Kapitel 4
Git in der SAPUI5-Entwicklung

Für die meisten Unternehmen ist der Einsatz einer Versionskontrolle im Zuge des Softwareentwicklungsprozesses nicht mehr wegzudenken. Auch im SAPUI5-Umfeld ist der Einsatz von solchen Tools relevant, besonders wenn es darum geht, im Team gemeinsam an verschiedenen Applikationen zu entwickeln.

In diesem Kapitel betrachten wir den Einsatz von Versionsverwaltungswerkzeugen in der SAPUI5-Entwicklung im Detail. Da Sie für die Einbindung von Git in die unterschiedlichen Entwicklungsumgebungen Repositorys benötigen, sehen wir uns in Abschnitt 4.1, »Erstellen eines Git-Repositorys«, kurz an, wie Sie ein Repository anlegen. In Abschnitt 4.2, »Einbindung in die SAP Web IDE«, und Abschnitt 4.3, »Einbindung in SAP Business Application Studio«, gehen wir auf die beiden Entwicklungsumgebungen der SAP BTP, die SAP Web IDE und SAP Business Application Studio, ein. Außerdem beschäftigen wir uns in Abschnitt 4.4, »Einbindung in Visual Studio Code«, mit der lokalen IDE Visual Studio Code, die Ihnen als weitere alternative Entwicklungsumgebung für SAPUI5-Projekte zur Verfügung steht.

Vor allem dann, wenn mehrere Personen an der Entwicklung einer Anwendung beteiligt sind, ist es hilfreich, auf solche Tools zurückzugreifen. Aber auch, wenn nur eine Person an einer Anwendung arbeitet, ist die Verwendung einer Versionsverwaltung von Nutzen. So können Sie beispielsweise eine CI/CD-Pipeline einrichten. Zusätzlich haben Sie eine Sicherung der Anwendung im jeweiligen Repository.

Während Sie beispielsweise in der ABAP-Entwicklung immer den aktuellen Stand der Sourcen sehen und diese Sourcen durch die Bearbeitung von einer Person gesperrt werden, ist dies in der SAPUI5-Entwicklung nicht so. Hier sehen Sie nur Ihre lokalen Datenstände. Erst durch das Ausführen eines Commits und durch Pushen werden die Änderungen in das Remote-Repository übertragen, von wo aus sie durch einen Pull wieder in die IDE geladen werden können.

4.1 Erstellen eines Git-Repositorys

Um die verschiedensten Softwareprojekte mit Versionsverwaltungswerkzeugen zu verwalten, ist es notwendig, Repositorys anzulegen. Ob Sie hier GitHub, Bitbucket, das Repository in der SAP BTP oder ein anderes Repository verwenden, obliegt Ihnen. Für dieses Beispiel verwenden wir GitHub. Ist ein Repository angelegt, ist die weitere Vorgehensweise unabhängig vom verwendeten Anbieter.

Repository erstellen

Zunächst loggen Sie sich in GitHub ein und wechseln in den Reiter **Repositories**. Hier würden bereits bestehende Repositorys angezeigt. Zudem haben Sie die Möglichkeit, mit **New** ein neues Repository anzulegen (siehe Abbildung 4.1).

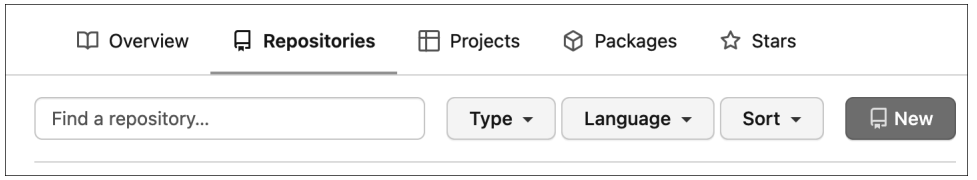

Abbildung 4.1 GitHub: Reiter »Repositories«

Verpflichtende Angaben

Nach einem Klick auf **New** werden Sie auf ein Formular weitergeleitet, in das die für die Anlage notwendigen Informationen einzugeben sind. In diesem Formular müssen Sie den Inhaber sowie den Namen des Repositorys verpflichtend angeben. Optional kann auch eine Beschreibung ergänzt werden.

