/**
 * perit.hu
 */

MicroDMS

USER MANUAL

1.0

Document history

Version	Date	Modified by	Modification
1.0	2025-02-02	Peter Nagy	First version

Table of Content

1. What is this **5**
1.1. Main characteristics *5*
1.2. Technologies *6*
2. System configuration **7**
2.1. The storage subsystem *7*
2.2. Keywords *8*
2.3. Document types *11*
2.4. Document type groups *12*
2.5. User groups *13*
3. MicroDMS API **15**
3.1. Authentication *15*
3.1.1. authenticate() 15
3.2. Config *15*
3.2.1. getConfig() 15
3.2.2. applyConfig() 16
3.3. System *16*
3.3.1. getDocumentTypes() 16
3.3.2. getDocumentTypeGroups() 17
3.3.3. getKeywordTypes() 17
3.4. Files *18*
3.4.1. upload() 18
3.5. Documents *18*
3.5.1. createDocument() 18
3.5.2. readDocument() 20
3.5.3. updateDocument() 20
3.5.4. deleteDocument 21
3.5.5. getDocumentKeywords() 21
3.5.6. getDocumentInfo() 22
3.5.7. queryDocument() 22
4. MicroDMS Client **28**
4.1. The Retrieval page *28*
4.2. The Import page *34*
5. Performance and volume measures **35**

6. Appendix **39**

6.1. Referenced Documents *39*

6.2. List of Figures *39*

6.3. Open Issues *39*

6.4. Requirement Keys *39*

1. WHAT IS THIS

The MicroDMS system is a document archive for storing and retrieving documents. It was inspired by Saperion and OnBase but does not want to compete with any of them. Both archives have many more features. The MicroDMS system focuses on the basic functionality of a document archive, which is storing, reading, updateing and deleting of documents.

However the system will be prepared for GDPR removal and recertification of digitally signed documents through external tools to provide a complete solution for various businesses.

1.1. MAIN CHARACTERISTICS

- Windows and Linux systems are supported.
- Various deployments are possible, starting from a Window service through Docker until Kubernetes or Openshift.
- Can be hosted in an on-premise infrastructure or in the cloud.
- The meta-data is stored in a database. Currently supported Microsoft SQL-Server and PostgreSQL.
- Multivalue keywords are supported.
- The following data types are supported:
    - INTEGER
    - LONG
    - ALPHANUMERIC
    - CURRENCY
    - SPECIFIC_CURRENCY
    - DATE
    - DATETIME
    - FLOATING_POINT
    - BOOLEAN
- Files can be stored in local drives or on SMB shares.
- The configuration of the archive system is stored in the database. The current configuration can be downloaded into a file in yaml format, or can be applied from a file.

1.2. TECHNOLOGIES

- Spring Boot
- Java17
- Sql-Server, PostgreSQL
- Ngface
- Spvitamin

2. SYSTEM CONFIGURATION

2.1. THE STORAGE SUBSYSTEM

The following model is used:

Media: a particular physical storage medium. It can be e.g. a local drive, an SMB share or later a cloud storage. In this entity we store the path of the physical medium and the data needed to access it, e.g. username and password. Also the maximum size of the volume will be stored here.

Volume: files are stored in volumes on the media. The primary goal of having volumes is the ability of making backups easily. A volume is filled only until the maximum size is reached. In case the volume reached its maximum size, a new volume will be created.

Storage: this is a logical entity, which references a particular media. The referenced media within the storage may change with the time. For example, if the currently used media gets full, we can assign another media to the storage, and archiving of documents can be continued using the same storage.

This is a sample configuration, which configures two media and a storage for each.

```
#-------------------------------------------------------------------------
media:
#-------------------------------------------------------------------------
  # LOCAL_DRIVE
  - kind: Media
    apiVersion: v1
    name: Z-drive
    mediaType: LOCAL_DRIVE
    connection:
      type: LOCAL_DRIVE
      path: z:\filestorage\microdms
    spaceLimit: 1 GB
  # SMB
  - kind: Media
    apiVersion: v1
    name: NETAPP
    mediaType: SMB
    connection:
      type: SMB
      host: smb://perit
      share: smb-test
      username: smb_test
      # alma
      password: ntJ9hrf4awc=
      domain: perit
    spaceLimit: 1 GB
#-------------------------------------------------------------------------
storages:
#-------------------------------------------------------------------------
  # PERF_TEST_STORAGE
  - kind: Storage
    apiVersion: v1
    name: PERF_TEST_STORAGE
    media: Z-drive
  # LONG_TERM_STORAGE
  - kind: Storage
    apiVersion: v1
    name: LONG_TERM_STORAGE
    media: NETAPP
```

2.2. KEYWORDS

Sample configuration:

```
#------------------------------------------------------------------------
keywords:
#------------------------------------------------------------------------
  # Alphanumeric
  - kind: Keyword
    apiVersion: v1
    name: Alphanumeric
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_UPPERCASE
    length: 50
  # Numeric9
  - kind: Keyword
    apiVersion: v1
    name: Integer
    dataType: INTEGER
  # Numeric20
  - kind: Keyword
    apiVersion: v1
    name: Long
    dataType: LONG
  # Date
  - kind: Keyword
    apiVersion: v1
    name: Date
    dataType: DATE
  # DateTime
  - kind: Keyword
    apiVersion: v1
    name: DateTime
    dataType: DATETIME
  # FloatingPoint
  - kind: Keyword
    apiVersion: v1
    name: FloatingPoint
    dataType: FLOATING_POINT
  # Boolean
  - kind: Keyword
    apiVersion: v1
    name: Boolean
    dataType: BOOLEAN
  # FirstName
  - kind: Keyword
```

```
    apiVersion: v1
    name: FirstName
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_MIXEDCASE
    length: 30
  # LastName
  - kind: Keyword
    apiVersion: v1
    name: LastName
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_MIXEDCASE
    length: 30
  # Birthdate
  - kind: Keyword
    apiVersion: v1
    name: Birthdate
    dataType: DATE
  # Address
  - kind: Keyword
    apiVersion: v1
    name: Address
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_MIXEDCASE
    length: 100
  # Salery
  - kind: Keyword
    apiVersion: v1
    name: Salery
    dataType: INTEGER
  # InvoiceNumber
  - kind: Keyword
    apiVersion: v1
    name: InvoiceNumber
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_MIXEDCASE
    length: 20
  # InvoiceDate
  - kind: Keyword
    apiVersion: v1
    name: InvoiceDate
    dataType: DATE
  # DueDate
  - kind: Keyword
    apiVersion: v1
    name: DueDate
    dataType: DATE
  # Currency
  - kind: Keyword
    apiVersion: v1
    name: Currency
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_UPPERCASE
    length: 3
  # IssuerName
  - kind: Keyword
    apiVersion: v1
    name: IssuerName
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_MIXEDCASE
    length: 100
  # IssuerAddress
  - kind: Keyword
    apiVersion: v1
    name: IssuerAddress
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_MIXEDCASE
    length: 100
  # IssuerContactPhone
  - kind: Keyword
    apiVersion: v1
    name: IssuerContactPhone
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_UPPERCASE
    length: 20
  # IssuerVATNumber
  - kind: Keyword
    apiVersion: v1
    name: IssuerVATNumber
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_UPPERCASE
    length: 20
  # IssuerAccountNumber
  - kind: Keyword
    apiVersion: v1
    name: IssuerAccountNumber
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_UPPERCASE
    length: 40
  # CustomerName
  - kind: Keyword
    apiVersion: v1
    name: CustomerName
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_MIXEDCASE
    length: 100
  # CustomerAddress
  - kind: Keyword
    apiVersion: v1
    name: CustomerAddress
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_MIXEDCASE
    length: 100
  # CustomerContactPhone
  - kind: Keyword
    apiVersion: v1
    name: CustomerContactPhone
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_UPPERCASE
    length: 20
  # CustomerVATNumber
  - kind: Keyword
    apiVersion: v1
    name: CustomerVATNumber
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_UPPERCASE
    length: 20
  # InvoiceDescription
  - kind: Keyword
```

```
    apiVersion: v1
    name: InvoiceDescription
    dataType: ALPHANUMERIC
    dataTypeOption: ALPHANUMERIC_MIXEDCASE
    length: 100
  # NetPrice
  - kind: Keyword
    apiVersion: v1
    name: NetPrice
    dataType: FLOATING_POINT
  # Tax
  - kind: Keyword
    apiVersion: v1
    name: Tax
    dataType: FLOATING_POINT
  # TotalAmount
  - kind: Keyword
    apiVersion: v1
    name: TotalAmount
    dataType: FLOATING_POINT
```

