# 客户信息管理系统设计文档

## 系统概述

本系统是一个高度可定制的客户信息管理平台，支持运维人员管理客户资料，并提供灵活的前端界面布局配置。系统采用SaaS架构，前后端分离，后端基于.NET微服务和SQLite数据库实现，前端使用Blazor框架结合Ant Design组件库。设计重点包括：**客户信息的动态扩展和版本化存储**、**高度可配置的前端显示布局（设计/运行模式）**、**丰富的字段交互功能（邮箱快捷操作、远程桌面连接、文件链接等）**、以及**多用户权限控制**和**多语言支持**。本设计遵循“前端优先”的迭代开发原则，每个阶段实现最小可用产品并保证系统可运行，不断在使用中完善功能。

## 系统架构设计

**总体架构：** 系统采用前后端分离的SaaS架构。前端是Blazor Web应用（可运行于WebAssembly或Server模式），使用Ant Design Blazor组件库实现现代UI。后端由若干.NET微服务组成，包括用户认证服务、客户信息服务等，通过REST API（或GraphQL）向前端提供数据。各服务将容器化部署，方便未来水平扩展。前端通过HTTP/HTTPS调用后端API，实现松耦合交互。这样的微服务架构保证了各模块的独立性和可伸缩性，同时满足SaaS多用户的需求。

**技术选型：** 后端采用最新的 .NET（例如 .NET 8）平台，使用ASP.NET Core构建Web API。持久层使用EF Core操作SQLite数据库（开发测试阶段SQLite方便部署，未来可切换SQL Server等）。鉴于“不重造轮子”的原则，我们尽量利用成熟的开源框架和组件：如身份认证使用ASP.NET Identity库实现用户注册登录；权限管理使用基于Role/Claim的机制；多语言支持利用.NET内置的IStringLocalizer资源系统或开源库实现；前端Ant Design Blazor提供了丰富的UI控件，减少大量UI开发工作[[1]](https://www.cnblogs.com/Can-daydayup/p/18927715#:~:text=%E9%A1%B9%E7%9B%AE%E7%89%B9%E6%80%A7)。这种技术栈确保系统简洁高效，并具备充分的扩展性。

**模型驱动开发(MDD)：** 本系统的数据模型设计具有高度的动态性。客户信息实体采用**模型驱动**思路，字段定义存储在元数据中，前端界面根据后端提供的模型元数据动态生成表单。这意味着新增字段无需更改前端代码，只需在后端元数据添加定义，前端即可自动显示对应输入控件。这样可以支持客户信息结构的频繁调整，满足业务变化需求。

**微服务划分：** 初期系统可采用**模块化单体**架构（即在一个应用内划分模块），随着功能增多再演进为微服务。主要模块包括： - **身份认证服务**：负责用户注册、登录、权限校验，采用JWT令牌机制在前后端传递身份。 - **客户信息服务**：核心业务服务，提供客户资料的增删改查及动态字段、版本管理等功能。 - **通知/邮件服务（可选）**：用于发送邮件（如邮件模板功能）等。 - **文件服务（可选）**：如需处理附件或网络文件的代理访问。 各模块通过轻量级REST API交互，保持无状态设计，方便将来独立部署和水平扩展。为了降低初期复杂度，第一版可以在同一后端中实现这些模块，通过命名空间或子项目区分，后续需要时再拆分为独立微服务。

## 数据模型设计

**客户信息实体：** 由于客户信息字段需要高度可扩展且版本化保存，我们采用**“版本号 + 通用扩展字段”**的设计方案[[2]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%20uid%E5%92%8Cname%E6%9C%89%E6%9F%A5%E8%AF%A2%E9%9C%80%E6%B1%82%EF%BC%8C%E5%BF%85%E9%A1%BB%E8%AE%BE%E8%AE%A1%E4%B8%BA%E5%8D%95%E7%8B%AC%E7%9A%84%E5%88%97%E5%B9%B6%E5%BB%BA%E7%AB%8B%E7%B4%A2%E5%BC%95)。具体而言，客户基础信息表设计如下：

Customer(  
 Id (主键),  
 Code (客户编码),  
 Name (客户名称),  
 Version (版本号),  
 ExtData (扩展字段存储)  
)

其中 Version 字段标识当前记录的数据结构版本，用于解释 ExtData 中包含的属性含义[[2]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%20uid%E5%92%8Cname%E6%9C%89%E6%9F%A5%E8%AF%A2%E9%9C%80%E6%B1%82%EF%BC%8C%E5%BF%85%E9%A1%BB%E8%AE%BE%E8%AE%A1%E4%B8%BA%E5%8D%95%E7%8B%AC%E7%9A%84%E5%88%97%E5%B9%B6%E5%BB%BA%E7%AB%8B%E7%B4%A2%E5%BC%95)。ExtData 则是一个**可扩展格式**（如JSON字符串）来承载除基础字段以外的所有自定义属性[[3]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%882%EF%BC%89%20version%E6%98%AF%E7%89%88%E6%9C%AC%E5%8F%B7%E5%AD%97%E6%AE%B5%EF%BC%8C%E5%AE%83%E5%AF%B9ext%E8%BF%9B%E8%A1%8C%E4%BA%86%E7%89%88%E6%9C%AC%E8%A7%A3%E9%87%8A)。这种设计允许我们在不改变表结构的情况下动态增加客户属性字段：当需要新增字段时，只需在后端定义新版本号和对应的字段解释，前端即可接受到新字段的定义和数据。