Optionale Angaben

Zusätzlich ist es möglich, abseits dieser Informationen die Sichtbarkeit festzulegen. Auch können Sie das Repository mit einer Lizenz versehen oder vorab eine **.gitignore**-Datei erstellen. Möchten Sie innerhalb des Repositorys weitere Informationen veranschaulichen, können Sie dazu eine Readme-Datei hinzufügen.

Wie in Abbildung 4.2 erkennbar, haben wir die notwendigen Informationen angegeben und können nun das Repository über die Schaltfläche **Create Repository** erstellen. Bei der Auswahl des Namens ist darauf zu achten, dass dieser innerhalb des Benutzerkontos eindeutig sein muss. Navigieren Sie nach der Erstellung des Repositorys zurück in die Übersicht, sehen Sie, dass das Repository angelegt wurde (siehe Abbildung 4.3).

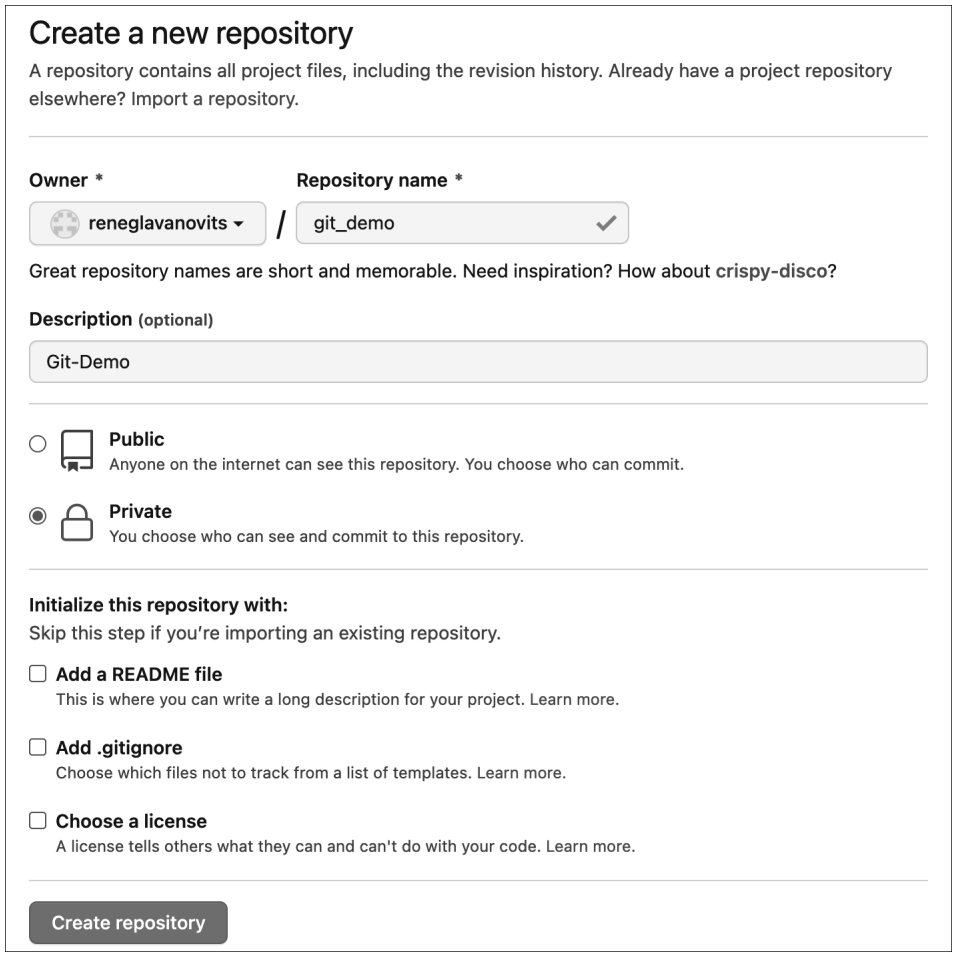