2.3. DOCUMENT TYPES

Here we can configure document types and assign keywords to them. Keywords may have the following flags: READONLY, HIDDEN, NOTNULL.

```
#-------------------------------------------------------------------------
documentTypes:
#-------------------------------------------------------------------------
  # KeywordTest
  - kind: DocumentType
    apiVersion: v1
    name: KeywordTest
    storage: LONG_TERM_STORAGE
    keywords:
      - name: Alphanumeric
      - name: Integer
      - name: Long
      - name: Date
      - name: DateTime
      - name: FloatingPoint
      - name: Boolean
      - name: AGE
        flags:
          - READONLY
          - HIDDEN
  # PerformanceTest
  - kind: DocumentType
    apiVersion: v1
    name: PerformanceTest
    storage: PERF_TEST_STORAGE
    keywords:
      - name: Long
      - name: Alphanumeric
      - name: FloatingPoint
      - name: DateTime
      - name: Date
  # Employee
  - kind: DocumentType
    apiVersion: v1
    name: Employee
    storage: LONG_TERM_STORAGE
    keywords:
      - name: FirstName
      - name: LastName
      - name: Birthdate
      - name: Address
      - name: Salery
  # Invoice
  - kind: DocumentType
    apiVersion: v1
    name: Invoice
    storage: LONG_TERM_STORAGE
    keywords:
      - name: InvoiceNumber
      - name: InvoiceDescription
      - name: InvoiceDate
      - name: DueDate
      - name: IssuerName
      - name: IssuerAddress
      - name: IssuerContactPhone
      - name: IssuerVATNumber
      - name: IssuerAccountNumber
      - name: CustomerName
      - name: CustomerAddress
      - name: CustomerContactPhone
      - name: CustomerVATNumber
      - name: Currency
      - name: NetPrice
      - name: Tax
      - name: TotalAmount
```

2.4. DOCUMENT TYPE GROUPS

```
#-------------------------------------------------------------------------
documentTypeGroups:
#-------------------------------------------------------------------------
  # Test
  - kind: DocumentTypeGroup
    apiVersion: v1
    name: Test
    documentTypes:
      - KeywordTest
      - PerformanceTest
  # HR Documents
  - kind: DocumentTypeGroup
    apiVersion: v1
    name: HR Documents
    documentTypes:
      - Employee
  # FIN Documents
  - kind: DocumentTypeGroup
```

```
    apiVersion: v1
    name: FIN Documents
    documentTypes:
      - Invoice
```

2.5. USER GROUPS

User groups control access to document types.

```
#-------------------------------------------------------------------------
usergroups:
#-------------------------------------------------------------------------
  - kind: UserGroup
    apiVersion: v1
    name: MANAGER
    documentTypes:
      - KeywordTest
      - PerformanceTest
      - Employee
      - Invoice
  - kind: UserGroup
    apiVersion: v1
    name: HR
    documentTypes:
      - Employee
  - kind: UserGroup
    apiVersion: v1
    name: FIN
    documentTypes:
      - Invoice
```

Configuration of local users or AD group assignments can be made in the usual way on the application.yaml.

```
#-------------------------------------------------------------------------------------
# Local users for test reasons
#-------------------------------------------------------------------------------------
# Please set either password or encrypted-password property
localuser:
  admin.encrypted-password: ij3HfoF+WYo=
  user.password: user
  manager.password: manager
  hr.password: hr
  fin.password: fin


#-------------------------------------------------------------------------------------
# AD group -> role mapping (all = '*')
#-------------------------------------------------------------------------------------
roles:
  # Site admin
  ROLE_ADMIN:
    #groups: any AD group
    users: nagy_peter,admin
    includes: ROLE_USER
```

```
# User
ROLE_USER:
  #groups: any AD group
  users: nagy_peter,user

# MANAGER
ROLE_MANAGER:
  #groups: any AD group
  users: manager
  includes: ROLE_USER

# HR
ROLE_HR:
  #groups: any AD group
  users: hr
  includes: ROLE_USER

# FIN
ROLE_FIN:
  #groups: any AD group
  users: fin
  includes: ROLE_USER
```

3. MICRODMS API

This is the backend interface to provide access to the stored documents for other backend systems. The REST API is fully stateless, allowing an easy load-balancing.

3.1. AUTHENTICATION

3.1.1. AUTHENTICATE()

3.1.1.1. Request

```
curl --location 'http://localhost:8488/api/spvitamin/authenticate' \
--header 'Authorization: Basic bWFuYWdlcjptYW5hZ2Vy'
```

3.1.1.2. Response

The access token can be found in the jwt field of the response. The access token must be provided with all subsequent API calls.

```
{
    "sub": "manager",
    "jwt":
"eyJhbGciOiJSUzUxMiJ9.eyJzdWIiOiJtYW5hZ2VyIiwiaWF0IjoxNzM1NjQ0Mjk5LCJleHAiOjE3MzU3MzA
2OTksInVpZCI6LTEsInJscyI6WyJVUERBVEVfRE9DVU1FTlQiLCJTWVNURU1fR0VUX0RPQ1VNRU5UX1RZUEVT
IiwiU1lTVEVNX0dFVF9LRVlXT1JEX1RZUEVTIiwiRE9XTkxPQUQiLCJTWVNURU1fR0VUX0RPQ1VNRU5UX1RZU
EVfR1JPVVBTIiwiUkVBRF9ET0NVTUVOVCIsIkdFVF9ET0NVTUVOVF9LRVlXT1JEUyIsIlFVRVJZX0RPQ1VNRU
5UIiwiUk9MRV9VU0VSIiwiVVBMT0FEIiwiUk9MRV9NQU5BR0VSIiwiREVMRVRFX0RPQ1VNRU5UIiwiUk9MRV9
FTVBUWSIsIkdFVF9ET0NVTUVOVF9JTkZPIiwiQ1JFQVRFX0RPQ1VNRU5UIl0sImxkYXAiOiJOb3QgYW4gQUQg
dXNlciJ9.UcqmsHE7N_LU6qgBXacml3SEMRLJWHCP_rC4OpLP4u2pV9_g5vF43mdWMZZ65QjWDr0qvz_q13lW
ZbVZ0uVqntfzwx4G0qHcvJCtBPIksAPWr9uLb1Mgl10RgQIi775mx-
ZqzX3Xv9NjJAqWG6e_1PtdBT7zIPHP3PknAfoxlQ-n-rasXxOCTid0drUGICyVo4nqeuqj-
m6DQQ_ToahxP8jLcZFgGrtiV07oruYct8Cm0FTR6nb99M-MsOfYxBceMxV9JJaXQfNWHMOvx2s-
r6VK_m9XaOzIokeMNqmhc7AQ0c1lqr0gsVAt2C1ltS61VqNJzzDXNjsCttbOoikVIw",
    "iat": "2024-12-31T12:24:59.687",
    "exp": "2025-01-01T12:24:59.687"
}
```

3.2. CONFIG

3.2.1. GETCONFIG()

3.2.1.1. Request

```
curl --location 'http://localhost:8488/api/config'
```

3.2.1.2. Response

The endpoint returnes the current configuration as a yaml file.

3.2.2. APPLYCONFIG()

3.2.2.1. Request

```
curl --location 'http://localhost:8488/api/config' \
--form 'file=@"/C:/np/github/microdms-project/test-data/microdms-config.yaml"'
```

3.2.2.2. Response

Only the HTTP response code is returned.

3.3. SYSTEM

3.3.1. GETDOCUMENTTYPES()

3.3.1.1. Request

```
curl --location 'http://localhost:8488/api/system/documenttypes'

curl --location
'http://localhost:8488/api/system/documenttypes?documentTypeGroupName=Test'
```

3.3.1.2. Response

The endpoint returns the accessible document types for the authenticated user. If the documentTypeGroupName query parameter is provided, the result contains only document types of the given document type group.

```
[
    {
        "id": "1",
        "name": "KeywordTest"
    },
    {
        "id": "2",
        "name": "PerformanceTest"
    },
    {
        "id": "3",
        "name": "Employee"
    },
    {
        "id": "4",
        "name": "Invoice"
    }
]
```

3.3.2. GETDOCUMENTTYPEGROUPS()

3.3.2.1. Request

```
curl --location 'http://localhost:8488/api/system/documenttypegroups'
```

3.3.2.2. Response

```
[
    "Test",
    "HR Documents",
    "FIN Documents"
]
```

3.3.3. GETKEYWORDTYPES()

3.3.3.1. Request

```
curl --location
'http://localhost:8488/api/system/keywordtypes?documentTypeName=Employee'
```

3.3.3.2. Response

```
[
    {
        "name": "FirstName",
        "id": 8,
        "dataType": "ALPHANUMERIC",
        "length": 30,
        "dataTypeOption": "ALPHANUMERIC_MIXEDCASE"
    },
    {
        "name": "LastName",
        "id": 9,
        "dataType": "ALPHANUMERIC",
        "length": 30,
        "dataTypeOption": "ALPHANUMERIC_MIXEDCASE"
    },
    {
        "name": "Birthdate",
        "id": 10,
        "dataType": "DATE"
    },
    {
        "name": "Address",
        "id": 11,
        "dataType": "ALPHANUMERIC",
        "length": 100,
        "dataTypeOption": "ALPHANUMERIC_MIXEDCASE"
    },
    {
        "name": "Salery",
        "id": 12,
        "dataType": "INTEGER"
    }
]
```

3.4. FILES

3.4.1. UPLOAD()

Files can be uploaded in binary form with the following endpoint.