**版本化存储：** 版本号与扩展数据结合实现了数据的版本化：旧数据可以保留旧版本格式，新数据使用新版本格式，同时存储于同一表中[[4]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%E5%8F%AF%E4%BB%A5%E9%9A%8F%E6%97%B6%E5%8A%A8%E6%80%81%E6%89%A9%E5%B1%95%E5%B1%9E%E6%80%A7)。例如，初始版本Version=0包含字段集合A；当新增字段集合B后，版本提升为1，新建或更新的客户记录Version=1，其ExtData包含A+B字段；老的记录仍为Version=0，ExtData只有A字段。系统通过Version判断如何解析ExtData，从而支持新旧两种数据格式并存[[4]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%E5%8F%AF%E4%BB%A5%E9%9A%8F%E6%97%B6%E5%8A%A8%E6%80%81%E6%89%A9%E5%B1%95%E5%B1%9E%E6%80%A7)。必要时，管理员可运行迁移脚本，将旧数据ExtData补齐新字段并更新版本号[[5]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%882%EF%BC%89%E6%96%B0%E6%97%A7%E4%B8%A4%E7%A7%8D%E6%95%B0%E6%8D%AE%E5%8F%AF%E4%BB%A5%E5%90%8C%E6%97%B6%E5%AD%98%E5%9C%A8)（初期也可选择不立即迁移，以降低风险）。这一方案满足了**随时扩展属性、后台版本化存储**的核心需求[[4]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%E5%8F%AF%E4%BB%A5%E9%9A%8F%E6%97%B6%E5%8A%A8%E6%80%81%E6%89%A9%E5%B1%95%E5%B1%9E%E6%80%A7)。

**注：** 我们选取“版本+扩展字段”方案是考虑到实现简便且满足版本化需求[[6]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%E5%8F%AF%E4%BB%A5%E9%80%9A%E8%BF%87%E2%80%9Cversion%2Bext%E2%80%9D%E6%88%96%E8%80%85%E2%80%9Ckey%2Bvalue%E2%80%9D%E7%9A%84%E6%96%B9%E5%BC%8F%E6%9D%A5%E6%BB%A1%E8%B6%B3%E4%BA%A7%E5%93%81%E6%96%B0%E5%A2%9E%E5%88%97%E7%9A%84%E9%9C%80%E6%B1%82%EF%BC%8C%E5%B8%8C%E6%9C%9B%E6%B2%A1%E6%9C%89%E6%B5%AA%E8%B4%B9%E4%BD%A0%E8%BF%99%E4%B8%80%E5%88%86%E9%92%9F%EF%BC%8C%E6%9C%89%E6%94%B6%E8%8E%B7%E5%B0%B1%E5%A5%BD%E3%80%82)。另一种常见方案是采用键值表存储(EAV模型)，例如定义表CustomerProperty(CustomerId, Key, Value)来保存扩展属性[[7]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%E4%BB%A5%E4%B8%8A%E9%9D%A2%E7%9A%84%E7%94%A8%E6%88%B7%E8%A1%A8%E4%B8%BA%E4%BE%8B%EF%BC%8C%E5%8F%AF%E4%BB%A5%E8%AE%BE%E8%AE%A1%E4%B8%BA)。该方案查询灵活，每个属性可独立索引[[8]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%E4%BC%98%E7%82%B9%EF%BC%9A)。但EAV在属性较多时会产生海量记录，管理复杂。相比之下，版本+Ext的JSON方式更直观简单，初期实现成本低。如未来对单个字段查询和索引有强需求，我们可考虑切换到支持JSON查询的数据库或引入NoSQL[[9]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%20ext%E9%87%8C%E7%9A%84%E5%AD%97%E6%AE%B5%E6%97%A0%E6%B3%95%E5%BB%BA%E7%AB%8B%E7%B4%A2%E5%BC%95)。

**字段定义元数据：** 为了让前端动态渲染字段，我们在后端维护一套**字段定义表**或配置。例如定义实体FieldDefinition(FieldKey, DisplayName, DataType, Tags, IsComposite, Options, Icon, Actions, etc)。每当有全局新字段加入，管理员通过后台界面添加一条字段定义记录，包括： - **唯一标识**：字段键(Key)或ID，在ExtData JSON中用作键名。 - **显示名称**：各语言的标签名（支持多语言）。 - **数据类型**：如文本、数字、日期、布尔、或者特殊类型（邮箱、RDS服务器、文件路径、链接等）。 - **关联标签**：用于对字段分类打标签（可多标签）。 - **是否复合字段**：标记该字段是否由子字段组成（例如RDS服务器包含地址、用户名、密码等子属性）。 - **预设选项**：如字段是枚举/选项列表，可在定义中提供选项集合。 - **关联图标和动作**：若该字段有特殊交互功能（详见后述第5-9点），则定义其图标、触发条件和动作接口等。

有了字段定义元数据，前端在展示客户详情时，会根据当前客户Version获取适用的字段列表及定义，然后动态生成页面元素。例如Version=1对应字段集合{FieldA, FieldB, FieldC...}，前端遍历这些字段定义，生成对应的标签和控件，并填充来自ExtData的值。这样使得**前端界面由后端数据驱动**，确保高扩展性。字段定义表本身也可版本化管理（例如存储一个字段是在哪个版本引入），不过通过Version号我们已可明确区分。

**标签分类与分组：** 每个字段可以关联一个或多个标签（Tags），标签用于在前端对字段分类分组，但不限制于单一类别。默认情况下，我们将按照标签对字段进行分块显示（拥有相同标签的字段归为一组）。如果字段存在多个标签，可支持**多标签多处显示**或者**多标签单处显示**两种策略，由用户在界面设计时自行决定。例如字段X同时有标签「联系信息」和「重要」，用户可以选择在「联系信息」分组中显示该字段，或者在两个分组中都显示。标签仅作为灵活的分类元数据存在，系统不会预设固定的分类列表，这给用户自由去组合视图提供了支持。