Abbildung 4.2 Repositorys in GitHub erstellen

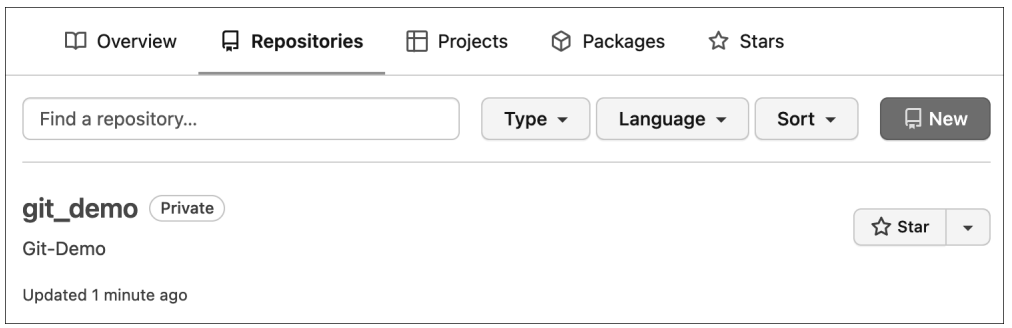

Abbildung 4.3 Übersicht der Repositorys in GitHub

Repository einbinden

Um das neu angelegte Repository zu verwenden, muss es in ein Projekt eingebunden werden. Wie Sie in den Folgeabschnitten sehen werden, bieten die verschiedenen Entwicklungsumgebungen hierzu diverse Funktionen an. Klicken Sie in der Übersicht auf das erstellte Repository, finden Sie auf der Detailseite eine Anleitung, wie Sie es über die Kommandozeile in Projekte einbinden (siehe Abbildung 4.4).

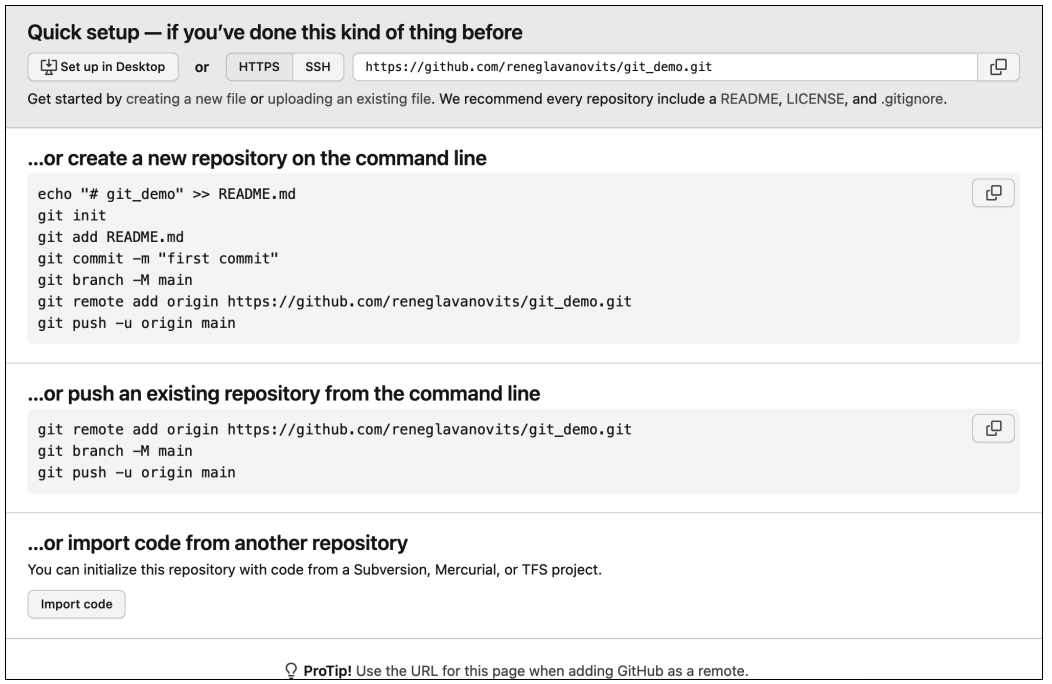

Abbildung 4.4 Details eines Repositorys in GitHub

In den nächsten Abschnitten gehen wir auf die Verwendung von Git in der SAPUI5-Entwicklung ein. Die verwendeten Repositorys legen Sie wie soeben beschrieben an.