3.4.1.1. Request

```
curl --location 'http://localhost:8488/api/files' \
--form 'file=@"/C:/np/github/microdms-project/test-data/SZEKK.pdf"'
```

3.4.1.2. Response

The response is the reference of the file within the temporary store. The file will be removed from the temp store after successful archiving or after 1 hour if not used.

```
{
    "location": "2024/12/31/12/51/6915b3ba-8167-48a9-9645-025db3f6e5a0"
}
```

3.5. DOCUMENTS

3.5.1. CREATEDOCUMENT()

3.5.1.1. Request

```
curl --location 'http://localhost:8488/api/documents' \
--header 'Content-Type: application/json' \
--data '{
    "documentTypeName": "Employee",
    "docRef": "2025/02/02/08/42/3fba3113-c933-47da-8be4-1312b2ee5f85",
    "fileName": "excelfile.pdf",
    "keywords": [
        {
            "name": "FirstName",
            "value": "John"
        },
        {
            "name": "LastName",
            "value": "Doe"
        },
        {
            "name": "Birthdate",
            "value": "1999-12-01"
        }
    ]
}'
```

Multivalue keywords are supported in the following form.

```
{
    "documentTypeName": "KeywordTest",
    "docRef": "{{fileRef}}",
    "fileName": "excelfile.pdf",
    "keywords": [
        {
            "name": "Alphanumeric",
            "values": [
                "alma",
                "körte"
            ]
        },
        {
            "name": "Integer",
            "values": [
                123,
                124
            ]
        },
        {
            "name": "Long",
            "values": [
                123456789,
                1234567890
            ]
        },
        {
            "name": "Date",
            "values": [
                "2024-11-30",
                "2024-12-01"
            ]
        }
    ]
}
```

Please note the syntax: instead of "value", the plural form is used: "values" and the value is supposed to be a list of objects.

3.5.1.2. Response

```
{
    "location": "http://localhost:8488/api/documents/12409602"
}
```

3.5.2. READDOCUMENT()

3.5.2.1. Request

```
curl --location 'http://localhost:8488/api/documents/12409602'
```

3.5.2.2. Response

The file content as application/octet-stream.

3.5.3. UPDATEDOCUMENT()

3.5.3.1. Request

```
curl --location --request PUT 'http://localhost:8488/api/documents' \
--header 'Content-Type: application/json' \
--data '{
    "docId": "10001",
    "keywords": [
        {
            "name": "Alphanumeric",
            "value": "TEST"
        }
    ],
    "docRef": "2024/12/31/12/51/6915b3ba-8167-48a9-9645-025db3f6e5a0",
    "fileName": "updated.pdf"
}'
```

The keyword will be deleted if the value is 'null'. Both keywords and docRef are optional, only the provided data will be updated.

3.5.3.1.1. Response

```
{
    "id": "10001",
    "rev": "1",
    "name": "updated.pdf",
    "documentDate": "2024-12-31T13:13:50.335+0100",
    "dateStored": "2024-12-31T13:13:50.362+0100",
    "documentTypeName": "PerformanceTest",
    "documentTypeId": "3",
    "createdBy": "manager",
    "latestAllowedRevisionID": 1,
    "status": 0,
    "fullFileName": "updated.pdf",
    "fileSize": 80728
}
```

3.5.4. DELETEDOCUMENT

3.5.4.1. Request

The body part shall contain the type of the removal. There are two types:

- LOGICAL
- PHYSICAL

```
curl --location --request DELETE 'http://localhost:8488/api/documents/12407517' \
--header 'Content-Type: application/json' \
--data '"PHYSICAL"'
```

3.5.4.2. Response

Only the HTTP status will be returned.

3.5.5. GETDOCUMENTKEYWORDS()

3.5.5.1. Request

```
curl --location --request POST 'http://localhost:8488/api/documents/8365730/keywords'
```

3.5.5.2. Response

```
{
    "keywords": {
        "LASTNAME": {
            "name": "LastName",
            "value": "Doe",
            "dataType": "ALPHANUMERIC",
            "dataTypeOption": "ALPHANUMERIC_MIXEDCASE"
        },
        "FIRSTNAME": {
            "name": "FirstName",
            "value": "John",
            "dataType": "ALPHANUMERIC",
            "dataTypeOption": "ALPHANUMERIC_MIXEDCASE"
        },
        "BIRTHDATE": {
            "name": "Birthdate",
            "value": "1999-12-01",
            "dataType": "DATE"
        }
    }
}
```

An example of a multivalue keyword:

```
{
    "keywords":
        "ALPHANUMERIC": {
            "name": "Alphanumeric",
            "values": [
                "ALMAKÖRTE",
                "BARACK",
                "SZILVA"
            ],
            "dataType": "ALPHANUMERIC",
            "dataTypeOption": "ALPHANUMERIC_UPPERCASE",
            "multiValue": true
        }
    }
}
```

3.5.6. GETDOCUMENTINFO()

3.5.6.1. Request

```
curl --location 'http://localhost:8488/api/documents/102/info'
```

3.5.6.2. Response

```
{
    "id": 8365730,
    "rev": 0,
    "name": "excelfile",
    "documentDate": "2025-02-02T08:43:10.386+0100",
    "dateStored": "2025-02-02T08:43:10.392+0100",
    "documentTypeName": "Employee",
    "documentTypeId": 3,
    "createdBy": "manager",
    "latestAllowedRevisionID": 0,
    "status": 0,
    "fileName": "excelfile.pdf",
    "fileSize": 80728,
    "media": "smb://perit/smb-test",
    "location": "2025/02/02/08/43/2ee618cc-d5ba-4284-bd9a-04196ec1fee6"
}
```

3.5.7. QUERYDOCUMENT()

The queryDocument() endpoint is a powerful tool for searching documents by document attributes or by keywords. The result set will be a distinct list of documents, even if the result set contains keywords with multiple values.

3.5.7.1. Request

```
curl --location 'http://localhost:8488/api/query?size=20&page=0' \
--data '{
    "where": [
        {
            "property": "Document.ID",
            "relation": "=",
            "value": 45209
        }
    ],
    "orderBy": {
        "property": "Alphanumeric",
        "direction": "ASC"
    },
    "include": ["Alphanumeric", "DateTime"],
    "countOnly": false
}'
```

The request has 4 body parts: **where**, **orderBy**, **include** and **countOnly**, and it has 2 query parameters for pagination: **size** and **page**.

3.5.7.1.1. The where clause

The where clause consists of a list of criteria, which will be handled using the AND relation. With other words the more criteria is provided here, the result set will be smaller. The property may be a keyword name or the following document attributes:

- Document.ID
- Document.Name
- Document.Type
- Document.Date
- Document.DateStored
- Document.CreatedBy

The relation can be one of the following:

- "="
- "<>"
- ">"
- ">="
- "<"
- "<="
- "IN"
- "BETWEEN"
- "LIKE"

3.5.7.1.2. Narrowing the result using Document.Type

By default all document types are queried which contain the given keywords. You can narrow down the result like this:

```
curl --location 'http://localhost:8488/api/query' \
--data '{
    "where": [
        {
            ...
        },
        {
            "property": "Document.Type",
            "relation": "=",
            "value": "Employee"
        }
    ]
}'
```

3.5.7.1.3. Using the IN relation

When querying for documents using the IN relation, the values must be a list of objects.

```
curl --location 'http://localhost:8488/api/query?size=3' \
--header 'Content-Type: application/json' \
--data '{
    "where": [
        {
            "property": "Alphanumeric",
            "relation": "IN",
            "values": [
                "ID_163",
                "ID_164",
                "ID_165"
            ]
        }
    ]
}'
```

3.5.7.1.4. Using the BETWEEN relation

In this case the values list must contain exactly 2 items.

```
curl --location 'http://localhost:8488/api/query' \
--data '{
    "where": [
        {
            "property": "Alphanumeric",
            "relation": "BETWEEN",
            "values": [
                "ID_1",
                "ID_10"
            ]
        }
    ]
}'
```

3.5.7.1.5. Using the LIKE relation

The LIKE relation may only be used with alphanumeric keywords. Since those keywords are stored as fixed length texts (they are padded with spaces), please use % at the end of the filter text.

```
curl --location 'http://localhost:8488/api/query' \
--data '{
    "where": [
        {
            "property": "Alphanumeric",
            "relation": "LIKE",
            "value": "%ALMA%"
        }
    ]
}'
```

3.5.7.1.6. Sorting the result

You can use the **orderBy** part of the request for sorting the result. Please consider that sorting will be applied on the result set specified by the where clause, that is why it may be slow if the set to be sorted is large.

For saving server resources, the orderBy clause will be ignored if the result set is larger than 100.000 documents. In that case a warning appears in the result.

```
Request:

{
    "where": [
        {
            "property": "Document.ID",
            "relation": "<>",
            "value": null
        }
    ],
    "orderBy": {
        "property": "Alphanumeric",
        "direction": "ASC"
    },
    "include": ["Alphanumeric"]
}

Response:
{
    "size": 100,
    "page": 1000,
    "totalPages": 124068,
    "totalItems": 12406736,
    "warning": "OrderBy clause ignored due to too many results!",
    "keywords": [
        "Alphanumeric"
    ],
    "list": [
        {
            "documentInfo": {
                "id": "100001",
                "rev": "0",
                "name": "tiny.txt",
                "documentDate": "2024-12-22T14:05:04.258+0100",
                "dateStored": "2024-12-22T14:05:04.258+0100",
                "documentTypeName": "PerformanceTest",
                "documentTypeId": "3",
                "createdBy": "manager",
                "status": 0
            },
            "keywords": [
                {
                    "name": "Alphanumeric",
                    "value": "ID_486",
                    "dataType": "ALPHANUMERIC",
                    "dataTypeOption": "ALPHANUMERIC_UPPERCASE"
                }
            ]
        },
        ...
    ]
}
```

The socket-timeout is set to 20 seconds, which will terminate each query after this amount of time.