## 前端功能与界面设计

**布局概览：** 前端采用响应式Web界面，主要包含两部分：左侧是客户列表和搜索栏，右侧是客户详细信息展示面板。用户登录系统后，首先看到客户列表（可按Code或Name搜索、排序）。点击某一客户条目，在右侧加载该客户的详细信息表单。详细信息表单根据用户定制的布局进行排版，呈现各字段值以及交互图标。界面提供中/日/英多语言切换，默认显示日文。

**设计模式 vs 运行模式：** 前端界面具有**设计模式**和**运行模式**两种状态，以满足布局可配置性要求。普通情况下处于运行模式，此时界面用于日常查看和输入数据，不允许调整布局；切换到设计模式后，界面元素变为可拖拽调整的状态，允许用户自定义布局。系统提供一个简单的开关（比如“布局设计”按钮）用于模式切换。设计模式下，用户可以执行以下操作： - **添加/移除字段显示：** 决定哪些字段在详情面板中可见（用户可能对某些字段不感兴趣，可以隐藏）。 - **分组和区块调整：** 基于字段标签，用户可以增删分组区块，或者将字段从一个分组拖动到另一个。默认初始布局按标签自动分组，如果用户不满意，可以重新分配。 - **拖拽移动**：用户可拖动整个分组区块或单个字段控件，在页面上自由调整其位置顺序。拖动时界面提供参考线对齐功能，以便多个区块自动对齐边缘或栅格[[10]](https://blazor.syncfusion.com/documentation/dashboard-layout/interaction-with-panels/dragging-moving-of-panels#:~:text=The%20Dashboard%20Layout%20component%20is,position%20without%20disturbing%20the%20layout)。 - **调整大小**：支持用户拉伸调整区块的宽度和高度。系统采用类似网格布局的对齐机制，当用户调整大小时，自动贴合预设的网格线以保持美观，也允许细微的像素级调整以满足个性化需求。

上述拖拽和调整通过Ant Design Blazor和自定义CSS/JS实现。我们考虑借助成熟的布局控件，例如Ant Design的栅格系统或第三方Dashboard布局组件，以实现**拖拽面板**和**自动对齐**功能[[10]](https://blazor.syncfusion.com/documentation/dashboard-layout/interaction-with-panels/dragging-moving-of-panels#:~:text=The%20Dashboard%20Layout%20component%20is,position%20without%20disturbing%20the%20layout)。例如，Syncfusion的Blazor Dashboard Layout组件就支持面板拖动、自动碰撞调整和占位提示[[10]](https://blazor.syncfusion.com/documentation/dashboard-layout/interaction-with-panels/dragging-moving-of-panels#:~:text=The%20Dashboard%20Layout%20component%20is,position%20without%20disturbing%20the%20layout)。在设计模式下，每当用户移动或缩放元素，我们实时保存新的坐标和尺寸（相对于容器的百分比或网格单元），确保退出设计模式时布局配置持久化到后台。运行模式下，这些配置被读取应用，所有元素固定在设定位置，避免误操作。

**字段呈现与操作：** 在客户详情界面，每个字段通常显示一个标签（字段名）和对应的值内容。对于一般文本/数字字段，显示为普通文本或输入框（视读写权限而定）。对于特殊类型字段，我们在显示时附加特定的**图标按钮**或**交互控件**来提供快捷操作（符合需求5-8）。以下是主要特殊字段类型的设计：