4.2 Einbindung in die SAP Web IDE

Nun betrachten wir, wie Sie die Versionsverwaltung in der SAP Web IDE verwenden und welche Möglichkeiten diese Entwicklungsumgebung Ihnen hier bietet. Da die SAP Web IDE eine übersichtliche grafische Git Pane bietet, sind hier keine weiteren Installationen notwendig. Außerdem sind die einzelnen Funktionen, die Ihnen Git bietet, relativ einfach anwendbar.

Git Pane

Die Git Pane ist eine grafische Oberfläche, die das Ausführen von Git-Funktionalitäten über verschiedene Schaltflächen ermöglicht.

Lokales Repository initialisieren

Üblicherweise ist einer der ersten Schritte bei der Entwicklung von Softwareprojekten das Anlegen eines Remote-Repositorys. Ob Sie an dieser Stelle GitHub, Bitbucket oder ein Repository in der SAP BTP verwenden, bleibt Ihnen überlassen. In unserem Fall verwenden wir GitHub. Bevor Sie Ihre Applikation in das Remote-Repository pushen können, müssen Sie allerdings das lokale Repository initialisieren. Wie aus Abbildung 4.5 ersichtlich, erreichen Sie diese Option mit einem Rechtsklick auf das Projekt und dann über den Pfad **Git • Initialize Local Repository**.

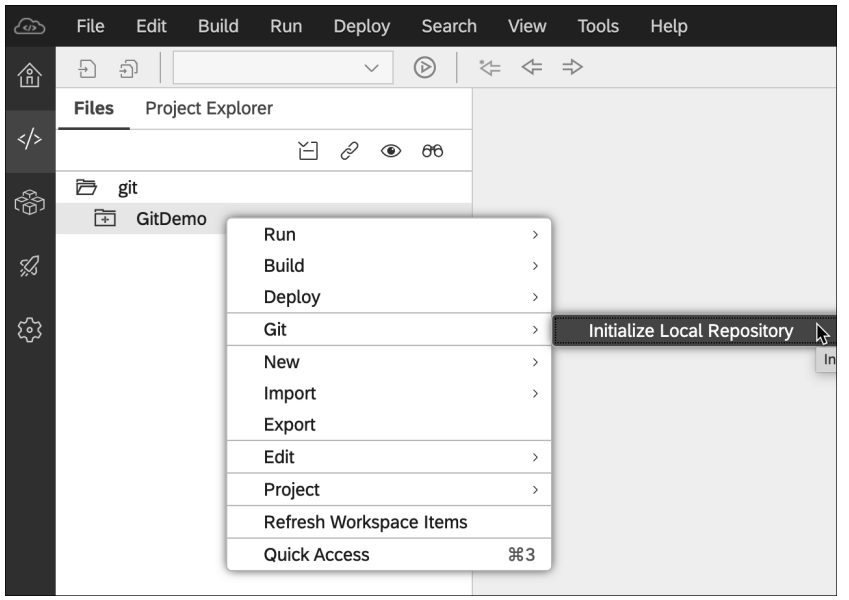

Abbildung 4.5 Lokales Repository initialisieren

Im folgenden Dialog werden Sie aufgefordert, Ihre Benutzerinformationen anzugeben. Danach erhalten Sie eine Erfolgsmeldung, dass Ihr lokales Repository initialisiert wurde. An dieser Stelle könnten Sie bereits alle Funktionen von Git verwenden, allerdings nur lokal.

