Dynamic Web Map Services

Summer School on Digital Humanities

Web site: https://bit.ly/dt4h-gis

Augusto Ciuffoletti

9 giugno 2025

Dynamic Web Map Services

- A local application does not facilitate map sharing
- We need an interactive web-based map service
- Web Mapping enables cartographers to maintain a shared map

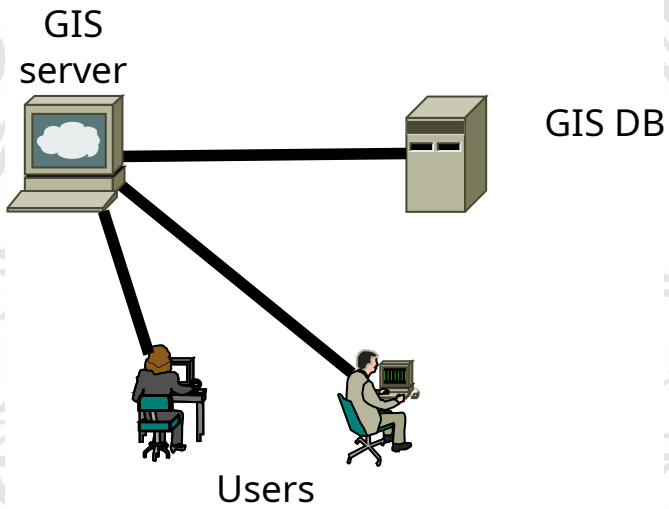
Dynamic Web Map Services

- A local application does not facilitate map sharing
- We need an interactive web-based map service

- Web Mapping enables cartographers to maintain a shared map

Dynamic Web Map Services

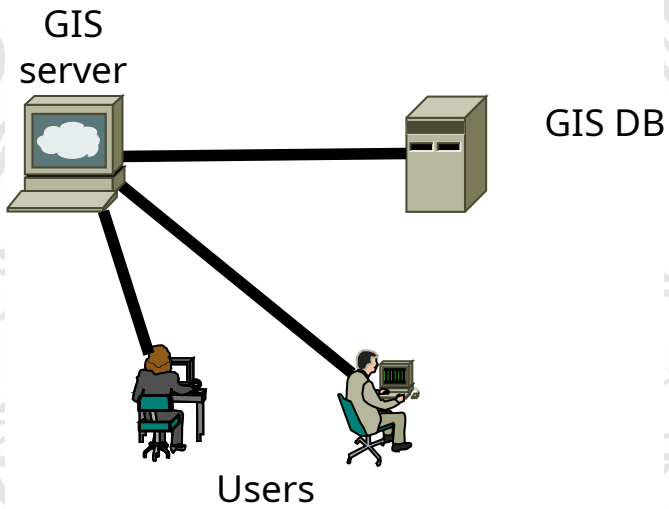
- A local application does not facilitate map sharing
- We need an interactive web-based map service



- Web Mapping enables cartographers to maintain a shared map
  - The cartographer accesses the mapping service via a web browser
  - The server generates a web page containing an up-to-date map
  - Embedded links allow the cartographer to navigate and update data
  - The cartographer can see the effect of their changes

Dynamic Web Map Services

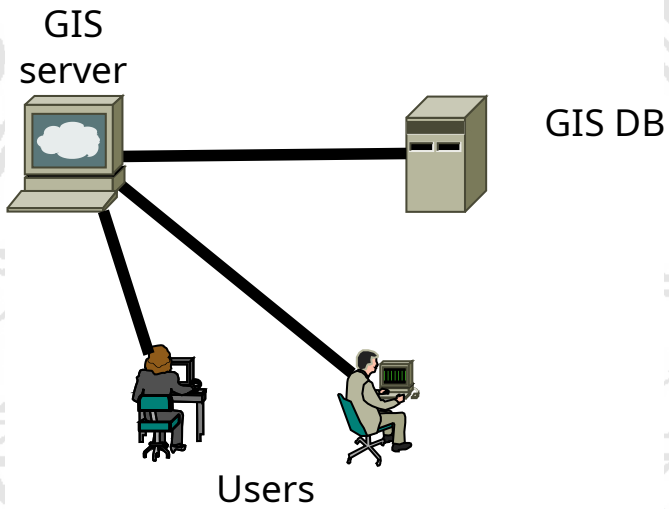
- A local application does not facilitate map sharing
- We need an interactive web-based map service