* **邮箱地址字段**：如果某字段类型被标记为Email，则在该字段内容旁显示一个小“信封”图标✉【要求5】。当用户将鼠标悬停在字段上时出现此图标，点击或悬浮图标会弹出二级操作菜单。例如菜单选项包括：“使用邮件模板撰写”（点击后在新窗口打开邮件编辑，自动套用预定义模板）、“创建新邮件模板”（打开模板管理界面）、“直接发送新邮件”（调用本地邮件客户端或系统内置邮件发送功能写信）。实现上，我们可利用Ant Design Blazor的下拉菜单组件，将这些功能项作为下拉菜单列在信封图标下方。当用户选择某一操作，如果是打开外部邮件客户端，可以使用 mailto: 链接预填收件人实现；如果是内部发送，则跳转到系统的邮件撰写页面并带上该邮箱地址。
* **RDS服务器字段**：RDS（远程桌面）类型的字段比较复杂，它实际是一个**组合字段**【要求6】。当管理员定义某字段为“RDS服务器”类型时，我们认为它由子字段组成，包括：服务器地址/IP、用户名、密码等。前端可将这几个子属性组合在一起显示，例如在一个区块中列出地址、用户名字段，而密码可隐藏显示。针对RDS字段，我们提供交互图标“远程连接”🔗。当鼠标悬浮在RDS字段区块上时，显示出一个小的“连接”图标（以及可能一个“管理员连接”图标）。点击连接图标，前端会调用浏览器的本地协议或触发一个下载含有RDP配置的文件，以启动远程桌面连接。具体实现方式有两种：一是让后端生成一个带凭据的.rdp文件并通过浏览器下载，用户打开该文件即自动连接；二是利用Windows内置的mstsc命令通过特定URL协议启动[[11]](https://www.reddit.com/r/sysadmin/comments/azflsc/create_rdp_file_with_encrypted_password_in_it/#:~:text=There%20are%20two%20ways%20you,to%20the%20windows%20credentials%20manager)。由于浏览器无法直接运行本地exe，我们倾向于**生成RDP文件**方案，并提示用户点击打开。为了实现免密登录，我们将在RDP文件中嵌入用户名和密码信息（密码需加密存储）。参考业界方案，可以通过在RDP文件中加入 username:s:... 和 password 51:b:... 字段来保存凭据，或事先使用cmdkey命令将凭据存入Windows凭据管理器[[11]](https://www.reddit.com/r/sysadmin/comments/azflsc/create_rdp_file_with_encrypted_password_in_it/#:~:text=There%20are%20two%20ways%20you,to%20the%20windows%20credentials%20manager)。系统实现上，后端可调用Windows API对密码加密，然后写入RDP文件内容[[11]](https://www.reddit.com/r/sysadmin/comments/azflsc/create_rdp_file_with_encrypted_password_in_it/#:~:text=There%20are%20two%20ways%20you,to%20the%20windows%20credentials%20manager)。当用户点击连接图标时，前端获取该RDP文件（例如通过一个受保护的下载链接），用户打开后即可登入远程桌面。
* 如果某RDS服务器需要经由另一台跳板RDS才能访问，我们允许在字段定义中维护**依赖关系**（依赖的RDS字段ID）。当用户尝试连接目标RDS时，系统会先提示/启动依赖的RDS连接，然后再连接目标。如依赖服务器未连接，我们可引导用户先连接跳板。理想情况下，可实现一键开启两个RDP会话：先打开跳板RDS，再打开目标RDS（可能需要一定延迟确保前者就绪）。所有配置了连接方式和凭据的RDS字段，在后端都会预生成对应的RDP文件快捷方式，存储于服务器或提供给用户下载。这样用户无需每次输入凭据，即可快速连接。
* **文件/文件夹路径字段**：对于Windows共享网络上的文件或文件夹类型字段，我们在界面上提供一个“打开”图标（例如一个文件夹样式）【要求7】。当鼠标悬停到该字段时出现图标，点击后尝试使用系统默认程序打开该路径。实现上，如果用户电脑在局域网并已挂载共享，我们可以通过 file:// 协议打开。例如字段值为 \\\\Server\\Share\\Folder 或一个文件路径，前端调用 window.open('file://Server/Share/Folder') 尝试打开。如果浏览器出于安全策略无法直接打开，我们将退而求其次：要么提示用户复制路径自行在资源管理器打开，要么通过后端提供文件访问代理（将文件下载）。但在内网环境下，可以假设用户有权限访问该共享路径，则点击时由浏览器调用系统资源管理器打开链接。如果是文件，会用关联应用程序（如Excel、Word）打开。
* **超链接字段**：如果字段值是一个URL链接（字段类型标记为Link），则前端会将其渲染为可点击的超链接【要求8】。点击该链接会在新标签页中打开目标网址（使用<a target="\_blank">）。此外，我们可能在字段旁放一个小“外链”图标以提示这是可点击的链接。用户操作上与普通网页链接一致，无需特殊处理。

**字段功能封装与扩展：** 针对上述特殊字段的操作，我们在体系上进行了封装设计。每种字段类型（Email, RDS, File, Link等）对应一个**动作组件**，包括图标、操作触发方式（点击/悬浮）、以及可展开的子动作列表等。系统通过**接口或抽象类**定义这些字段动作，例如定义接口IFieldAction，包括属性：Icon, Tooltip, ActionType(Hover/Click), SubActions等，以及方法Execute()或回调。当需要支持新的特殊字段类型时，只需实现一个新的IFieldAction提供给前端即可。前端在渲染时检查字段定义：如发现附加的动作列表，则在UI上生成相应的图标和交互（比如AntDesign的Dropdown菜单可以承载SubActions的层级菜单）。由于SubActions也可以是带有自身图标和子动作的结构，此模型支持**无限层级的嵌套菜单**【要求9】。例如邮件字段的信封图标点击后展开二级菜单，其中“应用模板写信”又可以进一步弹出模板列表三级菜单等。数据格式上，字段定义里可为动作以嵌套JSON数组形式配置，这些将在前端递归解析生成菜单。

通过以上机制，系统具备了一个可扩展的字段功能体系：将来如果需要新增例如“电话号码”字段的拨号动作、“地址”字段的地图导航动作等，只需开发对应的前端交互组件并在字段元数据中配置，即可无缝集成。

**多语言支持（国际化）：** 本系统面向多语言环境，默认语言为日语，并提供中文和英文支持。我们严格遵循本地化最佳实践：**界面文本与资源和代码分离**。所有前端显示的标签、提示、菜单项等文本均使用资源键表示，在运行时根据当前语言加载对应语言包。实现上，采用.NET的资源文件(.resx)或JSON资源机制，并在Blazor中使用IStringLocalizer等进行切换[[12]](https://learn.microsoft.com/zh-tw/aspnet/core/blazor/globalization-localization?view=aspnetcore-9.0#:~:text=%E5%85%A8%E7%90%83%E5%8C%96%E5%92%8C%E7%95%B6%E5%9C%B0%E8%AA%9E%E7%B3%BB%E5%8C%96)。由于要求前端不维护独立多语资源，一切多语言内容由后端提供，因此我们的资源文件/库集中存放在服务器。前端在初始化时会从后端下载当前语言的资源字典（或通过嵌入式方式获取）。多语言资源可以存储在数据库中以便动态修改，或使用.resx文件编译发布（三语内容提前准备）。考虑到易于管理，我们可以建立**资源表**（例如 LocalizationResources(Key, ZH, JA, EN) ）存放各语言文本，实现任意语言配置即可使用[[1]](https://www.cnblogs.com/Can-daydayup/p/18927715#:~:text=%E9%A1%B9%E7%9B%AE%E7%89%B9%E6%80%A7)。管理员可以通过后台界面更新这些资源表内容，实现即时调整翻译而无需重新部署前端。