Remote-Repository setzen

Um Änderungen später auch in das Remote-Repository zu übertragen, ist es erforderlich, dieses einzurichten. Wie Sie in Abbildung 4.6 und Abbildung 4.7 sehen, erfolgt die Anbindung entweder über die Schaltflächen in der Erfolgsmeldung oder den Menüpfad **Git • Set Remote**.

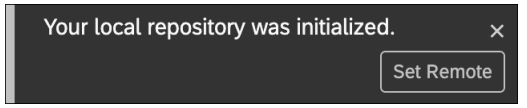

Abbildung 4.6 Meldung über das initialisierte lokale Repository in der SAP Web IDE

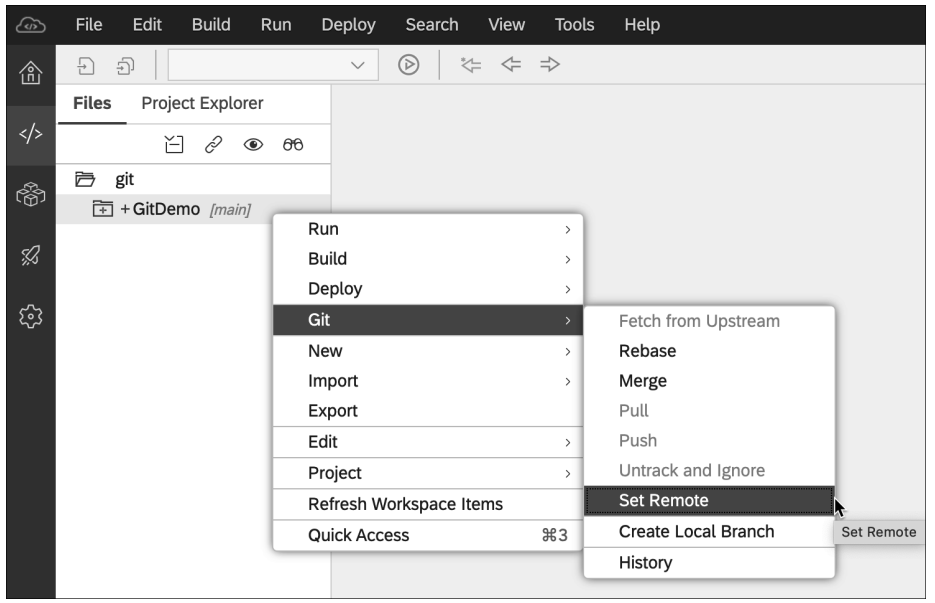

Abbildung 4.7 Remote-Repository in der SAP Web IDE aufsetzen

Nach einem Klick auf **Set Remote** erscheint ein Dialog, in dem Sie die URL zum Remote-Repository angeben. In unserem Fall wählen wir die URL des zuvor angelegten Remote-Repositorys (siehe Abbildung 4.8).

