Advanced Email & Phone Validation API

A professional API for advanced email address and phone number validation, built with FastAPI.

🚀 Features

Email Validation

- **RFC Format Check**: Verifies email format compliance
- **DNS Verification**: Checks domain existence
- **MX Records**: Verifies MX records for email delivery
- **Disposable Email Detection**: Identifies temporary/throwaway email services
- **Provider Identification**: Recognizes major email providers
- **Confidence Score**: Numerical score indicating email validity

Phone Validation

- **International Format**: Conversion to E.164 format
- **Geographic Information**: Country and region of the number
- **Line Type**: Mobile, landline, VoIP, toll-free, etc.
- **Carrier Information**: Identifies the carrier when available
- **Timezone**: Time zones associated with the number
- **Flexible Parsing**: Supports numbers with or without international prefix

📋 Requirements

```bash
fastapi
uvicorn
pydantic
phonenumbers
email-validator
dnspython
requests
```

🛠️ Installation

1. Clone the repository:

```bash
```

```
git clone <repository-url>
cd email-phone-validation-api
```

2. Install dependencies:

```bash
pip install -r requirements.txt
```

3. Start the server:

```bash
python main.py
```

The API will be available at `http://localhost:8001`

📚 API Documentation

Main Endpoints

`POST /api/validate/email`

Validates an email address with advanced checks.

Request Body:

```json
{
  "email": "user@example.com"
}
```

Response:

```json
```

```
{
  "valid": true,
  "disposable": false,
  "domain_exists": true,
  "mx_found": true,
  "provider": "Gmail",
  "suggestion": null,
  "confidence_score": 1.0,
  "details": {
    "normalized_email": "user@example.com",
    "domain": "example.com",
    "checks_performed": ["format", "dns", "mx", "disposable", "provider"]
  }
}
```


POST /api/validate/phone

Validates a phone number with detailed information.

Request Body:


```
json

{
  "phone": "+393331234567"
}
```


Response:


```
json

{
  "valid": true,
  "international_format": "+393331234567",
  "country": "Italy",
  "country_code": "IT",
  "type": "mobile",
  "carrier": "Vodafone",
  "line_type": "mobile",
  "timezone": ["Europe/Rome"],
  "confidence_score": 1.0
}
```


GET /

General API information.

`GET /api/health`

API health check.

🔧 Configuration

CORS

The API is configured to accept requests from any origin. For production environments, modify the CORS configuration in `main.py`:

```python
app.add_middleware(
    CORSMiddleware,
    allow_origins=["https://yourdomain.com"],  # Specify authorized domains
    allow_credentials=True,
    allow_methods=["GET", "POST"],
    allow_headers=["*"],
)
```

Logging

The logging system is configured at INFO level. To modify the level:

```python
logging.basicConfig(level=logging.DEBUG)  # For more detailed debugging
```

📊 Confidence Scores

Email

- **1.0**: Valid email with existing domain, MX records, recognized provider
- **0.8-0.9**: Valid email with some missing checks
- **0.6-0.7**: Valid format but domain issues
- **0.0-0.5**: Invalid or suspicious email

Phone

- **1.0**: Valid number with complete information (carrier, location)
- **0.7-0.9**: Valid number with partial information
- **0.0**: Invalid number

🌐 Supported Email Providers

The API automatically recognizes the following providers:

- Gmail / Google Mail
- Outlook / Hotmail / Microsoft Live
- Yahoo (including national domains)
- ProtonMail
- iCloud / me.com / mac.com
- Italian providers: Libero, Tiscali, Alice, Virgilio, TIN

🚫 Disposable Email Detection

The API identifies temporary email services such as:

- 10minutemail.com
- guerrillamail.com
- mailinator.com
- tempmail.org
- yopmail.com
- And many more...

🗺️ International Support

Phone Numbers

- Support for all countries worldwide
- Automatic international prefix parsing
- Fallback for numbers without prefix (US/IT)
- Carrier and timezone information

📝 Usage Examples

cURL

Email Validation:

```bash
curl -X POST "http://localhost:8001/api/validate/email" \
    -H "Content-Type: application/json" \
    -d '{"email": "test@gmail.com"}'
```

Phone Validation:

bash

```bash
curl -X POST "http://localhost:8001/api/validate/phone" \
    -H "Content-Type: application/json" \
    -d '{"phone": "+393331234567"}'
```

Python

python

```python
import requests

# Email validation
response = requests.post(
    "http://localhost:8001/api/validate/email",
    json={"email": "user@example.com"}
)
result = response.json()
print(f"Email valid: {result['valid']}")

# Phone validation
response = requests.post(
    "http://localhost:8001/api/validate/phone",
    json={"phone": "+393331234567"}
)
result = response.json()
print(f"Phone valid: {result['valid']}")
```

JavaScript

javascript

```javascript
// Email validation
fetch('http://localhost:8001/api/validate/email', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({email: 'user@example.com'})
})
.then(response => response.json())
.then(data => console.log('Email result:', data));

// Phone validation
fetch('http://localhost:8001/api/validate/phone', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({phone: '+393331234567'})
})
.then(response => response.json())
.then(data => console.log('Phone result:', data));
```

🔮 Future Features

The code includes structures for future implementations:

- Integration with external APIs for advanced email validation
- Typo correction suggestions for email addresses
- Automatic updates of disposable domains list
- Caching for improved performance

🚀 Deployment

Docker

```dockerfile
```

```
FROM python:3.9-slim

WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt

COPY . .
EXPOSE 8001

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8001"]
```

Production

For production environments, consider:

- Using an ASGI server like Gunicorn with Uvicorn workers
- Implementing rate limiting
- Setting up a reverse proxy (Nginx)
- Structured monitoring and logging
- Redis caching for improved performance

📖 Interactive Documentation

Once the API is running, visit:

- **Swagger UI**: `http://localhost:8001/docs`
- **ReDoc**: `http://localhost:8001/redoc`

🤝 Contributing

Contributions are welcome! Please:

1. Fork the project
2. Create a feature branch
3. Commit your changes
4. Push the branch
5. Open a Pull Request

📄 License

This project is released under the MIT License.

📞 Support

For questions or support, please open an issue in the GitHub repository.