前端Blazor在启动时根据用户或浏览器偏好设置选择语言（也可由用户在界面上手动切换语言）。选择语言后，通过API获取对应语言资源，将其缓存到前端。本地所有显示调用一个比如T("ResourceKey")的方法，由其从缓存字典中取出当前语言文本显示。这样的机制保证**前端不硬编码任何文案**，且用户可随时切换界面语言而无需刷新页面。默认语言日语下，如果用户未登录情况下也可以根据浏览器Accept-Language自动显示日语，以提升体验[[12]](https://learn.microsoft.com/zh-tw/aspnet/core/blazor/globalization-localization?view=aspnetcore-9.0#:~:text=%E5%85%A8%E7%90%83%E5%8C%96%E5%92%8C%E7%95%B6%E5%9C%B0%E8%AA%9E%E7%B3%BB%E5%8C%96)。

## 用户和权限设计

系统提供用户注册和登录功能，每个用户拥有自己定制的视图配置。权限上，设定“管理员”“普通用户”等角色： - **管理员**拥有全局配置权限，能定义新的客户字段、编辑所有客户信息、以及分配客户可见权限给用户等。 - **普通用户**只能查看和编辑被授权的客户信息，且不能影响他人视图。

**用户注册/登录：** 用户注册需要管理员审批（因为涉及数据权限划分）。登录采用JWT或Cookie认证，基于ASP.NET Identity实现。密码安全存储，支持多因素认证等可扩展。

**客户权限分配：** 每条客户记录可以设定哪些用户或角色可访问。实现上有一张关联表，例如CustomerAccess(CustomerId, UserId, CanEditFlag)。管理员在后台勾选即可给用户赋权。前端在列出客户列表时，会根据当前用户筛选只显示有权限的客户。如果某用户试图通过直接URL访问未授权客户，后端会拒绝（返回403）。

**个性化视图配置：** 每个用户对客户详情界面的布局调整仅对自己生效【要求11】。我们为此建立UserLayout(UserId, LayoutConfig)表，存储用户的设计模式配置结果。LayoutConfig可以是JSON，包含该用户对各字段的显示/隐藏、分组顺序、位置尺寸等偏好设置。当用户登录并查看客户详情时，系统会加载其LayoutConfig应用。如果用户从未定制过，则使用系统默认布局（按标签分类的默认分组顺序显示）【要求10】。用户在设计模式下做出的改动，会即时保存到这张表中。这样确保每位用户都可以拥有**专属的查看体验**，而不影响他人。值得注意的是，如果管理员新增了全局字段，所有用户即使没有定制也应能在默认布局下看到（或由管理员决定初始可见性）；如果用户已有定制布局，则新增字段默认隐藏或放在某个默认组里，用户可以手动将其添加到界面。

**全局字段管理：** 只有管理员可以新增（或删除）全局字段，因为此操作会影响全局数据结构【要求11】。当管理员通过字段定义界面添加一个新字段时，后台会： 1. 为字段定义表加入新记录，分配新的字段Key和属性（并关联一个新的Version号，如果这意味着数据结构版本升级）。 2. 更新全局当前版本号。例如全局维护一个当前Customer版本=X，在新增字段后递增为X+1。 3. 对于数据库中已有的Customer记录，我们采取**延迟迁移**策略：即保留原Version不变，只有当该客户信息下次被编辑时，才自动升级为新版本（在保存时补齐新字段，未填写的以空值占位）。管理员也可以选择立即批量更新所有记录到新版本（这需要遍历修改所有ExtData，可在系统闲时执行）。 4. 通知前端应用新的字段定义。对于在线用户，可以通过SignalR等实时推送，让前端更新模型元数据；否则下次刷新页面自然获取新字段。