- Web Mapping enables cartographers to maintain a shared map
  - The cartographer accesses the mapping service via a web browser
  - The server generates a web page integrating the map
  - Embedded code connects to a remote database to retrieve and update data
  - The cartographer can review the new or input new data

Dynamic Web Map Services

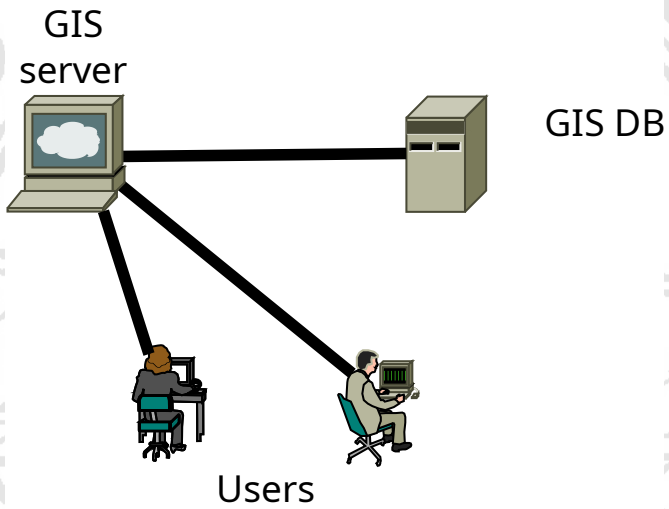
- A local application does not facilitate map sharing
- We need an interactive web-based map service



- Web Mapping enables cartographers to maintain a shared map
  - The cartographer accesses the mapping service via a web browser
  - The server generates a web page integrating the map
  - Embedded code connects to a remote database to retrieve and update data
  - The cartographer can modify the view or input new data

Dynamic Web Map Services

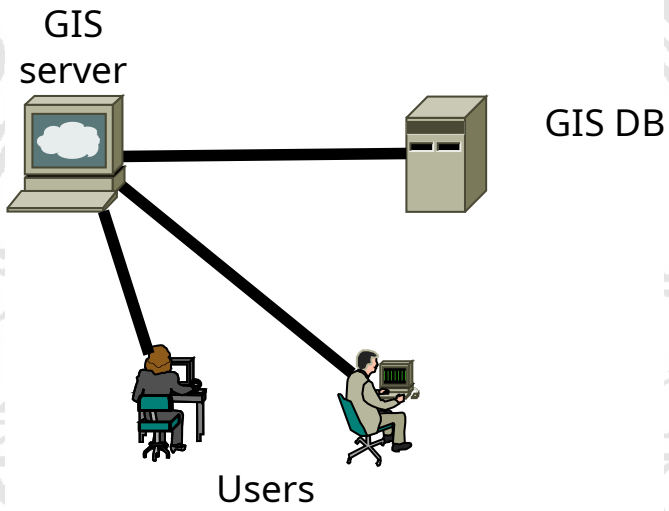
- A local application does not facilitate map sharing
- We need an interactive web-based map service



- Web Mapping enables cartographers to maintain a shared map
  - The cartographer accesses the mapping service via a web browser
  - The server generates a web page integrating the map
  - Embedded code connects to a remote database to retrieve and update data
  - The cartographer can modify the view or input new data

Dynamic Web Map Services

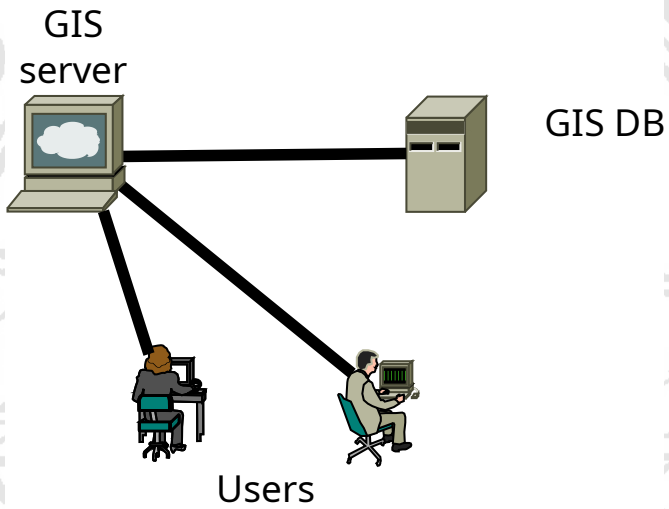
- A local application does not facilitate map sharing
- We need an interactive web-based map service