3.5.7.1.7. Including keywords in the result

You can get not only document attributes with the result, but also keywords. This is especially good if you want to show the result in a data table, because the whole table is retrieved in one step. Please use the "include" field in the request.

3.5.7.1.8. Pagination of the result

The response contains information about the size of the result set. You can iterate through the result set using the usual size and page parameters in the query like this:

```
curl --location 'http://localhost:8488/api/query?size=3&page=10'
```

If not provided, 100 items will be returned with offset = 0.

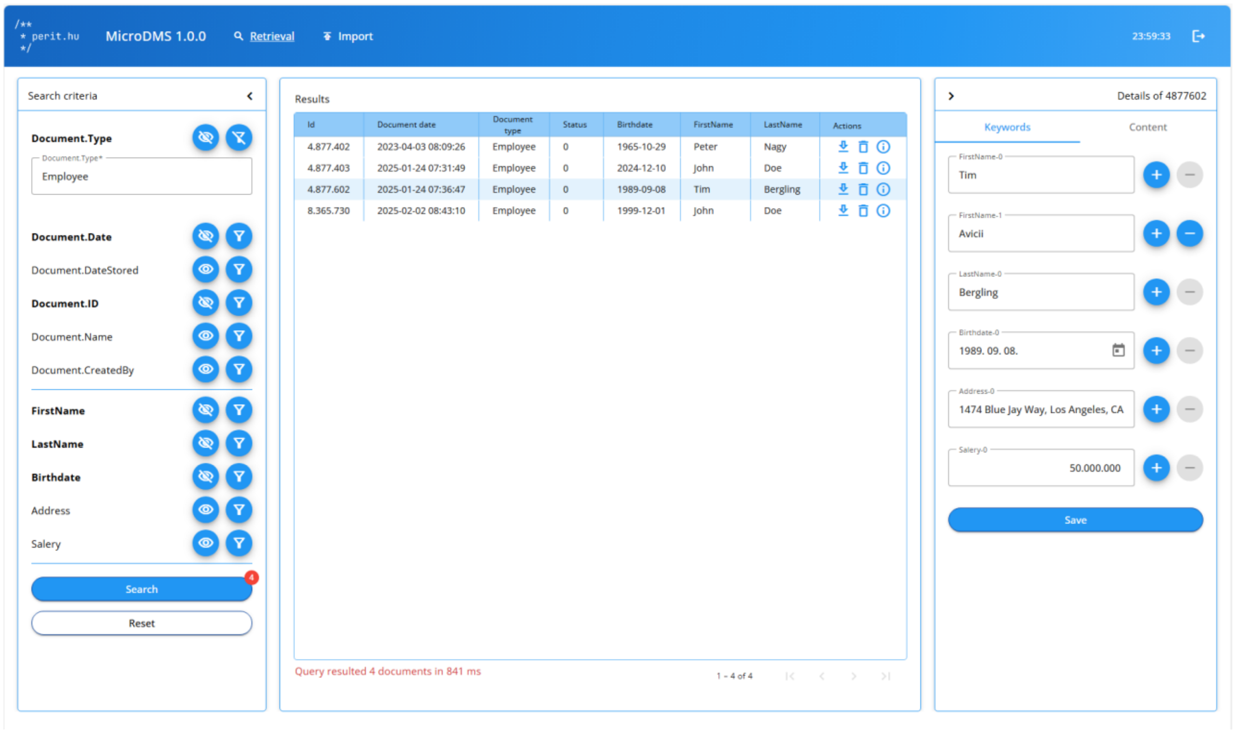
3.5.7.2. Response

```
{
    "size": 1,
    "page": 0,
    "totalPages": 1,
    "totalItems": 1,
    "keywords": [
        "Alphanumeric",
        "DateTime"
    ],
    "list": [
        {
            "documentInfo": {
                "id": "45209",
                "rev": "0",
                "name": "tiny.txt",
                "documentDate": "2024-12-22T13:48:56.273+0100",
                "dateStored": "2024-12-22T13:48:56.273+0100",
                "documentTypeName": "PerformanceTest",
                "documentTypeId": "3",
                "createdBy": "manager",
                "status": 0
            },
            "keywords": [
                {
                    "name": "Alphanumeric",
                    "value": "ID_748",
                    "dataType": "ALPHANUMERIC",
                    "dataTypeOption": "ALPHANUMERIC_UPPERCASE"
                },
                {
                    "name": "DateTime",
                    "value": "2024-12-22T12:48:56.236+0100",
                    "dataType": "DATETIME"
                }
            ]
        }
    ]
}
```

4. MICRODMS CLIENT

4.1. THE RETRIEVAL PAGE

The main purpose of the client is searching for documents. The result set is displayed on the central panel.



Document.Type

Document.Type*: Employee

Document.Date

Document.DateStored

Document.ID

Document.Name

Document.CreatedBy

FirstName

LastName

Birthdate

Address

Salery

Id	Document date	Document type	Status	Birthdate	FirstName	LastName	Actions
4.877.402	2023-04-03 08:09:26	Employee	0	1965-10-29	Peter	Nagy	
4.877.403	2025-01-24 07:31:49	Employee	0	2024-12-10	John	Doe	
4.877.602	2025-01-24 07:36:47	Employee	0	1989-09-08	Tim	Bergling	
8.365.730	2025-02-02 08:43:10	Employee	0	1999-12-01	John	Doe	

Query resulted 4 documents in 841 ms

1 – 4 of 4

Details of 4877602

FirstName-0: Tim

FirstName-1: Avicii

LastName-0: Bergling

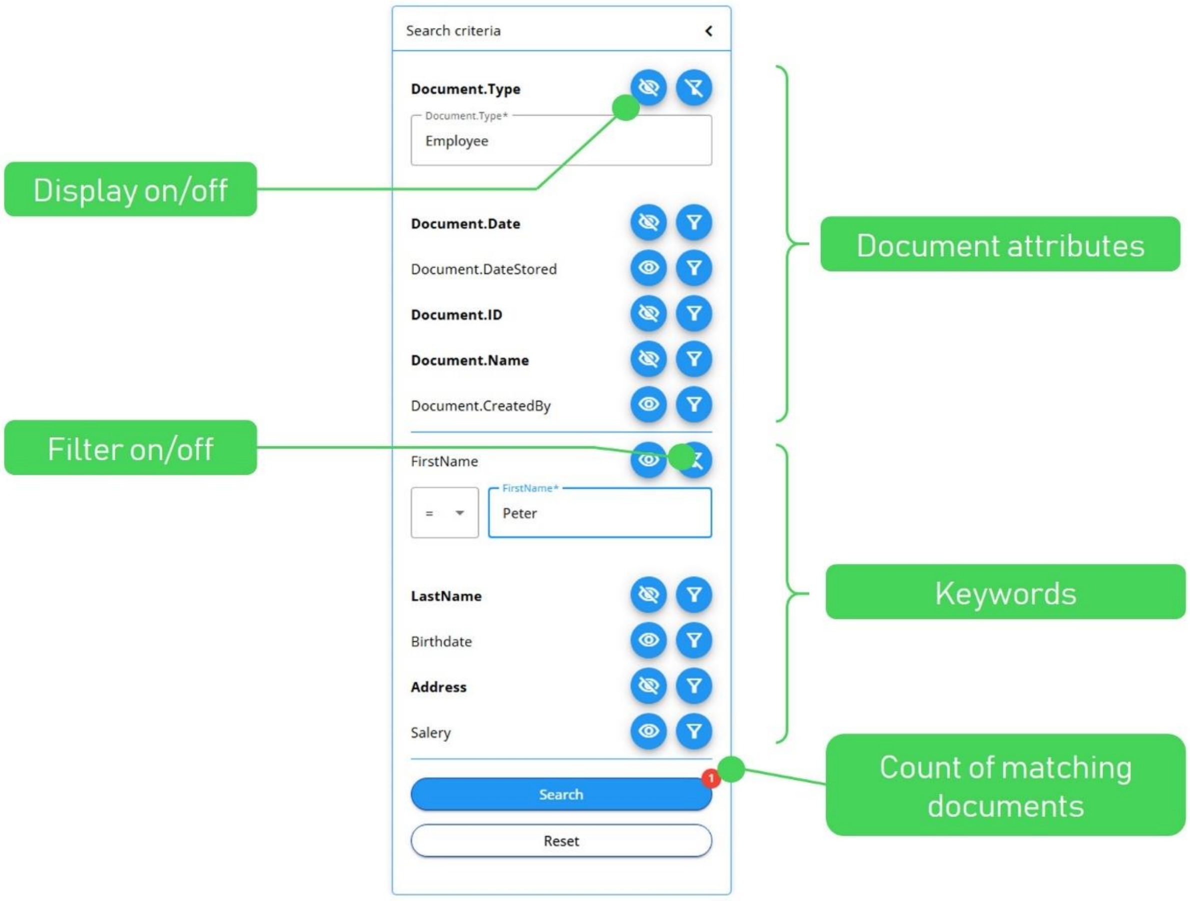
Birthdate-0: 1989. 09. 08.

Address-0: 1474 Blue Jay Way, Los Angeles, CA

Salery-0: 50.000.000

Figure 1: The MicroDMS web client

The search result can be narrowed down by specifying some search criteria on the left panel. Here we can switch on/off document attributes and keywords.