普通用户如果尝试“增加字段”（假设界面提供了该选项），应提示没有权限或者引导提交给管理员处理。一般情况下，为防止误操作，我们不允许普通用户新增影响全局的字段。

## 实现步骤

为确保每个阶段产品都可用并逐步扩展功能，我们规划以下迭代步骤：

1. **初始化项目结构** – 搭建基础的前后端工程。创建后端.NET解决方案，包含Web API项目和EF Core数据模型；创建Blazor前端项目并集成Ant Design组件库。实现基础的用户认证模块：设置ASP.NET Identity或简单的用户名/密码认证，建立用户表。实现多语言框架：准备日文为默认的资源文件或数据库资源，搭建IStringLocalizer服务，确保简单页面上的文本可以根据文化切换[[1]](https://www.cnblogs.com/Can-daydayup/p/18927715#:~:text=%E9%A1%B9%E7%9B%AE%E7%89%B9%E6%80%A7)。此阶段提交代码后，前端应用可以启动并显示一个登录页，支持日文界面和用户登录验证（可先写死一个管理员账户）。**（前端可运行）**
2. **客户基础管理功能** – 实现客户实体基础CRUD。设计Customer表（包含Id, Code, Name, Version, ExtData字段），以及初始的FieldDefinition表（预置几个基础字段如“Code/Name”本身作为字段定义，和示例扩展字段）。后端提供API接口：获取客户列表、按名称搜索客户、查看客户详情、创建/修改客户。前端开发客户列表页：调用API列出当前用户有权限的客户（暂时所有客户都可见，权限稍后添加），提供搜索框过滤Code/Name。实现点击客户项后，调用详情API获取客户数据并显示在详情面板。**在此步骤暂时不实现动态布局和设计模式**，详情面板可以简化为一个固定布局（例如上下排列几个基本字段）。由于ExtData解析尚未实现，这里可以暂时假设没有扩展字段或仅显示Code、Name、以及一个空的扩展区域。这个阶段完成后，用户能够通过界面增删改查客户的核心信息。**（基本增删查功能可用）**
3. **动态字段与版本化实现** – 引入动态扩展字段机制。在后台实现字段定义元数据管理：开发管理员界面或种子脚本来增加新的FieldDefinition，并调整客户详情API使其根据Version返回相应的字段结构和数据。具体来说，编写一个服务用于解析Customer的ExtData：当请求客户详情时，根据Customer.Version查询所有FieldDefinition中Version ≤ 客户Version的字段列表，构建出该客户应有的字段-值对；将其序列化为前端友好的格式（比如JSON数组，每项含FieldKey、DisplayName、多语言标签、Value等）。前端修改详情面板渲染逻辑：不再写死字段，而是遍历后端返回的字段列表，动态生成字段行。对于暂未支持的特殊类型字段，可先一律当作文本显示。实现管理员添加字段功能：提供一个简单页面，填写字段名称、类型、默认标签等，提交后调用后端API新增FieldDefinition，后端将全局Version+1并保存。为了简化，**此阶段可暂不处理旧记录版本迁移**：即假定管理员新增字段后，会通过运行小工具将所有Customer记录ExtData升级到新版本（比如填入空值），或者在查询时对缺失字段补默认值。更新后的系统应该能动态展示此前定义的所有字段，并在创建/编辑客户时一并处理这些字段值（需要修改客户保存API：接受动态字段数据并存入ExtData JSON）。**（动态字段生效）**
4. **布局设计模式** – 实现前端设计模式切换和布局配置保存。首先，定义前端布局配置的数据结构，例如：包含每个字段或分组的位置（行列坐标或像素）、尺寸（宽高）、所在分组、显示顺序、是否隐藏等信息。后端建立UserLayout表，用于保存用户的布局配置JSON。前端在详情页面上方添加“设计模式”开关按钮。设计模式启用时，为每个字段控件添加可拖拽的句柄和调整大小的控点。可以使用CSS position:absolute 配合拖拽事件来让元素移动，或采用Ant Design的栅格系统让元素位于不同的列宽位置，并通过增减列来实现宽度调整。为实现自动对齐和栅格吸附，可在元素拖动过程中，计算邻近元素边缘并自动贴靠。也可以考虑集成开源的拖拽库，例如 interact.js 或使用AntBlazor提供的拖拽指令。**简化实现**：这一版可以先实现“拖拽排序”而非完全自由布局——即允许字段块在垂直方向拖动换顺序，以及左右调整所在的列区域（比如两列布局）来改变宽度。这样复杂度较低但能满足基本需求。设计模式下，用户调整布局后，点击“保存布局”按钮，将当前布局配置通过API提交后端保存到UserLayout。运行模式再打开该客户时，就按照用户保存的布局来渲染：具体实现为后端在客户详情API增加用户布局参数，根据用户ID取其Layout配置应用在字段列表上（如有隐藏字段则不返回，有定位信息则附加给前端）。此阶段完成后，用户可以自主调整字段顺序和分组排列，并体验保存/还原布局的效果。**（布局定制可用）**
5. **特殊字段交互功能** – 丰富前端字段组件的交互行为，满足需求5-8。我们将逐项实现：
6. **邮件字段操作：** 在FieldDefinition中标记Email类型字段，并指定邮件图标和动作菜单。前端实现一个EmailField组件，渲染值加📧图标，使用Ant Design的Dropdown组件实现悬停弹出菜单。菜单项点击事件可以调用外部链接（mailto）或者触发内部功能。实现一个简易“写邮件”对话框页面，允许用户输入邮件内容发送（此步可集成SMTP发送邮件，或集成邮箱客户端暂略）。模板功能复杂，可暂时模拟：预置几个模板标题，点击后在写邮件对话框中自动填充内容即可。
7. **超链接字段点击：** 对于URL类型字段，前端直接用<a href>打开新窗口即可。确保值带有http://或https://前缀，否则在渲染时自动加上。
8. **文件路径字段打开：** 尝试通过浏览器打开文件路径。鉴于浏览器限制，为**最小实现**，我们提供一个按钮“复制路径”以方便用户手动打开。另外，如果是在Windows本地网络环境，尝试用window.open('file://..')。同时后端提供一个Download API，当检测到字段值是文件且用户点击“下载”选项时，后端通过网络路径读取文件流返回，实现直接下载。
9. **RDS字段连接：** 这是重点，实现前述生成RDP文件功能。需要在后端引用Windows的库（如果部署在Windows服务器）来使用System.Windows.Forms或RemoteDesktop API生成RDP文件。或者使用模板字符串填入目标地址、用户名、密码（密码需加密：可调用Windows DPAPI CryptProtectData在服务器上根据自身账户加密，这样生成的密码Blob仅服务器自身可解，不同客户端不能解密，但如果RDP文件在同一台服务器上打开则可用。这在分发上有限制，所以更可能采用cmdkey方法）。**简化方案**：不真正保存密码在RDP，改为在客户端使用cmdkey。即当用户点击连接时，先提供一段指引：要求用户在本机运行某命令保存凭据，然后RDP文件中仅保存服务器地址和用户名。考虑到用户体验，这里选择**嵌入加密密码**。实现上，可以使用开源脚本如PowerShell的ConvertTo-SecureString + ConvertFrom-SecureString得到密码blob[[11]](https://www.reddit.com/r/sysadmin/comments/azflsc/create_rdp_file_with_encrypted_password_in_it/#:~:text=There%20are%20two%20ways%20you,to%20the%20windows%20credentials%20manager)，填入RDP模板行。将RDP内容作为文件流返回，前端JavaScript触发一个隐藏链接下载 .rdp 文件并自动调用它（部分浏览器允许直接window.location = 'data:application/...'可能触发关联应用）。测试确保在典型Windows客户端上，一键点击可以打开远程桌面并自动登录。对于/admin模式，可以在生成文件时加入administrative session:i:1参数或创建另一个动作图标来提供管理员连接。
10. **依赖RDS顺序**：实现依赖关系：在FieldDefinition里如果配置Dependency字段，则前端在用户点击连接前，先检查依赖的RDS是否已连接（这可能无法程序化检测）。最简单做法：在依赖关系存在时，点击连接弹出对话框提示“请先连接依赖服务器X”，并提供一个按钮链接X的RDP。用户点击后打开X的RDP，然后需要再手动点击目标RDS连接。如果要自动链式打开，考虑通过浏览器延迟几秒后启动第二个RDP。但出于稳定性，此功能可在以后的版本逐步改进。先实现提醒机制即可。