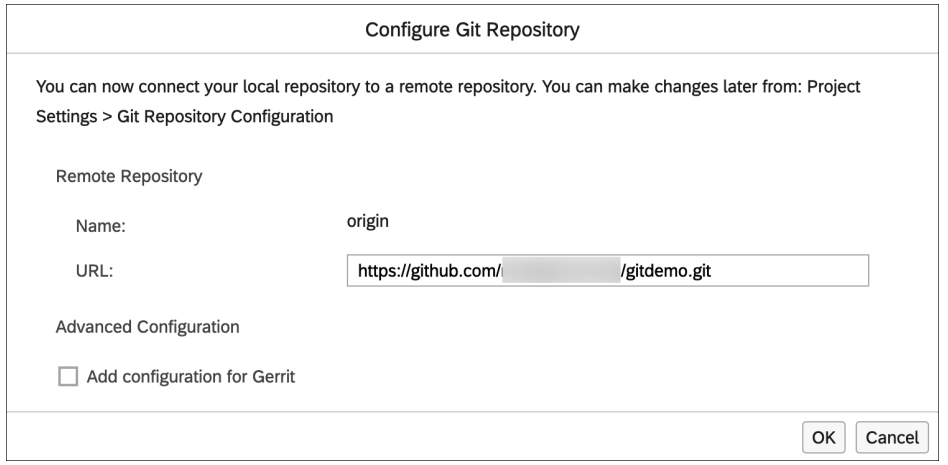

Abbildung 4.8 Git-Repository konfigurieren

Nach dem Einfügen der URL bestätigen Sie mit **OK**. Es erscheint ein weiterer Dialog, der ebenfalls mit **OK** zu bestätigen ist. Ist dies erledigt, ist es der nächste Schritt, falls noch nicht geschehen, einen Remote-Branch anzulegen.

Branch erstellen

Dazu navigieren Sie über den Menüpfad **Git • Create Remote Branch** (siehe Abbildung 4.9). Anschließend vergeben Sie im Dialog einen beliebigen Namen für den zu erstellenden Branch. Ein typischer Name für den Haupt-Branch ist beispielsweise »main«.

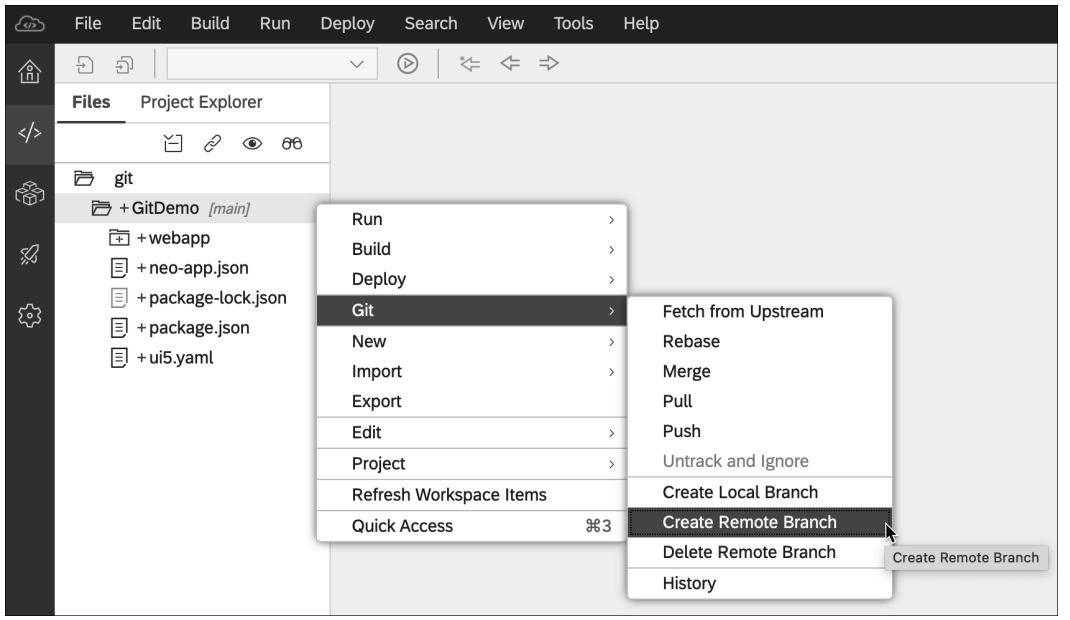

Abbildung 4.9 Remote-Branch anlegen

Nach der Eingabe eines Namens bestätigten Sie diesen Schritt wieder mit **Create** (siehe Abbildung 4.10). In unserem Fall korrelieren die Namen des lokalen und des Remote-Branch. Dies ist nicht zwingend erforderlich. Die Namen der lokalen Branches können durchaus von denen der Remote-Branches abweichen. Allerdings ist es ratsam, hier die gleichen Bezeichnungen zu verwenden, um Verwechslungen und Verwirrungen zu vermeiden.