Search criteria

Document.Type

Document.Type* Employee

Document.Date

Document.DateStored

Document.ID

Document.Name

Document.CreatedBy

FirstName

= FirstName* Peter

LastName

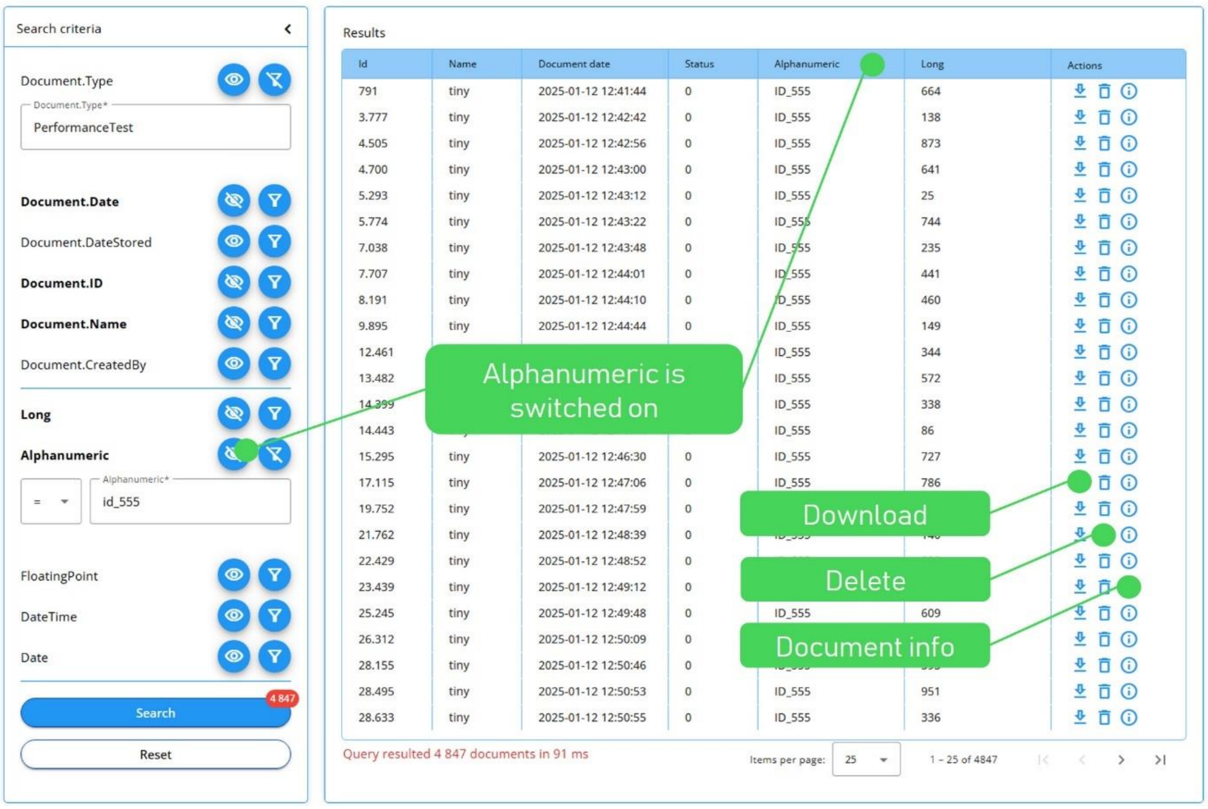
Birthdate

Address

Salery

__Figure 2: The Search criteria panel__

Document.ID, Document.Name, Document.Date and Document.Type are displayed on the result panel, others are hidden.



Search criteria

Document.Type	Document.Type*: PerformanceTest
Document.Date	
Document.DateStored	
Document.ID	
Document.Name	
Document.CreatedBy	
Long	
Alphanumeric	= Alphanumeric*: id_555
FloatingPoint	
DateTime	
Date	

Search 4 847

Reset

Results

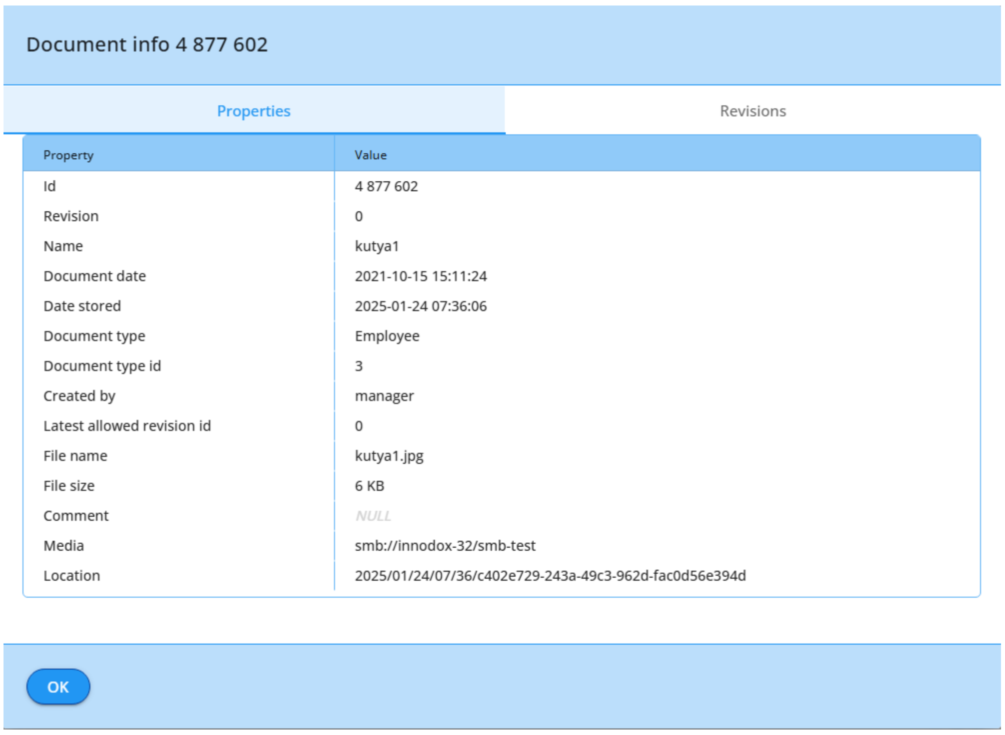
Id	Name	Document date	Status	Alphanumeric	Long	Actions
791	tiny	2025-01-12 12:41:44	0	ID_555	664	⤓ 🗑 ⓘ
3.777	tiny	2025-01-12 12:42:42	0	ID_555	138	⤓ 🗑 ⓘ
4.505	tiny	2025-01-12 12:42:56	0	ID_555	873	⤓ 🗑 ⓘ
4.700	tiny	2025-01-12 12:43:00	0	ID_555	641	⤓ 🗑 ⓘ
5.293	tiny	2025-01-12 12:43:12	0	ID_555	25	⤓ 🗑 ⓘ
5.774	tiny	2025-01-12 12:43:22	0	ID_555	744	⤓ 🗑 ⓘ
7.038	tiny	2025-01-12 12:43:48	0	ID_555	235	⤓ 🗑 ⓘ
7.707	tiny	2025-01-12 12:44:01	0	ID_555	441	⤓ 🗑 ⓘ
8.191	tiny	2025-01-12 12:44:10	0	ID_555	460	⤓ 🗑 ⓘ
9.895	tiny	2025-01-12 12:44:44	0	ID_555	149	⤓ 🗑 ⓘ
12.461	[illegible]	[illegible]	[illegible]	ID_555	344	⤓ 🗑 ⓘ
13.482	[illegible]	[illegible]	[illegible]	ID_555	572	⤓ 🗑 ⓘ
14.399	[illegible]	[illegible]	[illegible]	ID_555	338	⤓ 🗑 ⓘ
14.443	[illegible]	[illegible]	[illegible]	ID_555	86	⤓ 🗑 ⓘ
15.295	tiny	2025-01-12 12:46:30	0	ID_555	727	⤓ 🗑 ⓘ
17.115	tiny	2025-01-12 12:47:06	0	ID_555	786	🗑 ⓘ
19.752	tiny	2025-01-12 12:47:59	0	[illegible]	[illegible]	⤓ 🗑 ⓘ
21.762	tiny	2025-01-12 12:48:39	0	[illegible]	[illegible]	⤓ ⓘ
22.429	tiny	2025-01-12 12:48:52	0	[illegible]	[illegible]	⤓ 🗑 ⓘ
23.439	tiny	2025-01-12 12:49:12	0	[illegible]	[illegible]	⤓ 🗑
25.245	tiny	2025-01-12 12:49:48	0	ID_555	609	⤓ 🗑 ⓘ
26.312	tiny	2025-01-12 12:50:09	0	[illegible]	[illegible]	⤓ 🗑 ⓘ
28.155	tiny	2025-01-12 12:50:46	0	[illegible]	[illegible]	⤓ 🗑 ⓘ
28.495	tiny	2025-01-12 12:50:53	0	ID_555	951	⤓ 🗑 ⓘ
28.633	tiny	2025-01-12 12:50:55	0	ID_555	336	⤓ 🗑 ⓘ

Query resulted 4 847 documents in 91 ms

Items per page: 25 1 – 25 of 4847

Figure 3: Customized result set

The table columns can be sorted if the result set is smaller than 10.000 documents.