- Web Mapping enables cartographers to maintain a shared map
  - The cartographer accesses the mapping service via a web browser
  - The server generates a web page integrating the map
  - Embedded code connects to a remote database to retrieve and update data
  - The cartographer can modify the view or input new data

Dynamic Web Map Services

- A local application does not facilitate map sharing
- We need an interactive web-based map service



- Web Mapping enables cartographers to maintain a shared map
  - The cartographer accesses the mapping service via a web browser
  - The server generates a web page integrating the map
  - Embedded code connects to a remote database to retrieve and update data
  - The cartographer can modify the view or input new data

Web GIS vs. Desktop GIS Applications

- Compared to a desktop GIS application (like QGIS):
  - No installation required
  - No dependency on local computing power
  - Platform-independent (works on any OS)
  - Responsive design (different devices: PC, tablet, smartphone)
  - Designed for sharing — requires access control mechanisms
- Developing such a dynamic application from scratch is a specialized JavaScript library

Web GIS vs. Desktop GIS Applications

- Compared to a desktop GIS application (like QGIS):
  - No installation required
  - No dependency on local computing power
  - Platform-independent (works on any OS)
  - Responsive design: works on different devices (PC, tablet, smartphone)
  - Designed for sharing — requires access control mechanisms
- Developing such a dynamic application from scratch is a specialized JavaScript library

Web GIS vs. Desktop GIS Applications

- Compared to a desktop GIS application (like QGIS):
  - No installation required
  - No dependency on local computing power
  - Platform-independent (works on any OS)
  - Responsive design for different devices (PC, tablet, smartphone)
  - Designed for sharing — requires access control mechanisms
- Developing such a dynamic application from scratch is a specialized JavaScript library

Web GIS vs. Desktop GIS Applications

- Compared to a desktop GIS application (like QGIS):
  - No installation required
  - No dependency on local computing power
  - Platform-independent (works on any OS)
  - Responsive design for different devices (PC, tablet, smartphone)
  - Designed for sharing → requires access control mechanisms
- Developing such a dynamic application requires a specialized JavaScript library

Web GIS vs. Desktop GIS Applications

- Compared to a desktop GIS application (like QGIS):
  - No installation required
  - No dependency on local computing power
  - Platform-independent (works on any OS)
  - Responsive design for different devices (PC, tablet, smartphone)
  - Designed for sharing — requires access control mechanisms
- Developing such a dynamic application requires a specialized JavaScript library

Web GIS vs. Desktop GIS Applications

- Compared to a desktop GIS application (like QGIS):
  - No installation required
  - No dependency on local computing power
  - Platform-independent (works on any OS)
  - Responsive design for different devices (PC, tablet, smartphone)
  - Designed for sharing—requires access control mechanisms
- Developing such a dynamic application requires a specialized JavaScript library

Web GIS vs. Desktop GIS Applications

- Compared to a desktop GIS application (like QGIS):
  - No installation required
  - No dependency on local computing power
  - Platform-independent (works on any OS)
  - Responsive design for different devices (PC, tablet, smartphone)
  - Designed for sharing—requires access control mechanisms
- Developing such a dynamic application requires a specialized JavaScript library

Tools for Web Maps: JavaScript Libraries

- JavaScript enables complex functionalities in web pages
- The Leaflet library allows web pages to interact with GIS servers and store user data
- Users can modify and update the map interactively
- This setup creates a complex architecture:
- We will explore OpenStreetMap's tool is implemented using the *Leaflet* library

Tools for Web Maps: JavaScript Libraries

- JavaScript enables complex functionalities in web pages
- The Leaflet library allows web pages to interact with GIS servers and store user data
- Users can modify and update the map interactively
- This setup creates a complex architecture:
- We will explore OpenStreetMap which is implemented using the *Leaflet* library

Tools for Web Maps: JavaScript Libraries

- JavaScript enables complex functionalities in web pages
- The Leaflet library allows web pages to interact with GIS servers and store user data
- Users can modify and update the map interactively
- This setup creates a complex architecture:
  - The user downloads a web page (designed by the cartographer)
  - The web page interacts with the GIS server and the user data repository
- We will explore OpenStreetMap.it which is implemented using the *Leaflet* library

