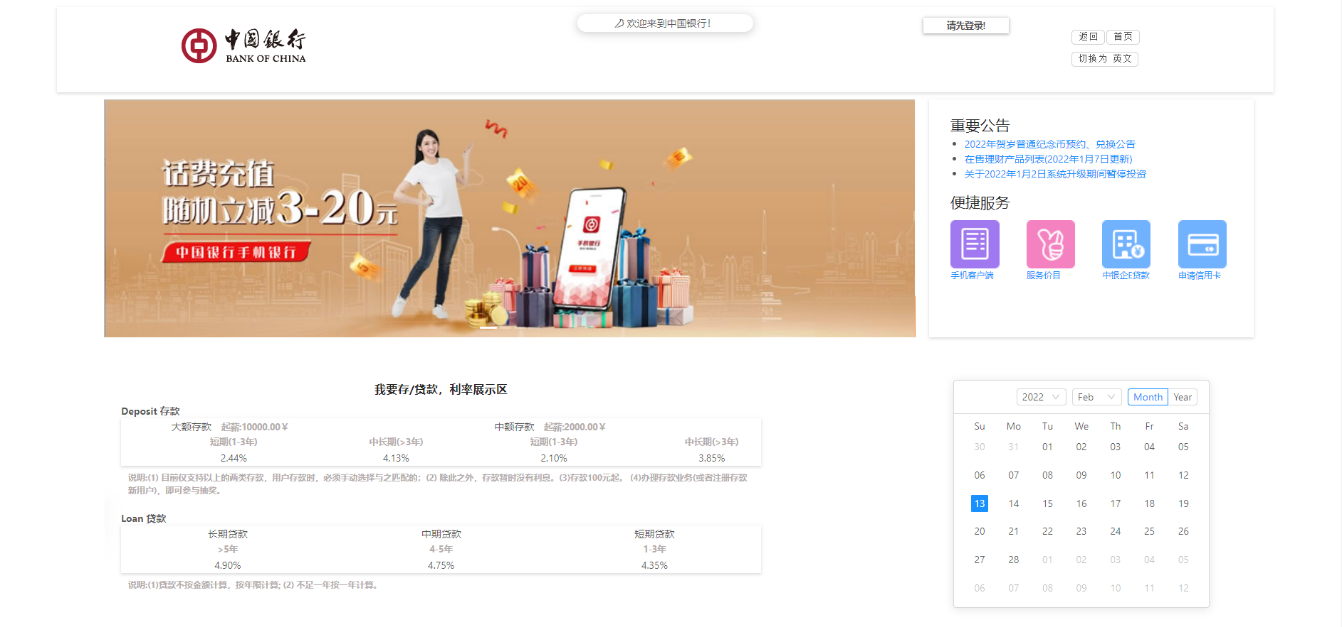


图5-5(1)：首页

5.6局部功能增强模块

该部分主要实现了网站功能的增强，以提升用户体验为目标，主要部分有网站顶部功能控制导航、多语言切换、快速回到页面顶部、进入网站初始化界面。接下来将详细的介绍各部分的实现过程。

5.6.1 网站顶部功能控制导航

该部分为整个网站的导航 ，如图5-5(1)；用户点击左侧银行的logo即可跳转到银行介绍页面；中间为一个消息提示栏，不同的页面提示不同的语句，给予用户良好的交互体验；右侧的【请先登录】登录按钮，提示用户登录，用户点击即可进入登录页面，用户登录之后替换为用户的登录信息，如图5-3-2(1)相应部分；右侧的【返回】和【首页】分别点击实现相应的功能；【切换语言】按钮点击即可实现系统提示语言的切换，下面将进行详细的介绍。

5.6.2多语言切换

该部分位于网站顶部功能控制导航的右下角的【切换语言】按钮，点击即可实现系统提示语言的切换；主要的实现思想是采用Vue数据仓库技术，编写相应的语言包，点击切换语言类型标志即可完成切换；该部分的主要代码实现如下：

// vue  src/store/languagePackage/admin.js

export default {

    // Admin page

    WelcomeManageBankFund: {

        cn: '欢迎管理银行资产',

        en: 'Welcome Manage Bank Fund'