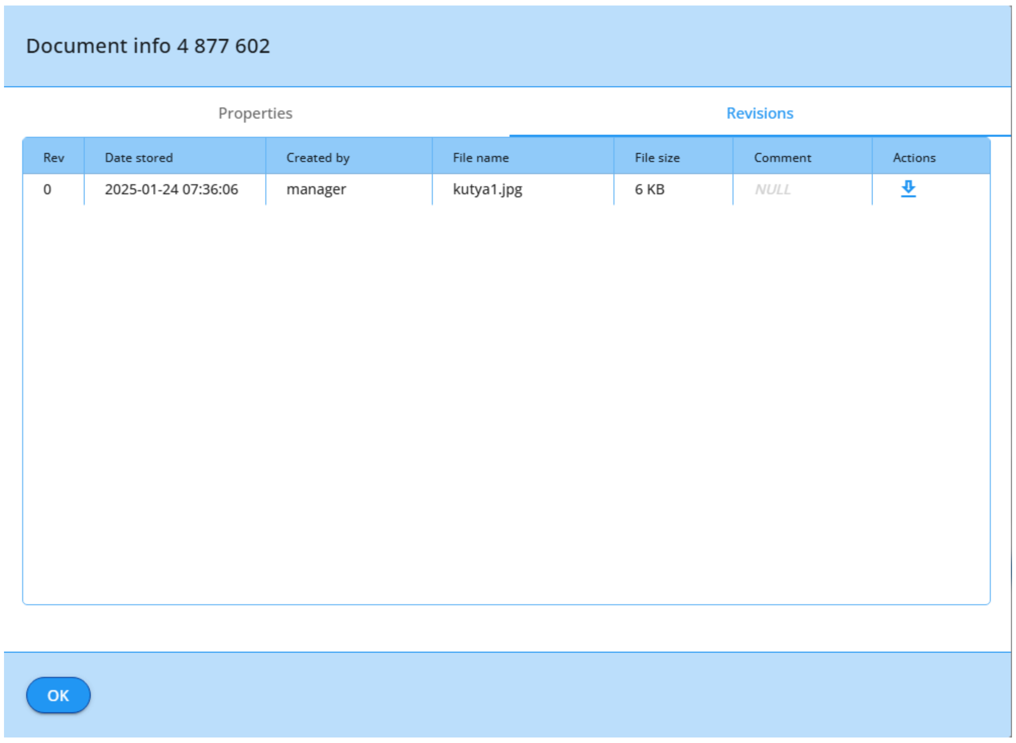
Document info 4 877 602

Properties | Revisions

Property	Value
Id	4 877 602
Revision	0
Name	kutya1
Document date	2021-10-15 15:11:24
Date stored	2025-01-24 07:36:06
Document type	Employee
Document type id	3
Created by	manager
Latest allowed revision id	0
File name	kutya1.jpg
File size	6 KB
Comment	NULL
Media	smb://innodox-32/smb-test
Location	2025/01/24/07/36/c402e729-243a-49c3-962d-fac0d56e394d

OK

Figure 4: Document info - properties



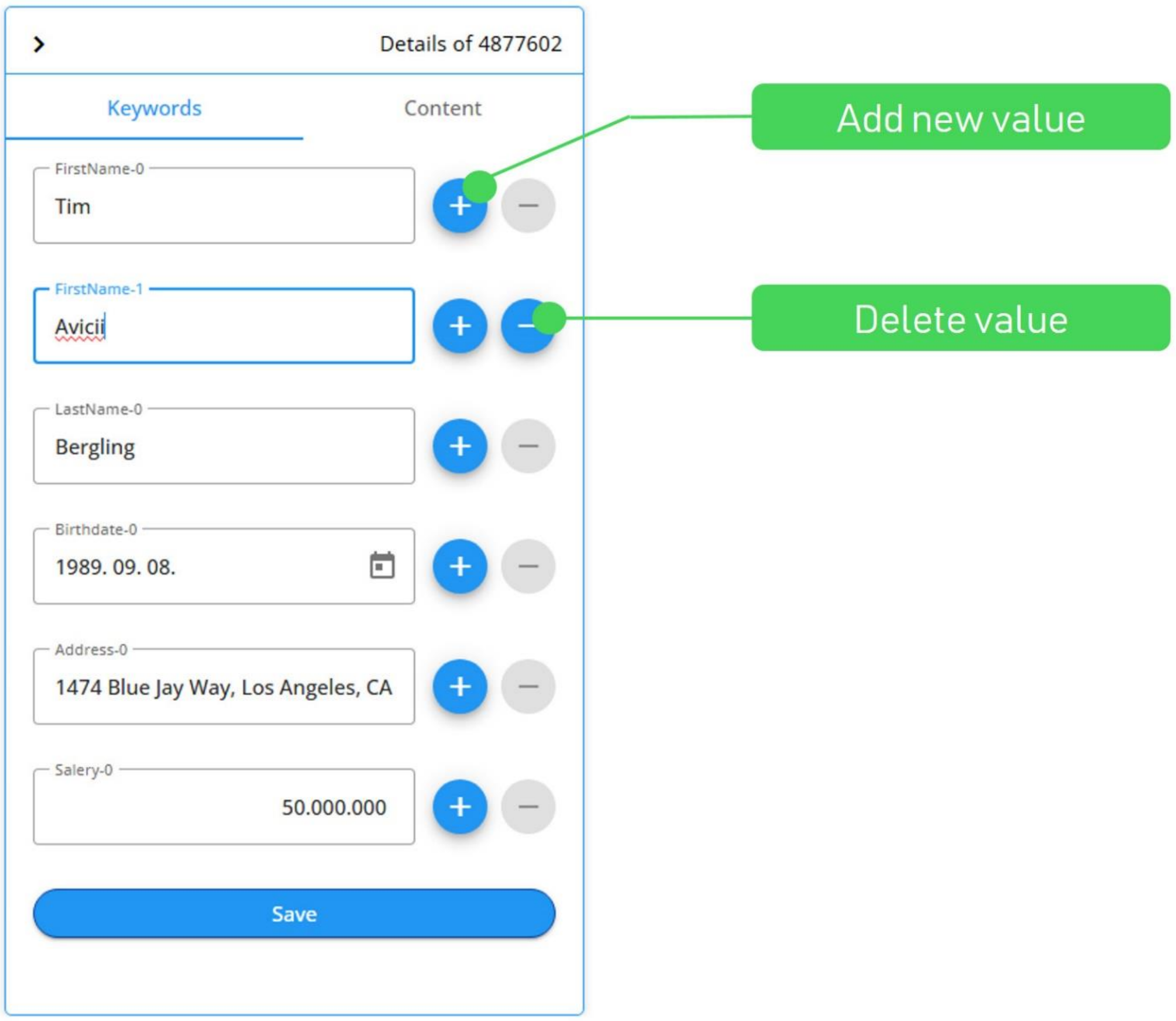
Document info 4 877 602

Properties | Revisions

Rev	Date stored	Created by	File name	File size	Comment	Actions
0	2025-01-24 07:36:06	manager	kutya1.jpg	6 KB	NULL	⬇

OK

Figure 5: Document info - revisions



Details of 4877602

Keywords | Content

FirstName-0: Tim

FirstName-1: Avicii

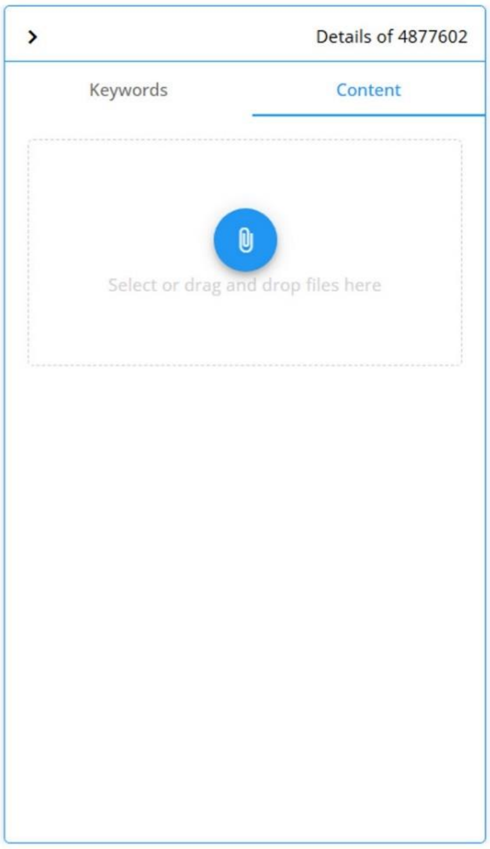
LastName-0: Bergling

Birthdate-0: 1989. 09. 08.