Tools for Web Maps: JavaScript Libraries

- JavaScript enables complex functionalities in web pages
- The Leaflet library allows web pages to interact with GIS servers and store user data
- Users can modify and update the map interactively
- This setup creates a complex architecture:
  - The user downloads a web page (designed by the cartographer)
  - The page interacts with a PostGIS server and a raster data repository
- We will explore OpenStreetMap, which is implemented using the *Leaflet* library

Tools for Web Maps: JavaScript Libraries

- JavaScript enables complex functionalities in web pages
- The Leaflet library allows web pages to interact with GIS servers and store user data
- Users can modify and update the map interactively
- This setup creates a complex architecture:
  - The user downloads a web page (designed by the cartographer)
  - The page interacts with a PostGIS server and a raster data repository
- We will explore OpenStreetMap, which is implemented using the *Leaflet* library

Tools for Web Maps: JavaScript Libraries

- JavaScript enables complex functionalities in web pages
- The Leaflet library allows web pages to interact with GIS servers and store user data
- Users can modify and update the map interactively
- This setup creates a complex architecture:
  - The user downloads a web page (designed by the cartographer)
  - The page interacts with a PostGIS server and a raster data repository
- We will explore OpenStreetMap, which is implemented using the *Leaflet* library

Tools for Web Maps: JavaScript Libraries

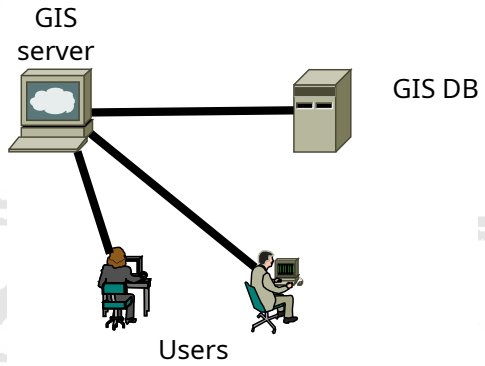
- JavaScript enables complex functionalities in web pages
- The Leaflet library allows web pages to interact with GIS servers and store user data
- Users can modify and update the map interactively
- This setup creates a complex architecture:
  - The user downloads a web page (designed by the cartographer)
  - The page interacts with a PostGIS server and a raster data repository
- We will explore OpenStreetMap, which is implemented using the *Leaflet* library

Example of an Open Web Map Service: OpenStreetMap

- The OpenStreetMap server (www.openstreetmap.org) renders a dynamic map in the browser, drawing data from a public database

Example of an Open Web Map Service: OpenStreetMap

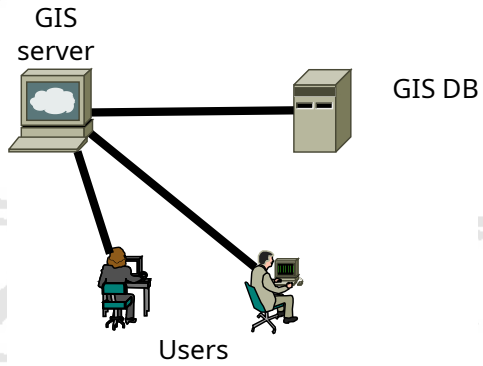
- The OpenStreetMap server (www.openstreetmap.org) renders a dynamic map in the browser, drawing data from a public database



- Public Collaboration
  - Anyone can edit, access, and update the database – all changes are publicly visible
  - There is no private or personal workspace
- Using the built-in Id editor

Example of an Open Web Map Service: OpenStreetMap

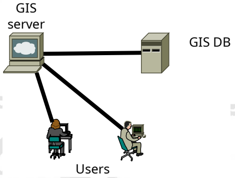
- The OpenStreetMap server (www.openstreetmap.org) renders a dynamic map in the browser, drawing data from a public database



- Public Collaboration:
  - Anyone with write access can update the database—all changes are publicly visible
  - There is no option for a private workspace
- Using the built-in Id editor