    },

    WelcomeManageUserInfo: {

        cn: '欢迎管理银行资产',

        en: 'Welcome Manage User Information'

    },

}

// vue  src/store/idnex.js

import mutation from './mutations'

import action from "./actions"

import LanguagePackage from './languagePackage' // clinet 本地的，不用去server获取

const store = new Vuex.Store({

  // 仓库配置对象 全局

  state: {

    lang: 'cn', // 语言种类标志

    language: LanguagePackage // 语言包

  },

  // 同步

  mutations: {

    ...mutation

  },

  // 异步

  actions: {

    ...action

  },

})

  // vue src/components/TopBanner/index.js

    // 点击切换语言

    handleSwtichLang() {

      if (this.lang === "en") {

        this.$store.commit("setLang", "cn");

      } else if (this.lang === "cn") {

        this.$store.commit("setLang", "en");

      }

    },



图5-6-2(1)：中文

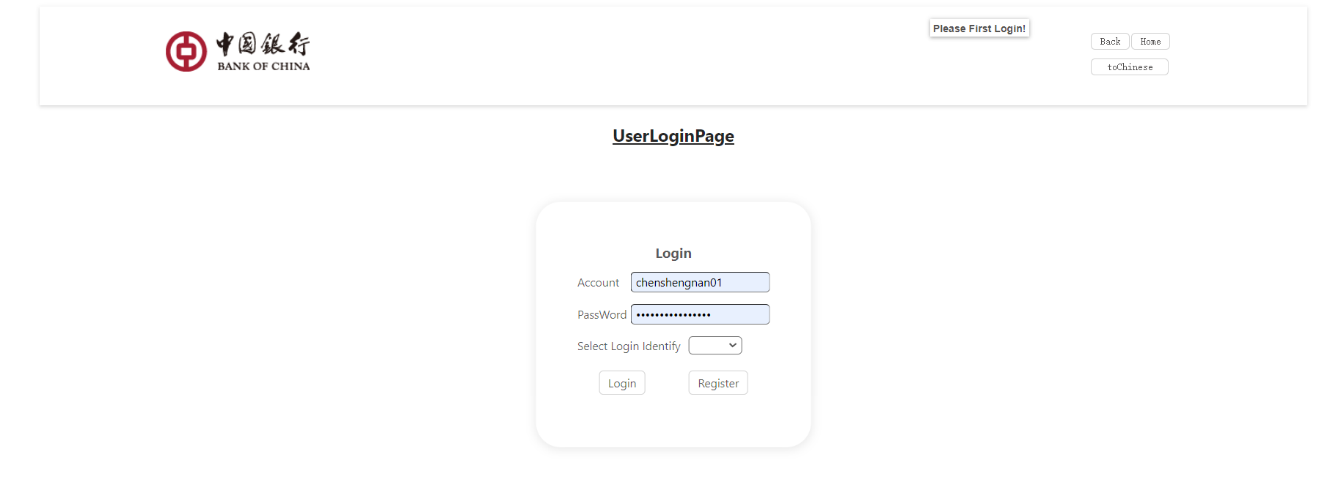


图5-6-2(2)：英文

5.6.3快速回到页面顶部

该部分如图5-2(2)的右下角所示；该部分作为一个独立的组件，应用在整个网站中，默认影藏，当页面滚动距离顶部超过500像素时，该组件显示，点击即可回到页面顶部。主要代码实现如下：

// vue src/components/ToTop/index.vue

<template>

  <div id="goTop" class="to-top-container">

    <transition name="fade">

      <div class="goTop" v-show="goTopShow" @click="goTop">

        <i class="goTopIcon"></i>

      </div>

    </transition>

  </div>

</template>

<script>

export default {

  name: "goTop",

  data() {

    return {

      scrollTop: "",

      goTopShow: false,

    };

  },

  methods: {

    /\*\*

     \* 监听滚动条

     \*/

    handleScroll() {

      this.scrollTop =

        window.pageYOffset ||

        document.documentElement.scrollTop ||

        document.body.scrollTop;

      if (this.scrollTop > 500) {

        this.goTopShow = true;

      } else {

        this.goTopShow = false;

