Fundação Getúlio Vargas
School of Applied Mathematics

Vertical Life Database

Nicholas Farrel Ferraz de Faria

1 Introduction

Vertical Life is an application designed for climbers that provides a comprehensive log of climbing sectors and routes. Users can track their friends' ascents and also share their own. Each route in the application is classified into one of the three most popular climbing styles: *Sport*[1], *Boulder*[2], and *Trad*[3]. The system also allows users to log their ascents and rate the experience using a 1 to 5 star scale.
To better understand how the app works, the next section outlines the database requirements.

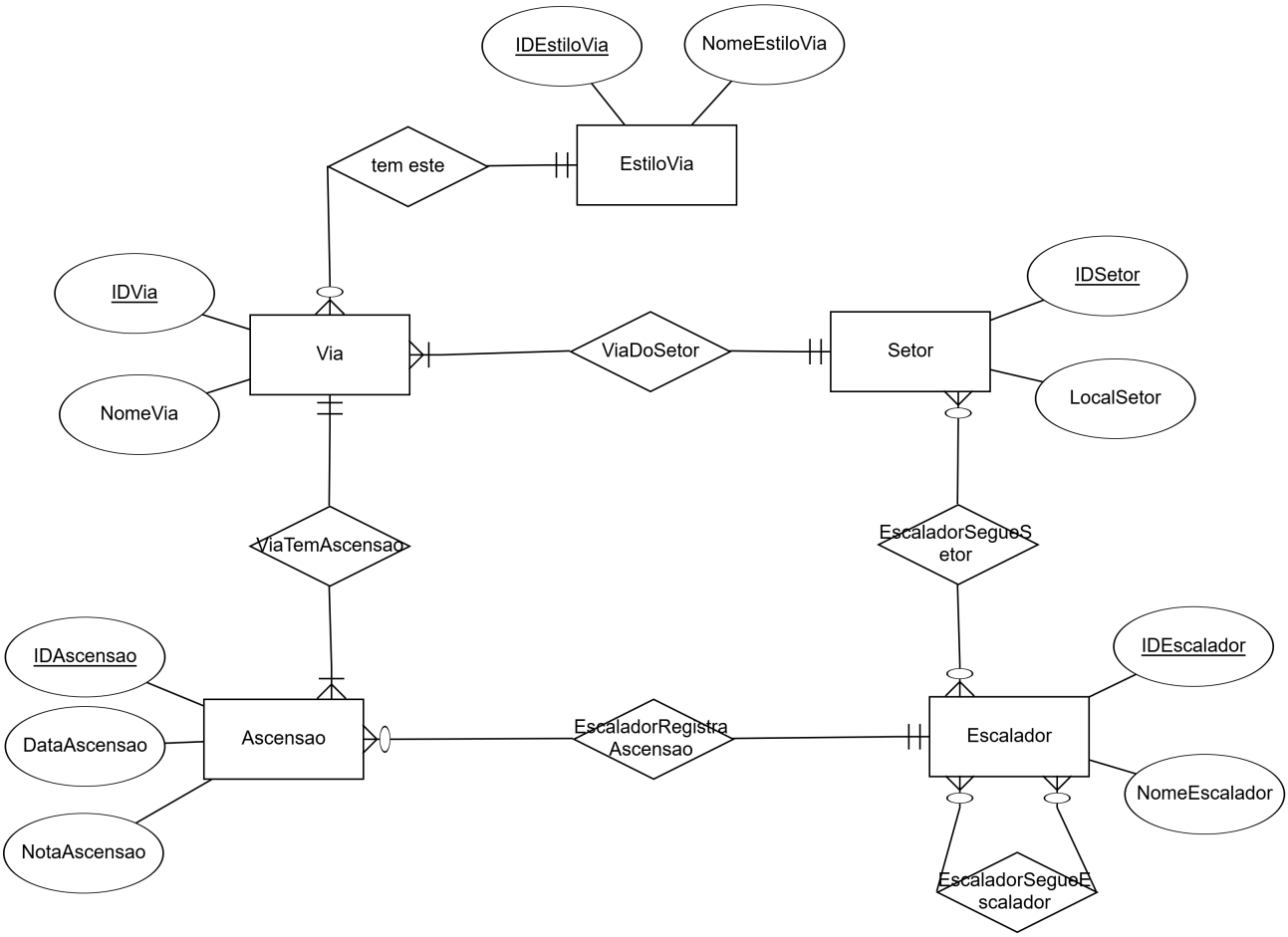
2 Requirements

1. Strong entity **Climber** with attributes **ClimberID** and **ClimberName**. **Climber** represents the app's user.
2. Strong entity **Route** with attributes **RouteID** and **RouteName**. **Route** represents a climbing route.
3. Weak entity **RouteStyle** used for classification, with attributes **RouteStyleID** and **RouteStyleName**. It is used to categorize **Route** instances according to their climbing style. Each **RouteStyle** instance represents a fixed value like *Sport*, *Trad*, or *Boulder*, and is used solely to identify the climbing type of a **Route**.
4. Strong entity **Ascent** with attributes **AscentID**, **AscentDate**, and **AscentRating** (an integer from 1 to 5). **Ascent** represents the log of a climber completing a specific **Route**.
5. Strong entity **Sector** with attributes **SectorID** and **SectorLocation**, representing a geographical or organizational grouping of **Route** instances. **SectorLocation** is composed of latitude and longitude.
6. Simple relation **ClimberLogsAscent** between **Climber** and **Ascent**, with a **1:N** relationship (optional for **Climber**, mandatory for **Ascent**). This relation represents a climber logging an ascent. Each **Climber** may be associated with many or no **Ascents**, while every **Ascent** must be associated with exactly one **Climber**. The only way to insert new **Route** or **Sector** instances into the system is by registering a related **Ascent**.
7. Simple relation **RouteHasAscent** between **Route** and **Ascent**, with a **1:N** relationship (mandatory for both). This relation associates a **Route** with the **Ascents** that have been logged on it. A **Route** only exists if it is linked to at least one **Ascent**.
8. Simple relation **RouteHasStyle** between **Route** and **RouteStyle**, with a **N:1** relationship (optional for **Route**, mandatory for **RouteStyle**). This relation classifies each **Route** according to its style. Some **RouteStyle** categories might not have any associated **Route** instances.

[1] A climbing style where permanent protections are pre-installed in the rock (bolts, anchors)
[2] A climbing style practiced on short boulders (usually up to 5 meters) with crash pads for protection
[3] A climbing style where the climber places protection devices (friends, nuts, etc.) during the ascent

9. Simple relation **RouteInSector** between **Route** and **Sector**, with a **N:1** relationship (mandatory for both). This relation links a **Route** to its **Sector**. A **Sector** only exists in the system if there is at least one **Route** (with at least one **Ascent**) linked to it.
10. Simple relation **ClimberFollowsSector** between **Climber** and **Sector**, with a **N:N** relationship (optional for both). This relation allows a **Climber** to follow a **Sector**, receiving updates about new **Ascents** in that area.
11. Simple relation **ClimberFollowsClimber**, a self-relationship in **Climber**, with a **N:N** cardinality (optional for both). This relation allows a **Climber** to follow another, receiving updates about their logged **Ascents**.

3 ER Diagram



4 Relational Diagram