Example of an Open Web Map Service: OpenStreetMap

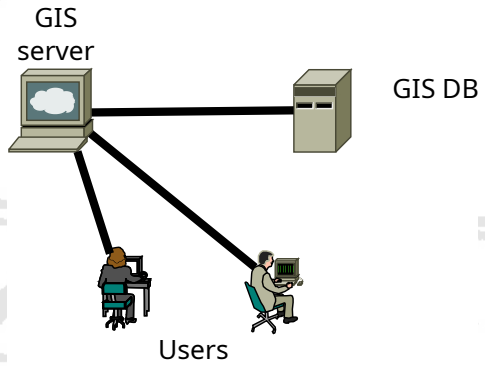
- The OpenStreetMap server (www.openstreetmap.org) renders a dynamic map in the browser, drawing data from a public database



- Public Collaboration:
  - Anyone with write access can update the database—all changes are publicly visible
  - There is no option for a private workspace
- Using the built-in Id editor:

Example of an Open Web Map Service: OpenStreetMap

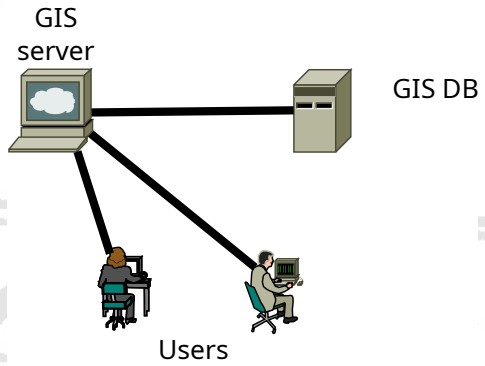
- The OpenStreetMap server (www.openstreetmap.org) renders a dynamic map in the browser, drawing data from a public database



- Public Collaboration:
  - Anyone with write access can update the database—all changes are publicly visible
  - There is no option for a private workspace
- Using the built-in **Id** editor:
  - Easily create features like a bar, swimming pool, or street
  - Save changes can be viewed immediately by everyone

Example of an Open Web Map Service: OpenStreetMap

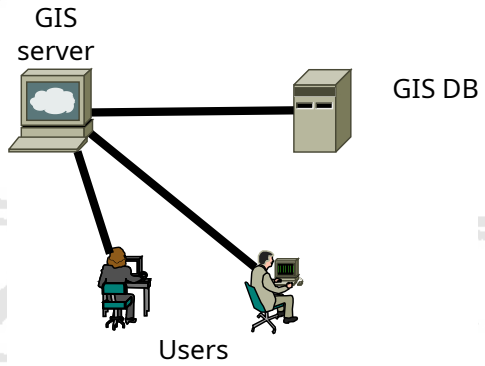
- The OpenStreetMap server (www.openstreetmap.org) renders a dynamic map in the browser, drawing data from a public database



- Public Collaboration:
  - Anyone with write access can update the database—all changes are publicly visible
  - There is no option for a private workspace
- Using the built-in **Id** editor:
  - Easily create features like a bar, swimming pool, or street
  - **Save** changes cautiously—they become immediately visible to everyone

Example of an Open Web Map Service: OpenStreetMap

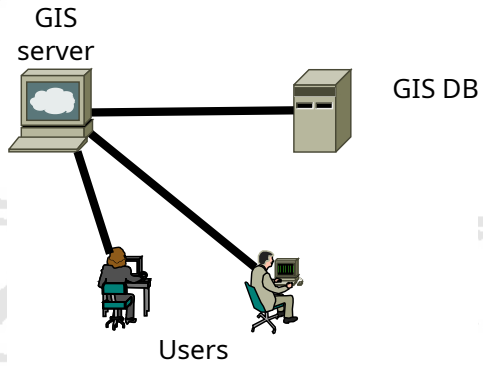
- The OpenStreetMap server (www.openstreetmap.org) renders a dynamic map in the browser, drawing data from a public database



- Public Collaboration:
  - Anyone with write access can update the database—all changes are publicly visible
  - There is no option for a private workspace
- Using the built-in **Id** editor:
  - Easily create features like a bar, swimming pool, or street
  - **Save** changes cautiously—they become immediately visible to everyone

Example of an Open Web Map Service: OpenStreetMap

- The OpenStreetMap server (www.openstreetmap.org) renders a dynamic map in the browser, drawing data from a public database



- Public Collaboration:
  - Anyone with write access can update the database—all changes are publicly visible
  - There is no option for a private workspace
- Using the built-in **Id** editor:
  - Easily create features like a bar, swimming pool, or street
  - **Save** changes cautiously—they become immediately visible to everyone

Getting Started with OpenStreetMap

- Open a browser and visit OpenStreetMap
- To access the service:
  - Sign in with an existing account or a third-party service (e.g. Google, Microsoft, Facebook, ...)
  - Register a new account

