

cafeMuji 注文管理システム API仕様書

概要

cafeMuji注文管理システムのAPI仕様を定義します。現在はDjangoテンプレートベースのシステムですが、将来的なAPI化を見据えた設計仕様です。

API設計方針

設計原則

- **RESTful API**: REST原則に従った設計
- **JSON形式**: データ交換はJSON形式
- **HTTPステータスコード**: 適切なHTTPステータスコードの使用
- **認証・認可**: セキュアなAPIアクセス制御
- **バージョンング**: APIバージョンの管理

技術スタック

- **フレームワーク**: Django REST Framework
- **認証**: JWT (JSON Web Token)
- **データ形式**: JSON
- **ドキュメント**: OpenAPI 3.0 (Swagger)

認証・認可

JWT認証

```
{
  "token_type": "Bearer",
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
  "refresh_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
  "expires_in": 3600
}
```

認証エンドポイント

POST /api/auth/login/	# ログイン
POST /api/auth/logout/	# ログアウト
POST /api/auth/refresh/	# トークン更新
POST /api/auth/register/	# ユーザー登録

データモデルAPI

1. フード注文API

1.1 注文一覧取得

GET /api/food/orders/
Authorization: Bearer {token}

レスポンス例

```
{
  "count": 25,
  "next": "http://api.example.com/api/food/orders/?page=2",
  "previous": null,
  "results": [
    {
      "id": 1,
      "menu": "からあげ丼",
      "quantity": 2,
      "eat_in": true,
      "clip_color": "yellow",
      "clip_number": 1,
      "group_id": "group_001",
      "status": "ok",
      "is_completed": false,
      "timestamp": "2025-08-18T14:30:00Z",
      "completed_at": null,
      "note": "特急をお願いします"
    }
  ]
}
```

1.2 注文作成

POST /api/food/orders/
Authorization: Bearer {token}
Content-Type: application/json

```
{
  "menu": "からあげ丼",
  "quantity": 2,
  "eat_in": true,
  "clip_color": "yellow",
  "clip_number": 1,
  "group_id": "group_001",
  "note": "特急をお願いします"
}
```

1.3 注文更新

PUT /api/food/orders/{id}/
Authorization: Bearer {token}
Content-Type: application/json

```
{
  "status": "completed",
  "is_completed": true
}
```

1.4 注文削除

DELETE /api/food/orders/{id}/
Authorization: Bearer {token}

2. アイスクリーム注文API

2.1 注文一覧取得

GET /api/ice/orders/
Authorization: Bearer {token}

レスポンス例

```
{
  "count": 15,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "group_id": "group_001",
      "size": "W",
      "container": "cone",
      "flavor1": "jersey",
      "flavor2": "mango",
      "is_completed": false,
      "timestamp": "2025-08-18T14:30:00Z",
      "status": "ok",
      "clip_color": "white",
      "clip_number": 2,
      "completed_at": null,
      "is_auto_stopped": false,
      "note": "ダブルをお願いします",
      "is_pudding": false
    }
  ]
}
```

2.2 注文作成

POST /api/ice/orders/
Authorization: Bearer {token}
Content-Type: application/json

```
{
  "group_id": "group_001",
  "size": "W",
  "container": "cone",
  "flavor1": "jersey",
  "flavor2": "mango",
  "clip_color": "white",
  "clip_number": 2,
  "note": "ダブルをお願いします"
}
```

3. かき氷注文API

3.1 注文一覧取得

GET /api/shavedice/orders/
Authorization: Bearer {token}

レスポンス例

```
{
  "count": 8,
  "next": null,
```

```

"previous": null,
"results": [
  {
    "id": 1,
    "flavor": "抹茶",
    "group_id": "group_001",
    "is_completed": false,
    "clip_color": "yellow",
    "clip_number": 3,
    "timestamp": "2025-08-18T14:30:00Z",
    "completed_at": null,
    "status": "ok",
    "is_auto_stopped": false,
    "note": "甘さ控えめで"
  }
]
}

```

検索・フィルタリングAPI

1. 高度な検索

GET /api/food/orders/?menu=からあげ丼&status=ok&clip_color=yellow&date_from=2025-08-01&date_to=2025-08-18

クエリパラメータ - menu: メニュー名でフィルタ - status: 注文状態でフィルタ - clip_color: クリップ色でフィルタ - clip_number: クリップ番号でフィルタ - date_from: 開始日 - date_to: 終了日 - is_completed: 完了状態 - eat_in: 店内/テイクアウト

2. 統計情報API

GET /api/statistics/orders/
Authorization: Bearer {token}

レスポンス例

```

{
  "total_orders": 48,
  "completed_orders": 35,
  "pending_orders": 13,
  "today_orders": 12,
  "menu_statistics": {
    "からあげ丼": 25,
    "ルーロー飯": 23
  },
  "flavor_statistics": {
    "jersey": 18,
    "mango": 12,
    "ocha": 8
  },
  "average_completion_time": "8.5"
}

```

モバイルAPI

1. 簡易注文API

POST /api/mobile/quick-order/
Content-Type: application/json

```
{
```

```

    "order_type": "food",
    "menu": "からあげ丼",
    "quantity": 1,
    "eat_in": true
  }

```

2. 注文状況確認API

GET /api/mobile/order-status/{group_id}/

レスポンス例

```

{
  "group_id": "group_001",
  "status": "preparing",
  "estimated_completion": "2025-08-18T15:00:00Z",
  "orders": [
    {
      "type": "food",
      "menu": "からあげ丼",
      "status": "preparing"
    },
    {
      "type": "ice",
      "flavor": "jersey",
      "status": "waiting"
    }
  ]
}

