API 接口文档 · Emerge 电商 Demo（GraphQL）

本 Demo 的前台与后台均通过 GraphQL 与后端交互。本文给出常用操作与 cURL 示例，便于快速集成与演示。

- 本地 GraphQL Endpoint：`http://localhost:8001/graphql/`
- 生产/测试环境请替换为相应地址；Dashboard 登录设置也需指向该地址。

> 提示：支持 GraphQL Introspection，可使用 Insomnia/Altair/GraphiQL 等工具进行探索。

0. 约定与通用能力

- 身份认证：JWT（`Authorization: JWT <token>`），员工与顾客使用同一端点
- 多渠道：多数查询/变更需要 `channel` 参数（如 `default-channel`）
- 分页：Relay Connection 规范，`first/after` + `pageInfo{hasNextPage,endCursor}`
- 过滤/排序：使用 `filter`/`sortBy` 输入对象
- 错误：业务错误在 `data.*.errors[]`，系统错误返回 5xx
- 文件/图片：`/thumbnail/<globalId>/<size>/<format?>` 将 302 到真实媒体 URL

1. 认证（登录/获取 Token）

请求：

```
curl -sS -H 'Content-Type: application/json' \
  --data '{
    "query": "mutation($email:String!,$password:String!){
tokenCreate(email:$email,password:$password){ errors{field message code}
token user{email isStaff}} }",
    "variables": {"email": "admin@gmail.com", "password": "<你的密码>"}
  }' \
  http://localhost:8001/graphql/
```

响应返回 `data.tokenCreate.token`（JWT）。后续请求在 Header 中加入：

```
Authorization: JWT <token>
```

1.1 Token 校验与刷新

```
mutation Verify($token:String!){ tokenVerify(token:$token){ isValid
payload errors{field message code} } }
mutation Refresh($token:String!){ tokenRefresh(csrfToken:"",
refreshToken:$token){ token user{ email } errors{ field message } } }
```

1.2 当前用户信息（顾客/员工通用）

```
query Me{ me{ email firstName lastName addresses{ id country area } } }
```

1.3 注册与重置密码（顾客）

```
mutation Register($email:String!){ accountRegister(input:{email:$email,
channel:"default-channel"}){ errors{field message code} } }
mutation RequestReset($email:String!){ requestPasswordReset(email:$email,
channel:"default-channel"){ errors{field message code} } }
```

2. 查询商品列表（分页/筛选）

```
query Products($first:Int=12,$after:String,$channel:String="default-
channel"){
  products(first:$first, after:$after, channel:$channel){
    pageInfo{ hasNextPage endCursor }
    edges{ node{ id name slug category{ name } thumbnail(size:1024,
format:WEBP){ url alt } } }
  }
}
```

cURL：

```
curl -sS -H 'Content-Type: application/json' \
 --data '{"query":"query{ products(first:6, channel: \"default-channel\"){
edges{ node{ name slug thumbnail{url} } } } }"}' \
 http://localhost:8001/graphql/
```

2.1 过滤与排序（示例）

```
query ProductsFiltered{
  products(first:12, channel:"default-channel", filter:{ search:"tee" },
sortBy:{field:NAME, direction:ASC}){
    edges{ node{ name slug category{ name } } }
  }
}
```

3. 查询商品详情

```
query Product($slug:String!,$channel:String="default-channel"){
  product(slug:$slug, channel:$channel){
    id name description seoTitle seoDescription
    media{ url }
    variants{ id name sku quantityAvailable pricing{ price{ gross{ amount
currency } } } }
    thumbnail(size:1024, format:WEBP){ url alt }
  }
}
```

4. 购物车 / 结算（Checkout）

4.1 创建 Checkout

```
mutation CreateCheckout($channel:String!, $email:String!, $lines:
[CheckoutLineInput!]!){
  checkoutCreate(input:{ channel:$channel, email:$email, lines:$lines }){
    checkout{ id token }
    errors{ field message code }
  }
}
```