Getting Started with OpenStreetMap

- Open a browser and visit OpenStreetMap
- To access the service:
  - **Sign in** with an existing account or a third-party service (e.g. Google, Microsoft, Facebook) or
  - **Register** a new account

Getting Started with OpenStreetMap

- Open a browser and visit OpenStreetMap
- To access the service:
  - **Sign in** with an existing account or a third-party service (e.g. Google, Microsoft, Facebook) or
  - **Register** a new account

Getting Started with OpenStreetMap

- Open a browser and visit OpenStreetMap
- To access the service:
  - **Sign in** with an existing account or a third-party service (e.g. Google, Microsoft, Facebook) or
  - **Register** a new account

Creating a Point Feature in OpenStreetMap

- To add a point feature (but do not press **Save**):
  - Zoom in using the trackpad until **Edit** is enabled
  - Select the **Edit** option (opens the *iD* editor)
  - Zoom until the "Zoom in to edit" banner disappears
  - Click the point icon in the top toolbar (it turns blue)
  - Click on the map to place the point
  - Choose a feature type (e.g. Cafe, Restaurant, ...)
  - Fill in several attributes
  - Press Undo (back arrow) instead of Save

Creating a Point Feature in OpenStreetMap

- To add a point feature (but do not press **Save**):
  - Zoom in using the trackpad until **Edit** is enabled
  - Select the **Edit** option (opens the *iD* editor)
  - Zoom until the "Zoom in to edit" banner disappears
  - Click the Point tool in the top toolbar (it turns blue)
  - Click on the map to place the point
  - Choose a feature type (e.g. Cafe, Restaurant, Hotel, ...)
  - Fill in relevant attributes
  - Press Undo (back arrow) to avoid saving

Creating a Point Feature in OpenStreetMap

- To add a point feature (but do not press **Save**):
  - Zoom in using the trackpad until **Edit** is enabled
  - Select the **Edit** option (opens the *iD* editor)
  - Zoom until the "Zoom in to edit" banner disappears
  - Click the **Point** tool in the top toolbar (it turns blue)
  - Click on the map to place the point
  - Choose a feature type (e.g. Cafe, Restaurant, Fast food)
  - Fill in relevant attributes
  - Press Undo (back arrow) to remove

Creating a Point Feature in OpenStreetMap

- To add a point feature (but do not press **Save**):
  - Zoom in using the trackpad until **Edit** is enabled
  - Select the **Edit** option (opens the *iD* editor)
  - Zoom until the "Zoom in to edit" banner disappears
  - Click the **Point** tool in the top toolbar (it turns blue)
  - Click on the map to place the point
  - Choose a feature type (e.g. Cafe) from the left sidebar
  - Fill in relevant attributes
  - Press Undo (back arrow) to remove

Creating a Point Feature in OpenStreetMap

- To add a point feature (but do not press **Save**):
  - Zoom in using the trackpad until **Edit** is enabled
  - Select the **Edit** option (opens the *iD* editor)
  - Zoom until the "Zoom in to edit" banner disappears
  - Click the **Point** tool in the top toolbar (it turns blue)
  - Click on the map to place the point
  - Choose a feature type (e.g. Cafe) from the left sidebar
  - Fill in relevant attributes
  - Press Undo (back arrow) instead of Save

Creating a Point Feature in OpenStreetMap

- To add a point feature (but do not press **Save**):
  - Zoom in using the trackpad until **Edit** is enabled
  - Select the **Edit** option (opens the *iD* editor)
  - Zoom until the "Zoom in to edit" banner disappears
  - Click the **Point** tool in the top toolbar (it turns blue)
  - Click on the map to place the point
  - Choose a feature type (e.g., **Cafe**) from the left sidebar
  - Fill in relevant attributes
  - Press **Undo** (back arrow next to **Save**)

Creating a Point Feature in OpenStreetMap

- To add a point feature (but do not press **Save**):
  - Zoom in using the trackpad until **Edit** is enabled
  - Select the **Edit** option (opens the *iD* editor)
  - Zoom until the "Zoom in to edit" banner disappears
  - Click the **Point** tool in the top toolbar (it turns blue)
  - Click on the map to place the point
  - Choose a feature type (e.g., **Café**) from the left sidebar
  - Fill in relevant attributes
  - Press **Undo** (back arrow next to "Save")