Address-0: 1474 Blue Jay Way, Los Angeles, CA

Salery-0: 50.000.000

Figure 6: Changing keywords of a document



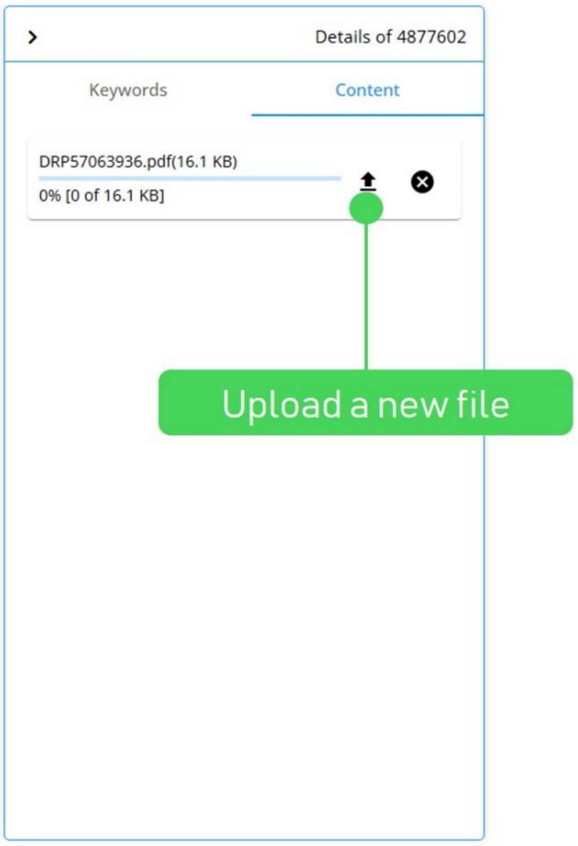
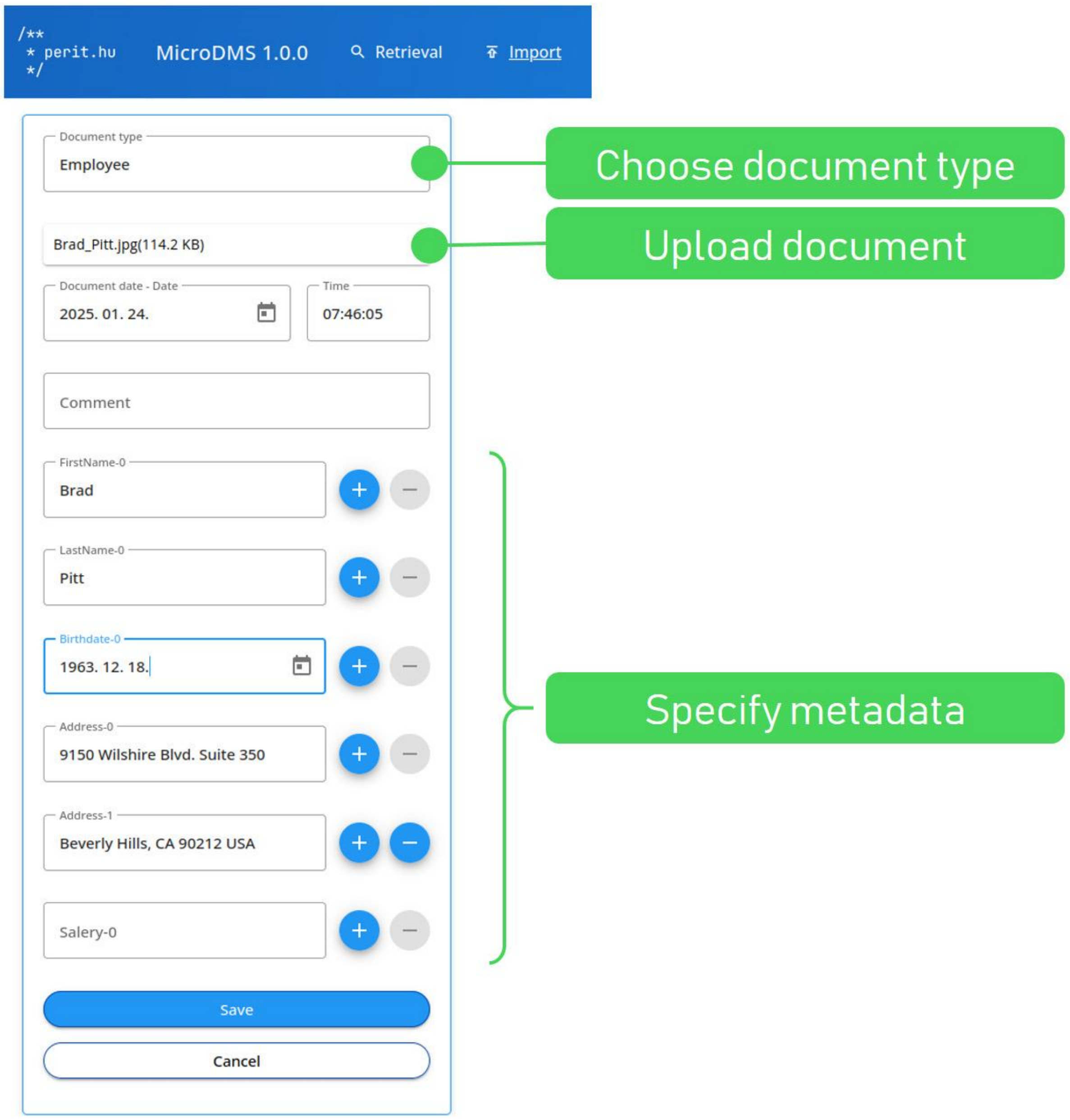


Figure 7: Updateing the content

4.2. THE IMPORT PAGE



Document type: Employee

Brad_Pitt.jpg(114.2 KB)

Document date - Date: 2025. 01. 24.

Time: 07:46:05

Comment

FirstName-0: Brad

LastName-0: Pitt

Birthdate-0: 1963. 12. 18.

Address-0: 9150 Wilshire Blvd. Suite 350

Address-1: Beverly Hills, CA 90212 USA

Salery-0

Figure 8: Import a new file into the archive

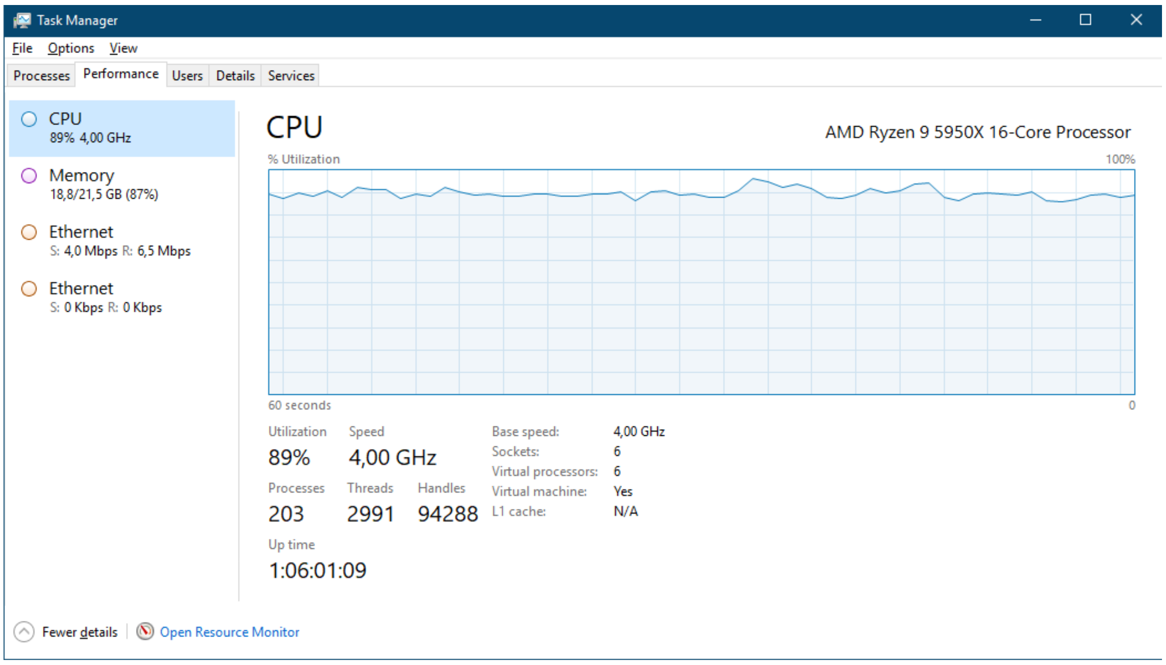
5. PERFORMANCE AND VOLUME MEASURES

The performance was measured on a Windows Server 2019 computer with 6 cores and 20 GB RAM. The MicroDMS service was running in one single instance. The PostgreSQL database was running in Docker on an Ubuntu system with 4 cores and 30 GB RAM.

The load was generated by 3 JMeter scripts, each running in 10 threads for an hour. The 3 scripts are:

- createDocument() using 5 keywords and a very small text document.
- readDocument() with random docIds between 0 and 10.000.000.
- queryDocument() using 2 keywords for the query which results approximatelly 10-20 results.