变量示例：

```
{
  "channel": "default-channel",
  "email": "demo@example.com",
  "lines": [{"quantity": 1, "variantId": "<Variant ID>"}]
}
```

4.2 增加/更新行项目

```
mutation AddLines($checkoutId:ID!, $lines:[CheckoutLineInput!]!){
  checkoutLinesAdd(id:$checkoutId, lines:$lines){
    checkout{ id totalPrice{ gross{ amount currency } } }
    errors{ field message code }
  }
}
```

4.3 设置地址与配送

```
mutation SetAddresses($id:ID!, $shipping:AddressInput!,
$billing:AddressInput!){
```

```
  checkoutShippingAddressUpdate(id:$id, shippingAddress:$shipping){
errors{field message} }
  checkoutBillingAddressUpdate(id:$id, billingAddress:$billing){
errors{field message} }
}

mutation SetDelivery($id:ID!, $method:ID!){
  checkoutDeliveryMethodUpdate(id:$id, deliveryMethodId:$method){
errors{field message code} }
}
```

> 配送方式可通过 `checkoutDeliveryMethods` 查询（依赖渠道与运费配置）。

4.4 支付与提交订单

- Demo 默认未接入真实网关，可选择：
  - 启用“手动支付/模拟网关”完成下单；
  - 或接入 Stripe/Adyen 沙箱，使用 `paymentInitialize / paymentGatewayInitialize` 等能力。

提交订单：

```
mutation Complete($id:ID!){
  checkoutComplete(id:$id){ order{ id number status } errors{ field
message code } }
}
```

5. 订单查询

```
query Orders($first:Int=10){
  orders(first:$first, sortBy:{ field:CREATED_AT, direction:DESC }){
    edges{ node{ id number status total{ gross{ amount currency } }
userEmail } }
  }
}
```

> 需要员工权限（Authorization 头携带管理员/员工 Token）。

6. 错误与返回约定

- GraphQL 层：`errors[{field,message,code}]` 与 HTTP 200 并存；业务错误查看 `errors`，系统错误返回 5xx
- 权限：无权限返回 `GraphQLError: Permission denied`，或 `errors.code` 为 `GRAPHQL_ERROR/INVALID` 等

6.1 常见错误码

- `GRAPHQL_ERROR`：请求格式或字段错误
- `INVALID`：参数校验失败
- `REQUIRED`：缺少必填字段
- `NOT_FOUND`：目标不存在
- `UNIQUE`：唯一性冲突
- `PERMISSION_DENIED`：无权限

7. 代码生成与类型安全（可选）

项目已集成 GraphQL Code Generator（前台）。当新增/修改 `src/graphql/*.graphql` 时，执行：

```
pnpm codegen
```

将生成严格类型的请求与结果（TypedDocumentString），减少手写类型成本。

8. Webhook（只读概览）

- 事件示例：`ORDER_CREATED`、`PAYMENT_CAPTURED`、`CHECKOUT_CREATED`
- 签名校验：建议在回调端验证 Secret 与时间戳，避免重放
- 详见 `docs/Webhooks.md`

9. 环境与安全建议

- 生产环境使用 HTTPS；将 `ALLOWED_HOSTS`、`SECRET_KEY` 等安全变量妥善配置
- 图片/媒体建议迁移至对象存储（S3 等）；开启 CDN
- 开启网关级限流、WAF 与审计；使用独立的缓存与队列服务

10. 集成示例（Node.js fetch）

```
async function gql(query: string, variables?: any, token?: string) {
  const res = await fetch("http://localhost:8001/graphql/", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      ...(token ? { Authorization: `JWT ${token}` } : {}),
    },
    body: JSON.stringify({ query, variables }),
  });
  const json = await res.json();
  if (json.errors) throw new Error(JSON.stringify(json.errors));
  return json.data;
}
```