完成以上后，前端针对特殊类型字段的交互基本满足需求。此时用户可以方便地点击邮箱图标发邮件、点击链接浏览网页、打开共享文件，以及一键远程连接服务器，大大提高效率。**（特殊字段操作可用）**

1. **权限和多用户** – 扩展系统支持多用户协作和权限控制。实现管理员分配客户权限界面：在客户列表页为管理员显示“可见性设置”选项，勾选哪些用户可访问该客户，提交后更新CustomerAccess表。修改获取客户列表API，使普通用户只能收到授权的客户列表。实现用户角色概念：在用户管理界面可以将用户设为管理员或普通用户，不同角色看到不同菜单功能（比如只有管理员看到字段定义管理、权限分配等）。前端测试用两个账号登录，一个管理员一个普通用户，验证权限生效。还需要确保普通用户无法通过前端接口调用越权修改（后端在每个API都根据UserID检查权限）。

此外，完善每个用户独立的布局配置。确保先前的UserLayout在多用户场景下正常工作——不同用户登录定制各自布局互不影响。可能需要提供一个“重置布局”功能以防用户布局错乱想回到默认。

在多用户基础上，进一步完善多语言：允许用户个人选择界面语言（保存到用户设置Profile中），下次登录加载其偏好语言。管理员也可设置系统默认语言（默认日文）。验证中英文资源包正确加载显示。**（多用户支持、权限控制、多语言切换可用）**

1. **持续改进和测试发布** – 经过上述步骤，我们已有一个功能完整的系统原型。最后一步侧重于完善细节、优化性能和用户体验：
2. **完善前端细节：** 如在设计模式下提供撤销/重置按钮、拖拽时的占位提示（参考Dashboard布局拖拽高亮位置的效果[[10]](https://blazor.syncfusion.com/documentation/dashboard-layout/interaction-with-panels/dragging-moving-of-panels#:~:text=The%20Dashboard%20Layout%20component%20is,position%20without%20disturbing%20the%20layout)），让用户明白拖拽结果。增强字段组件的UX，比如密码字段的显示/隐藏切换、小图标的hover提示（title属性）。
3. **安全性检查：** 对敏感信息如RDS密码确保在前端不明文显示，ExtData中的密码可加密存储（在数据库中可以额外加密敏感字段或者统一对ExtData加密存储，传输中走HTTPS保障）。用户权限严格校验，防止通过修改请求篡改他人数据。
4. **性能优化：** 由于采用SQLite，检查并添加必要的索引（如Customer Code索引用于搜索）。ExtData JSON字段无法直接索引，如果有查询需要可考虑维护部分冗余字段或使用全文检索方案。当前数据量可能不大，可暂不处理大规模性能问题。
5. **多浏览器兼容：** 测试在Chrome、Edge上文件打开、RDP下载是否正常。针对无法直接打开文件的浏览器行为，提供替代方案或提示。
6. **编写文档和部署脚本：** 撰写用户使用手册和运维部署文档。准备初始多语言资源文件（中文、英文翻译）以供选择。部署时以Docker方式容器化各服务，前端发布为静态文件由Nginx托管，提高在云环境下的易用性。

经过这一阶段的改进，我们将发布1.0版本。此版本功能虽不完备但胜在架构合理、扩展方便。在真实使用过程中，我们会根据用户反馈迭代，逐步完善高级功能（如更复杂的模板管理、自动RDP链路、移动端适配等），同时保持每次迭代后系统都可平滑升级运行。

**【参考文献】**

1. 王家奇士. “高可扩展表结构设计方案.” *CSDN博客*, 2017[[2]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%20uid%E5%92%8Cname%E6%9C%89%E6%9F%A5%E8%AF%A2%E9%9C%80%E6%B1%82%EF%BC%8C%E5%BF%85%E9%A1%BB%E8%AE%BE%E8%AE%A1%E4%B8%BA%E5%8D%95%E7%8B%AC%E7%9A%84%E5%88%97%E5%B9%B6%E5%BB%BA%E7%AB%8B%E7%B4%A2%E5%BC%95)[[4]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%E5%8F%AF%E4%BB%A5%E9%9A%8F%E6%97%B6%E5%8A%A8%E6%80%81%E6%89%A9%E5%B1%95%E5%B1%9E%E6%80%A7).
2. Syncfusion 文档. “Blazor Dashboard Layout 组件的拖拽移动面板.” *Syncfusion*, 2025[[10]](https://blazor.syncfusion.com/documentation/dashboard-layout/interaction-with-panels/dragging-moving-of-panels#:~:text=The%20Dashboard%20Layout%20component%20is,position%20without%20disturbing%20the%20layout).
3. Reddit讨论. “如何在RDP文件中嵌入加密密码.” *r/sysadmin*, 7年前[[11]](https://www.reddit.com/r/sysadmin/comments/azflsc/create_rdp_file_with_encrypted_password_in_it/#:~:text=There%20are%20two%20ways%20you,to%20the%20windows%20credentials%20manager).
4. 追逐时光者. “基于.NET 8 + Ant Design Blazor的开源后台框架特性.” *博客园*, 2025[[1]](https://www.cnblogs.com/Can-daydayup/p/18927715#:~:text=%E9%A1%B9%E7%9B%AE%E7%89%B9%E6%80%A7).
5. Microsoft Learn. “ASP.NET Core Blazor 全球化和本地化.” *Microsoft Docs*, 2025[[12]](https://learn.microsoft.com/zh-tw/aspnet/core/blazor/globalization-localization?view=aspnetcore-9.0#:~:text=%E5%85%A8%E7%90%83%E5%8C%96%E5%92%8C%E7%95%B6%E5%9C%B0%E8%AA%9E%E7%B3%BB%E5%8C%96).

