Yes, it is absolutely possible to create such a project using only free and open-source software. There are many free and open-source tools and frameworks available that can help you build and deploy microservices.

You can create the microservices one by one, starting with the most foundational ones and gradually building upon them. Here's a suggested order:

1. User Service: Since user registration, authentication, and profile management are essential for the overall functionality of the bot, it's a good starting point.
2. Hobbies Service: Once the user service is in place, you can move on to implementing the hobbies service. This will allow users to add, edit, and delete hobbies associated with their profiles.
3. Geolocation Service: With the user and hobbies services established, you can proceed to implement the geolocation service. This service will handle geolocation-related operations, such as storing and managing geolocation data associated with user profiles.
4. Matching Service: After the previous services are implemented, you can focus on building the matching service. This service will perform the matching algorithms based on shared hobbies and proximity. It will communicate with the hobbies and geolocation services to fetch relevant data.
5. Messaging Service: Lastly, you can implement the messaging service, which will handle user interactions and communication. This service will manage conversations and message exchanges between users, enabling real-time communication capabilities.

By following this order, you'll gradually build the core functionality of the bot, starting with user management and progressing towards the more advanced features like matching and messaging.

Remember to consider the principles of microservice architecture, such as loose coupling, independent deployment, and separation of concerns, as you design and implement each service. This will enable you to develop and deploy each microservice independently, making it easier to iterate and scale your application.

Certainly! Here's a step-by-step high-level plan for the Telegram bot project with microservices:

Step 1: User Service

* Timeline: 1-2 weeks (depending on your familiarity with the technologies)
* Technologies:
  + Programming Language: Go (Golang)
  + Framework: Gin (a lightweight web framework for Go)
  + Database: PostgreSQL
  + Libraries/Tools: GORM (for database ORM), JWT (for authentication), bcrypt (for password hashing)

Step 2: Hobbies Service

* Timeline: 1-2 weeks
* Technologies:
  + Programming Language: Go (Golang)
  + Framework: Gin
  + Database: PostgreSQL
  + Libraries/Tools: GORM

Step 3: Geolocation Service

* Timeline: 1-2 weeks
* Technologies:
  + Programming Language: Go (Golang)
  + Framework: Gin
  + Database: PostgreSQL with PostGIS extension (for geospatial data)
  + Libraries/Tools: GORM

Step 4: Matching Service

* Timeline: 2-3 weeks
* Technologies:
  + Programming Language: Go (Golang)
  + Framework: Gin
  + Database: PostgreSQL
  + Libraries/Tools: GORM

Step 5: Messaging Service

* Timeline: 2-3 weeks
* Technologies:
  + Programming Language: Go (Golang)
  + Framework: Gin
  + Database: PostgreSQL
  + Libraries/Tools: GORM, Websockets (for real-time communication)

Overall Timeline: Approximately 8-12 weeks (can vary based on your learning pace and project complexity)

Note: The technologies mentioned above are all free and open-source.

During the development process, you can follow these general steps for each microservice:

1. Design the API endpoints and data models for the microservice.
2. Implement the database schema and set up database connectivity using GORM.
3. Develop the API endpoints for CRUD operations and other required functionalities.
4. Test the microservice thoroughly, including unit tests and integration tests.
5. Deploy the microservice to a hosting platform of your choice, such as Heroku or AWS.

Remember to break down each step into smaller tasks and allocate time accordingly. It's also recommended to learn and understand the basics of each technology before starting the implementation.

Lastly, keep in mind that this plan provides a high-level overview, and you may encounter challenges or require additional time for troubleshooting and learning. Don't hesitate to refer to official documentation, online tutorials, and community resources for guidance throughout the project.