Creating a Point Feature in OpenStreetMap

- To add a point feature (but do not press **Save**):
  - Zoom in using the trackpad until **Edit** is enabled
  - Select the **Edit** option (opens the *iD* editor)
  - Zoom until the "Zoom in to edit" banner disappears
  - Click the **Point** tool in the top toolbar (it turns blue)
  - Click on the map to place the point
  - Choose a feature type (e.g., **Café**) from the left sidebar
  - Fill in relevant attributes
  - Press **Undo** (back arrow next to "Save")

Creating a Point Feature in OpenStreetMap

- To add a point feature (but do not press **Save**):
  - Zoom in using the trackpad until **Edit** is enabled
  - Select the **Edit** option (opens the *iD* editor)
  - Zoom until the "Zoom in to edit" banner disappears
  - Click the **Point** tool in the top toolbar (it turns blue)
  - Click on the map to place the point
  - Choose a feature type (e.g., **Café**) from the left sidebar
  - Fill in relevant attributes
  - Press **Undo** (back arrow next to "Save")

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
- Keyboard shortcuts:
- Pressing Save commits changes to OpenStreetMap—please refrain from saving test edits!

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
- Keyboard shortcuts:
- Pressing **Save** commits changes to OpenStreetMap—please refrain from saving test edits

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
  - Keyboard shortcuts:
  - Pressing Save commits changes to OpenStreetMap—please refrain from saving test edits!

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Move
    - Rotate
    - Reflect
    - Duplicate
    - Delete
  - Keyboard shortcuts
  - Pressing Save commits changes to OpenStreetMap—please refrain from saving test edits

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Convert to a circle
    - Convert to a point
    - Align angles to 90°
    - Tidy or reverse
- Keyboard shortcuts
- Pressing Save commits changes to OpenStreetMap—please refrain from saving test edits

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Convert to a circle
    - Convert to a point
    - Align angles to 90°
    - Flip or rotate
- Keyboard shortcuts
- Pressing Save commits changes to OpenStreetMap—please refrain from saving test edits

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Convert to a circle
    - Convert to a point
    - Align angles to 90°
    - Flip or rotate
- Keyboard shortcuts:
- Pressing Save commits changes to OpenStreetMap—please refrain from saving test edits!

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Convert to a circle
    - Convert to a point
    - Align angles to 90°
    - Flip or rotate
- Keyboard shortcuts
- Pressing Save commits changes to OpenStreetMap—please refrain from saving test edits

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Convert to a circle
    - Convert to a point
    - Align angles to 90°
    - Flip or rotate
- Keyboard shortcuts:
  - Ctrl+C / Ctrl+V to copy and paste
  - Ctrl+Z to undo changes
- Pressing Save commits changes to OpenStreetMap—please refrain from saving test edits!

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Convert to a circle
    - Convert to a point
    - Align angles to 90°
    - Flip or rotate
- Keyboard shortcuts:
  - Ctrl+C / Ctrl+V to copy and paste
  - Ctrl+Z to undo changes
- Pressing **Save** commits changes to OpenStreetMap—please refrain from saving test edits!

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Convert to a circle
    - Convert to a point
    - Align angles to 90°
    - Flip or rotate
- Keyboard shortcuts:
  - Ctrl+C / Ctrl+V to copy and paste
  - Ctrl+Z to undo changes
- Pressing **Save** commits changes to OpenStreetMap—please refrain from saving test edits!

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Convert to a circle
    - Convert to a point
    - Align angles to 90°
    - Flip or rotate
- Keyboard shortcuts:
  - Ctrl+C / Ctrl+V to copy and paste
  - Ctrl+Z to undo changes
- Pressing **Save** commits changes to OpenStreetMap—please refrain from saving test edits!

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Convert to a circle
    - Convert to a point
    - Align angles to 90°
    - Flip or rotate
- Keyboard shortcuts:
  - Ctrl+C / Ctrl+V to copy and paste
  - Ctrl+Z to undo changes
- Pressing **Save** commits changes to OpenStreetMap—please refrain from saving test edits

Lab Activity

- Scenario: South of Pescara lies "Francavilla al Mare," a seaside resort town
  - Locate "Lido Merope"
  - Add an Area for the beach
  - Set Beach Resort as the feature type
  - Set the Name field to "Spiaggia del Lido Merope"
  - Undo...