[[1]](https://www.cnblogs.com/Can-daydayup/p/18927715#:~:text=%E9%A1%B9%E7%9B%AE%E7%89%B9%E6%80%A7) 一个基于 .NET 8 + Ant Design Blazor 开发的简洁现代后台管理框架 - 追逐时光者 - 博客园

<https://www.cnblogs.com/Can-daydayup/p/18927715>

[[2]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%20uid%E5%92%8Cname%E6%9C%89%E6%9F%A5%E8%AF%A2%E9%9C%80%E6%B1%82%EF%BC%8C%E5%BF%85%E9%A1%BB%E8%AE%BE%E8%AE%A1%E4%B8%BA%E5%8D%95%E7%8B%AC%E7%9A%84%E5%88%97%E5%B9%B6%E5%BB%BA%E7%AB%8B%E7%B4%A2%E5%BC%95) [[3]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%882%EF%BC%89%20version%E6%98%AF%E7%89%88%E6%9C%AC%E5%8F%B7%E5%AD%97%E6%AE%B5%EF%BC%8C%E5%AE%83%E5%AF%B9ext%E8%BF%9B%E8%A1%8C%E4%BA%86%E7%89%88%E6%9C%AC%E8%A7%A3%E9%87%8A) [[4]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%E5%8F%AF%E4%BB%A5%E9%9A%8F%E6%97%B6%E5%8A%A8%E6%80%81%E6%89%A9%E5%B1%95%E5%B1%9E%E6%80%A7) [[5]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%882%EF%BC%89%E6%96%B0%E6%97%A7%E4%B8%A4%E7%A7%8D%E6%95%B0%E6%8D%AE%E5%8F%AF%E4%BB%A5%E5%90%8C%E6%97%B6%E5%AD%98%E5%9C%A8) [[6]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%E5%8F%AF%E4%BB%A5%E9%80%9A%E8%BF%87%E2%80%9Cversion%2Bext%E2%80%9D%E6%88%96%E8%80%85%E2%80%9Ckey%2Bvalue%E2%80%9D%E7%9A%84%E6%96%B9%E5%BC%8F%E6%9D%A5%E6%BB%A1%E8%B6%B3%E4%BA%A7%E5%93%81%E6%96%B0%E5%A2%9E%E5%88%97%E7%9A%84%E9%9C%80%E6%B1%82%EF%BC%8C%E5%B8%8C%E6%9C%9B%E6%B2%A1%E6%9C%89%E6%B5%AA%E8%B4%B9%E4%BD%A0%E8%BF%99%E4%B8%80%E5%88%86%E9%92%9F%EF%BC%8C%E6%9C%89%E6%94%B6%E8%8E%B7%E5%B0%B1%E5%A5%BD%E3%80%82) [[7]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%E4%BB%A5%E4%B8%8A%E9%9D%A2%E7%9A%84%E7%94%A8%E6%88%B7%E8%A1%A8%E4%B8%BA%E4%BE%8B%EF%BC%8C%E5%8F%AF%E4%BB%A5%E8%AE%BE%E8%AE%A1%E4%B8%BA) [[8]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%E4%BC%98%E7%82%B9%EF%BC%9A) [[9]](https://blog.csdn.net/qijiqiguai/article/details/78658229#:~:text=%EF%BC%881%EF%BC%89%20ext%E9%87%8C%E7%9A%84%E5%AD%97%E6%AE%B5%E6%97%A0%E6%B3%95%E5%BB%BA%E7%AB%8B%E7%B4%A2%E5%BC%95) [架构师之路] 高可扩展表结构系列-CSDN博客

<https://blog.csdn.net/qijiqiguai/article/details/78658229>

[[10]](https://blazor.syncfusion.com/documentation/dashboard-layout/interaction-with-panels/dragging-moving-of-panels#:~:text=The%20Dashboard%20Layout%20component%20is,position%20without%20disturbing%20the%20layout) Drag and Drop in Blazor Dashboard Layout Component | Syncfusion

<https://blazor.syncfusion.com/documentation/dashboard-layout/interaction-with-panels/dragging-moving-of-panels>

[[11]](https://www.reddit.com/r/sysadmin/comments/azflsc/create_rdp_file_with_encrypted_password_in_it/#:~:text=There%20are%20two%20ways%20you,to%20the%20windows%20credentials%20manager) Create rdp file with encrypted password in it : r/sysadmin

<https://www.reddit.com/r/sysadmin/comments/azflsc/create_rdp_file_with_encrypted_password_in_it/>

[[12]](https://learn.microsoft.com/zh-tw/aspnet/core/blazor/globalization-localization?view=aspnetcore-9.0#:~:text=%E5%85%A8%E7%90%83%E5%8C%96%E5%92%8C%E7%95%B6%E5%9C%B0%E8%AA%9E%E7%B3%BB%E5%8C%96) ASP.NET Core Blazor 全球化和當地語系化 | Microsoft Learn

<https://learn.microsoft.com/zh-tw/aspnet/core/blazor/globalization-localization?view=aspnetcore-9.0>