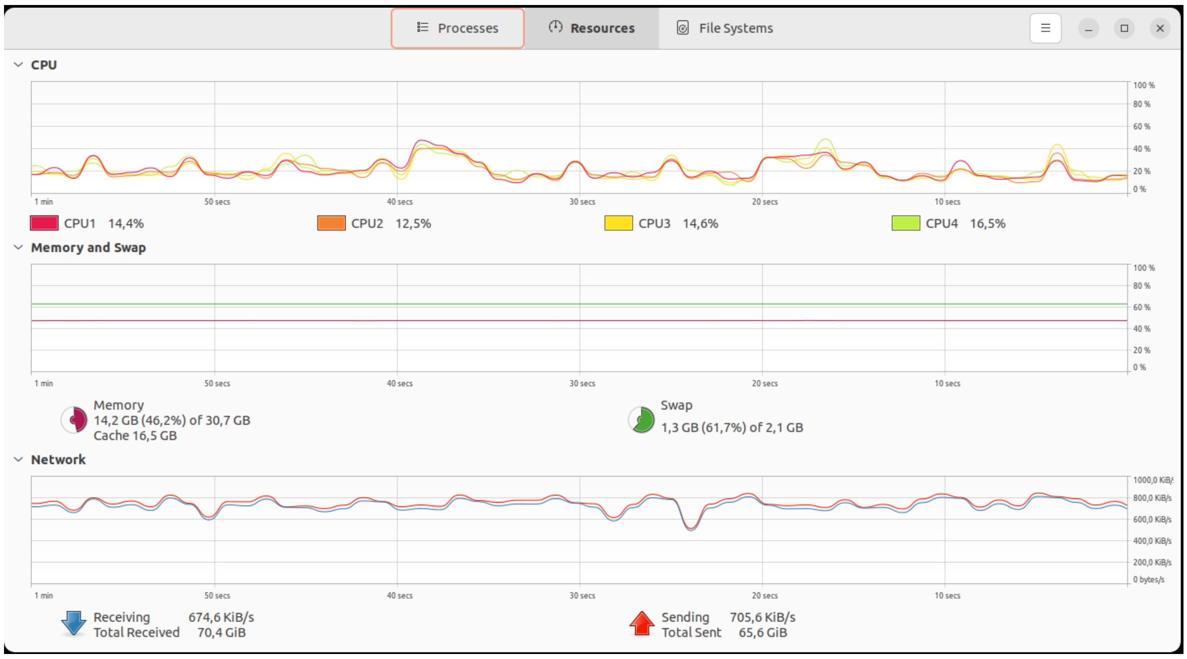
The database contained ~12.500.000 documents at the time of testing. The web client worked fine during testing, it was responsive, one could work with it without any problems.



Utilization 89%
Speed 4,00 GHz
Processes 203
Threads 2991
Handles 94288
Up time 1:06:01:09

Base speed:	4,00 GHz
Sockets:	6
Virtual processors:	6
Virtual machine:	Yes
L1 cache:	N/A

Figure 9: CPU load of the Windows system during testing



🟥 CPU1 14,4% 🟧 CPU2 12,5% 🟨 CPU3 14,6% 🟩 CPU4 16,5%

Memory
14,2 GB (46,2%) of 30,7 GB
Cache 16,5 GB

Swap
1,3 GB (61,7%) of 2,1 GB

Receiving 674,6 KiB/s
Total Received 70,4 GiB

Sending 705,6 KiB/s
Total Sent 65,6 GiB

Figure 10: Resource consumption of the Ubuntu computer hosting the database

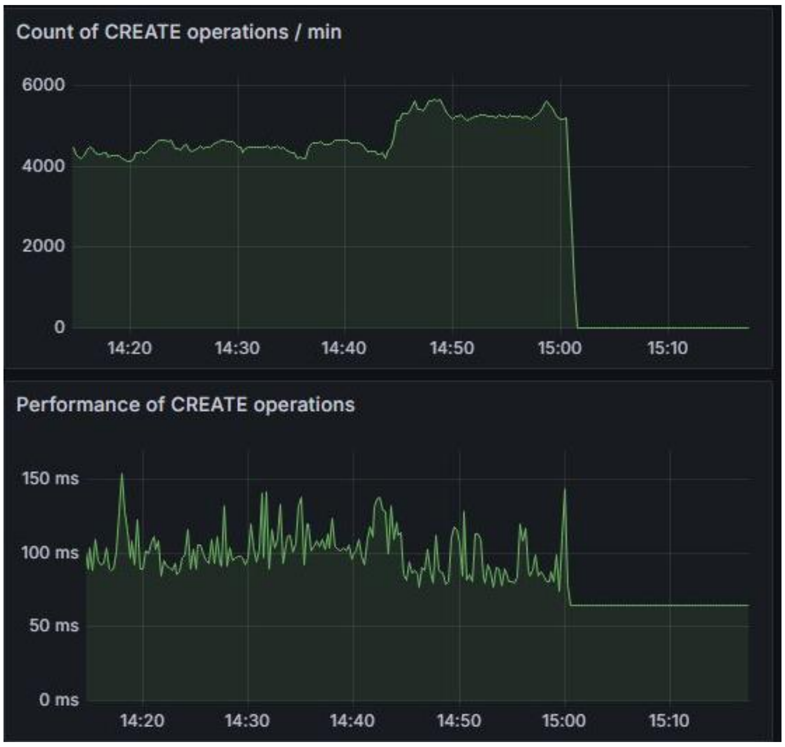


Figure 11: Performance of createDocument()

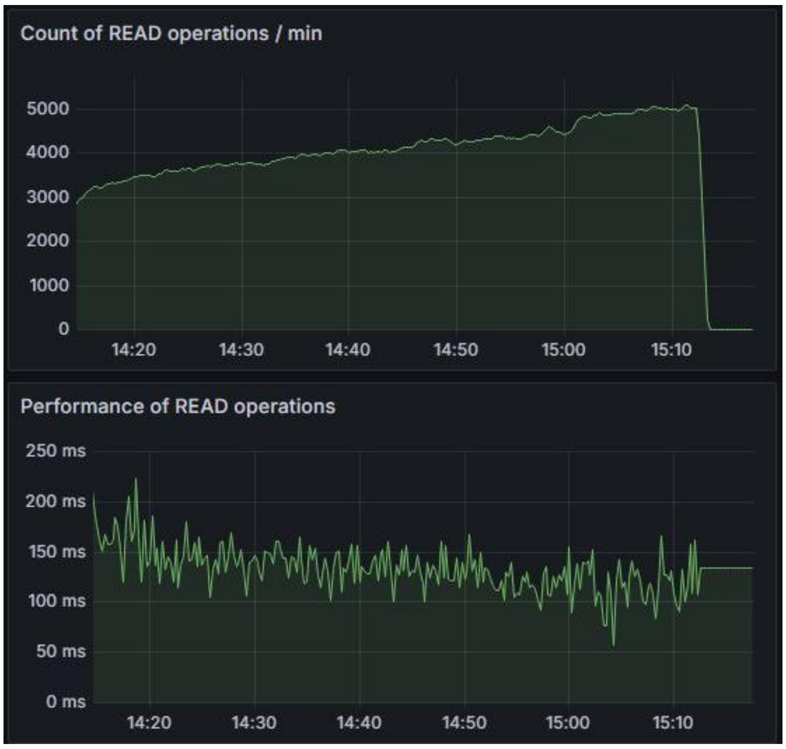


Figure 12: Performance of readDocument()

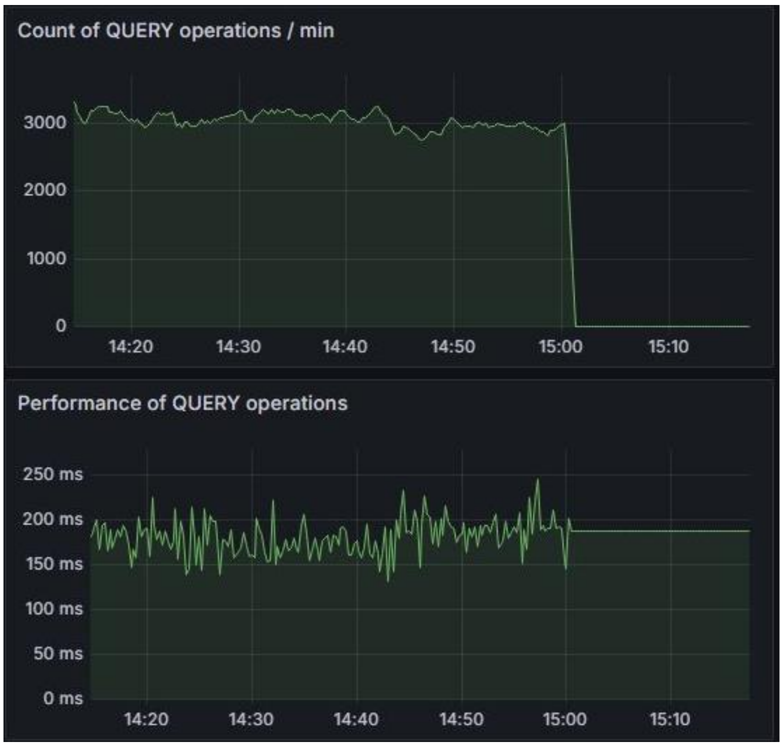


Figure 13: Performance of queryDocument()

Of course the system is horizontally scalable by running more than one instances of the service.

6. APPENDIX

6.1. REFERENCED DOCUMENTS

Ref	Név
1	
2	
3	

6.2. LIST OF FIGURES

Figure 1: The MicroDMS web client 28
Figure 2: The Search criteria panel 29
Figure 3: Customized result set 30
Figure 4: Document info - properties....31
Figure 5: Document info - revisions 31
Figure 6: Changing keywords of a document 32
Figure 7: Updateing the content 33
Figure 8: Import a new file into the archive 34
Figure 9: CPU load of the Windows system during testing....35
Figure 10: Resource consumption of the Ubuntu computer hosting the database 36
Figure 11: Performance of createDocument() 36
Figure 12: Performance of readDocument() 37
Figure 13: Performance of queryDocument() 37

6.3. OPEN ISSUES

6.4. REQUIREMENT KEYS

No index entries found.