```

リアルタイム通信API

WebSocket接続

```

// WebSocket接続例
const socket = new WebSocket('ws://api.example.com/ws/orders/');

socket.onmessage = function(event) {
  const data = JSON.parse(event.data);

  if (data.type === 'order_update') {
    updateOrderDisplay(data.order);
  } else if (data.type === 'new_order') {
    addNewOrder(data.order);
  }
};

```

WebSocketメッセージ形式

```

{
  "type": "order_update",
  "order": {
    "id": 1,
    "status": "completed",
    "completed_at": "2025-08-18T15:00:00Z"
  }
}

```

レポート・分析API

1. 売上レポート

```
GET /api/reports/sales/?date_from=2025-08-01&date_to=2025-08-18
Authorization: Bearer {token}
```

レスポンス例

```
{
  "period": "2025-08-01 to 2025-08-18",
  "total_sales": 125000,
  "total_orders": 156,
  "average_order_value": 801.28,
  "daily_sales": [
    {
      "date": "2025-08-01",
      "sales": 8500,
      "orders": 12
    }
  ],
  "menu_performance": [
    {
      "menu": "からあげ丼",
      "quantity": 45,
      "revenue": 45000
    }
  ]
}
```

2. 在庫レポート

```
GET /api/reports/inventory/
Authorization: Bearer {token}
```

3. 顧客分析

```
GET /api/reports/customer-analytics/
Authorization: Bearer {token}
```

セキュリティAPI

1. 権限管理

```
GET /api/auth/permissions/
Authorization: Bearer {token}
```

レスポンス例

```
{
  "user_id": 1,
  "username": "admin",
  "permissions": [
    "food.add_foodorder",
    "food.change_foodorder",
    "food.delete_foodorder",
    "ice.add_order",
    "ice.change_order",
    "ice.delete_order"
  ],
  "groups": ["admin", "kitchen_staff"]
}
```

2. アクセスログ

GET /api/auth/access-logs/
Authorization: Bearer {token}



エラーハンドリング

エラーレスポンス形式

```
{
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "入力データが無効です",
    "details": {
      "menu": ["このフィールドは必須です"],
      "quantity": ["1以上の値を入力してください"]
    }
  },
  "timestamp": "2025-08-18T14:30:00Z",
  "request_id": "req_123456789"
}
```

HTTPステータスコード

- 200: 成功
- 201: 作成成功
- 400: バリデーションエラー
- 401: 認証エラー
- 403: 権限エラー
- 404: リソースが見つからない
- 500: サーバー内部エラー



開発・テスト用API

1. テストデータ作成

POST /api/dev/create-test-data/
Authorization: Bearer {admin_token}
Content-Type: application/json

```
{
  "order_count": 10,
  "include_completed": true
}
```

2. システム状態確認

GET /api/health/

レスポンス例

```
{
  "status": "healthy",
  "timestamp": "2025-08-18T14:30:00Z",
  "version": "1.0.0",
  "database": "connected",
  "cache": "connected",
}
```

```
"services": {  
  "food_service": "running",  
  "ice_service": "running",  
  "shavedice_service": "running"  
}
```

APIドキュメント

Swagger UI

GET /api/docs/

ReDoc

GET /api/redoc/

OpenAPI仕様

GET /api/schema/

実装例

Django REST Framework実装

```
# views.py  
from rest_framework import viewsets  
from rest_framework.permissions import IsAuthenticated  
from .models import FoodOrder  
from .serializers import FoodOrderSerializer  
  
class FoodOrderViewSet(viewsets.ModelViewSet):  
    queryset = FoodOrder.objects.all()  
    serializer_class = FoodOrderSerializer  
    permission_classes = [IsAuthenticated]  
  
    def get_queryset(self):  
        queryset = FoodOrder.objects.all()  
        menu = self.request.query_params.get('menu', None)  
        if menu is not None:  
            queryset = queryset.filter(menu=menu)  
        return queryset
```

シリアライザー

```
# serializers.py  
from rest_framework import serializers  
from .models import FoodOrder  
  
class FoodOrderSerializer(serializers.ModelSerializer):  
    class Meta:  
        model = FoodOrder  
        fields = '__all__'  
        read_only_fields = ('id', 'timestamp', 'completed_at')
```

パフォーマンス最適化

1. ページネーション

```
{
  "count": 1000,
  "next": "http://api.example.com/api/food/orders/?page=2",
  "previous": null,
  "results": [...]
}
```

2. キャッシュ

```
from django.utils.decorators import method_decorator
from django.views.decorators.cache import cache_page

@method_decorator(cache_page(60 * 15)) # 15分間キャッシュ
def list(self, request):
    # 注文一覧の取得
    pass
```

3. データベース最適化

```
# select_relatedとprefetch_relatedの使用
queryset = FoodOrder.objects.select_related('user').prefetch_related('items')
```

今後の拡張

1. GraphQL対応

```
query {
  foodOrders {
    id
    menu
    quantity
    user {
      username
      email
    }
    items {
      name
      price
    }
  }
}
```

2. gRPC対応

```
service OrderService {
  rpc CreateOrder(CreateOrderRequest) returns (OrderResponse);
  rpc GetOrder(GetOrderRequest) returns (OrderResponse);
  rpc UpdateOrder(UpdateOrderRequest) returns (OrderResponse);
  rpc DeleteOrder(DeleteOrderRequest) returns (DeleteOrderResponse);
}
```

3. マイクロサービス化

- 注文サービス: 注文の作成・管理
- 在庫サービス: 在庫の管理・更新
- 決済サービス: 決済処理

- **通知サービス:** 通知・メール送信

作成日: 2025年8月 **作成者:** 村岡 優次郎 **バージョン:** 1.0