      }

    },

    /\*\*

     \* 回到顶部

     \*/

    goTop() {

      let timer = null,

        \_that = this;

      cancelAnimationFrame(timer);

      timer = requestAnimationFrame(function fn() {

        if (\_that.scrollTop > 0) {

          \_that.scrollTop -= 50;

          document.body.scrollTop = document.documentElement.scrollTop =

            \_that.scrollTop;

          timer = requestAnimationFrame(fn);

        } else {

          cancelAnimationFrame(timer);

          \_that.goTopShow = false;

        }

      });

    },

  },

  mounted() {

    window.addEventListener("scroll", this.handleScroll);

  },

  destroyed() {

    window.removeEventListener("scroll", this.handleScroll);

  },

};

</script>

5.6.4进入网站初始化界面

如图5-6-4，当用户初次在浏览器端打开该网站时，由于浏览器会去服务器中获取数据，由于网络的原因可能有一点延迟，此时为了提升用户体验，可以设置一个动态加载中的界面提示（左侧为中国银行的logo，右侧是一个动图）。



图5-6-4：网站动态加载中

6 系统测试

6.1 系统测试的目的与意义

系统测试在整个项目开发的过程中起着举足轻重的作用，主要分为局部测试和整体测试；局部测试：（测试主体：自己）每个模块开发完成之后，需要测试功能是否完整，是否实现了该模块的业务需求，是否存在功能漏洞等，尤其是用户办理业务时复杂且严谨的计算是否无误。整体测试：（测试主体：他人）邀请多位用户来参与测试使用，要求覆盖到所有的业务部分，发现系统存在的问题和有待提升的地方，使得系统能够根号的推广和使用，以一种近乎完美的姿态呈现给用户。

6.2 测试过程

6.2.1 模块测试

由于该部分是银行系统的核心业务部分，投入使用后不能出现任何问题，所以采用局部测试+整体测试和自己测试+他人测试两种测试相结合的形式来尽可能地确保银行系统问题排查完毕；

测试内容为管理员测试管理用户和银行信息是否达到预期，实现了业务需求，要求覆盖每一个业务需求和功能点；存款用户是否可以正常地进行存款和取款、抽奖等操做，以及金额计算是否准确无误等；贷款用户是否可以正常地进行贷款和还款操作，尤其是一些边界值进行重点测试。

6.2.2 功能测试

该部分主要针对于系统功能增强模块，采用他人测试地形式，提出一些良好地修改意见等；

主要测试内容为网站顶部功能控制导航、多语言切换、快速回到页面顶部等是否达到了预期效果。

6.3 测试结果

通过本轮多次测试，采用多种方式（局部测试+整体测试和自己测试+他人测试、抽检和全覆盖）相结合地形式，发现了系统中存在地一些问题和漏洞，例如当用户进行注册时，首先选取一种身份并输入账号通过账号查重，然后再迅速切换另一种身份（由于原有地查重验证已通过，所以不再进行验证）就有可能在另一种身份中重复账号，为此通过Vue技术的watch来检测用户是否再次切换身份，一旦切换就触发账号查重操做来予以解决。通过不断地修改和完善，本系统以一个比较理想地状态呈现在了用户视野中，取得了用户地肯定，达到了预期的结果和目的。

7 结束语

本系统介绍+本课程的要求实现

参考文献和技术网站

[1]《基于web的毕业论文管理系统》 <https://www.doc88.com/p-5714921975775.html>

致谢

本系统的完成，首先感谢自己重视这次毕业设计选题的开始，很早就开始准备了，以及过程中不断地综合在校期间所学的各种技术，以及将其融会贯通以完成业务需求，感谢自己无数个日日夜夜的奋战，感谢自己再遇到问题时坚忍不拔，到处寻找解决方案，通过本此的系统设计，很好的锻炼了我综合大学期间所学的一些技术的使用和掌握和解决问题的能力；感谢我的好同学兰育平（2021届毕业生），多次参与和指导了我的毕设，提出了一些宝贵的指导意见，感谢参与我系统测试部分的多位同学，是他们让我的系统更加完善，最后，感谢我的指导老师。另外，还要感谢在百忙之中对论文进行评审的专家，论文撰写过程难免存在一些不之处，恳请老师专家们予以批评指正。