../alex-visualizer/cmd/main.go

```
package main

import (
    "fmt"
    "log"
    "neo4j-mysql-bridge/internal/mysql"
    "neo4j-mysql-bridge/internal/neo4j"
    "neo4j-mysql-bridge/internal/visualization"
)

func main() {
    mysqlClient, err := mysql.NewClient()
    if err != nil {
        log.Fatalf("Failed to connect to MySQL: %v", err)
    }
    defer mysqlClient.Close()

    neo4jClient, err := neo4j.NewClient()
    if err != nil {
        log.Fatalf("Failed to connect to Neo4j: %v", err)
    }
    defer neo4jClient.Close()

    // Transform data
    transformer := analysis.NewTransformer()
    mysqlData, err := mysqlClient.FetchData() // Replace with the actual function to fetch data from MySQL
    if err != nil {
        log.Fatalf("Failed to fetch data from MySQL: %v", err)
    }
    transformedData := transformer.TransformData(mysqlData)

    // Import transformed data to Neo4j
    // Example: neo4jClient.InsertData(transformedData)

    // Start visualization
    visualizer := visualization.NewVisualizer()
    visualizer.ServeVisualization()

    fmt.Println("Data transfer and visualization complete!")
}
```

../alex-visualizer/config/config.go

```
package config

import (
    "github.com/spf13/viper"
)

func LoadConfig() error {
    viper.SetConfigName("config")
    viper.AddConfigPath(".")
    viper.AutomaticEnv()

    return viper.ReadInConfig()
}
```

../alex-visualizer/internal/analysis/transformer.go

```
package analysis

import "neo4j-mysql-bridge/internal/mysql"

type Transformer struct {
    // add any necessary fields
}

func NewTransformer() *Transformer {
    return &Transformer{}
}

func (t *Transformer) TransformData(mysqlData []mysql.DataType) []neo4j.DataType {
    // Implement the transformation logic here
    var neo4jData []neo4j.DataType
    // Example: loop through mysqlData and convert to neo4jData
    return neo4jData
```

}

../alex-visualizer/internal/migration/migration.go

```
package migration

import (
 "database/sql"

 "github.com/neo4j/neo4j-go-driver/v4/neo4j"
)

func migrateTable(mysqlDB *sql.DB, neo4jDriver neo4j.Driver, migration MigrationConfig) error {
 rows, err := fetchData(mysqlDB, migration.SourceTable)
 if err != nil {
  return err
 }
 defer rows.Close()

 for rows.Next() {
  // Implementace pÅ™evodu Å™Ã¡dku z MySQL do Neo4j uzlu
 }

 return nil
}
```

../alex-visualizer/internal/mysql/client.go

```
package mysql

import (
    "database/sql"
    _ "github.com/go-sql-driver/mysql"
    "github.com/spf13/viper"
)

type Client struct {
    db *sql.DB
}

func NewClient() (*Client, error) {
    dsn := viper.GetString("mysql.dsn")
    db, err := sql.Open("mysql", dsn)
    if err != nil {
        return nil, err
    }

    return &Client{db}, nil
}

func (c *Client) Close() error {

    return c.db.Close()

}

func (c *Client) FetchData() ([]map[string]interface{}, error) {

    rows, err := c.db.Query("SELECT * FROM your_table")

    if err != nil {
        return nil, err
    }

    defer rows.Close()



    columns, err := rows.Columns()

    if err != nil {
        return nil, err
    }



    var results []map[string]interface{}
```

```
    for rows.Next() {

        row := make(map[string]interface{})

        columnPointers := make([]interface{}, len(columns))

        for i := range columns {
            columnPointers[i] = new(interface{})
        }

        if err := rows.Scan(columnPointers...); err != nil {
            return nil, err
        }

        for i, colName := range columns {
            row[colName] = *(columnPointers[i].(*interface{}))
        }

        results = append(results, row)
    }

    return results, nil
}
```

../alex-visualizer/internal/neo4j/client.go

```
package neo4j

import (
    "github.com/neo4j/neo4j-go-driver/v5/neo4j"
    "github.com/spf13/viper"
)

type Client struct {
    driver neo4j.DriverWithContext
}

func NewClient() (*Client, error) {
    uri := viper.GetString("neo4j.uri")
    username := viper.GetString("neo4j.username")
    password := viper.GetString("neo4j.password")

    driver, err := neo4j.NewDriverWithContext(uri, neo4j.BasicAuth(username, password, ""))
    if err != nil {
        return nil, err
    }

    return &Client{driver}, nil
}
```

../alex-visualizer/internal/visualization/visualize.go

```
package analysis

import (
    "neo4j-mysql-bridge/internal/mysql"
    "github.com/neo4j/neo4j-go-driver/v5/neo4j"
)
type Transformer struct {
    // add any necessary fields
}

func NewTransformer() *Transformer {
    return &Transformer{}
}

func (t *Transformer) TransformData(mysqlData []mysql.DataType) []neo4j.DataType {
    // Implement the transformation logic here
    var neo4jData []neo4j.DataTypes
    // Example: loop through mysqlData and convert to neo4jData
    return neo4jData
}
```