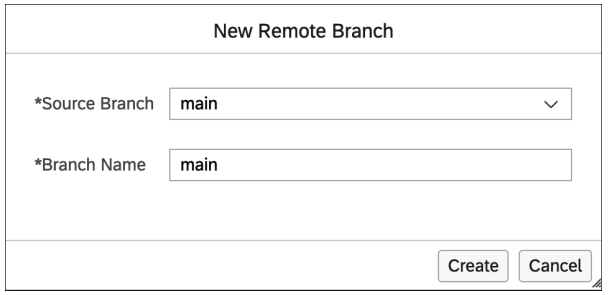

Abbildung 4.10 Der Name des Remote-Branch stimmt mit dem Namen des lokalen Branch überein.

Nachdem Sie die Anlage des Remote-Branch über die Schaltfläche **Create** bestätigt haben, wird automatisch ein Push ausgeführt. Sollten Sie bereits Änderungen in Ihrem lokales Repository vorgenommen haben, sind diese ebenfalls im Push enthalten.

Wenn Sie nun Änderungen durchführen und diese im Remote-Repository zur Verfügung stellen möchten, gehen Sie wie folgt vor:

Zunächst öffnen Sie in der rechten Menüleiste die Git Pane. Dort finden Sie die einzelnen Funktionen (beispielsweise **Commit**, **Push** und **Pull**). Diese Funktionen sind auch per Rechtsklick auf das Projekt unter dem Menüpunkt **Git** zu finden.

In dieser Ansicht finden Sie u. a. die Staging Area (näheres zur Staging Area finden Sie in Abschnitt 2.2.1 »Commit«). Hier sind alle Dateien aufgelistet, an denen seit dem letzten Commit Änderungen vorgenommen wurden. Erst durch die Auswahl einer Datei wird sie für den Commit vorselektiert. Durch einen anschließenden Commit werden die ausgewählten Änderungen in das lokale Repository in den Commit aufgenommen, und durch einen Push sind sie auch im Remote-Repository verfügbar.

Commit-Nachrichten

Wenn Sie Änderungen in einem Commit zusammenfassen, sollten Sie immer eine sprechende Commit-Nachricht verfassen. Wenn Änderungen durch Tickets ausgelöst werden, ist es oftmals hilfreich, hier die entsprechende Ticketnummer anzugeben.

Dateien vergleichen

Wenn Sie sich vor einem Commit die Änderungen an den Dateien ansehen möchten, können Sie über die Staging Area eine Vergleichssicht der Datei öffnen. In dieser sehen Sie die Datei ohne Änderungen und die Datei mit Änderungen (siehe Abbildung 4.11).

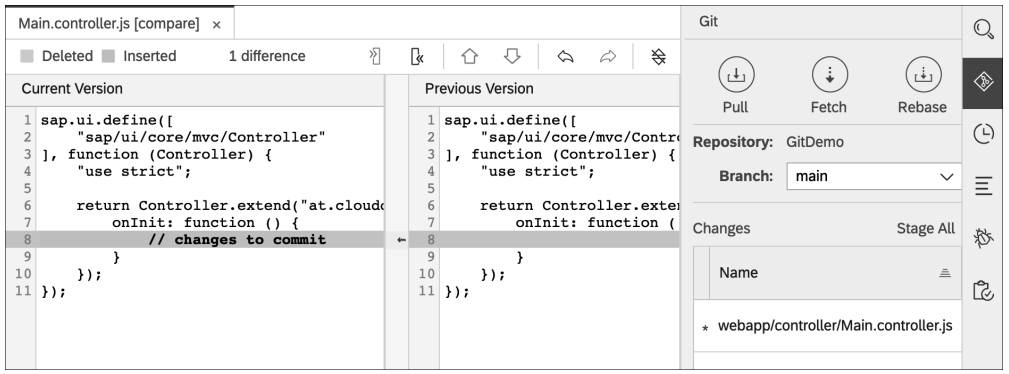

Abbildung 4.11 Dateien vor dem Commit vergleichen

Nachdem Sie die gewünschten Dateien zum Commit hinzugefügt und eine passende Commit-Nachricht verfasst haben, werden die durchgeführten Änderungen in einen Commit »verpackt«, und es wird ein Push durchgeführt (siehe Abbildung 4.12).

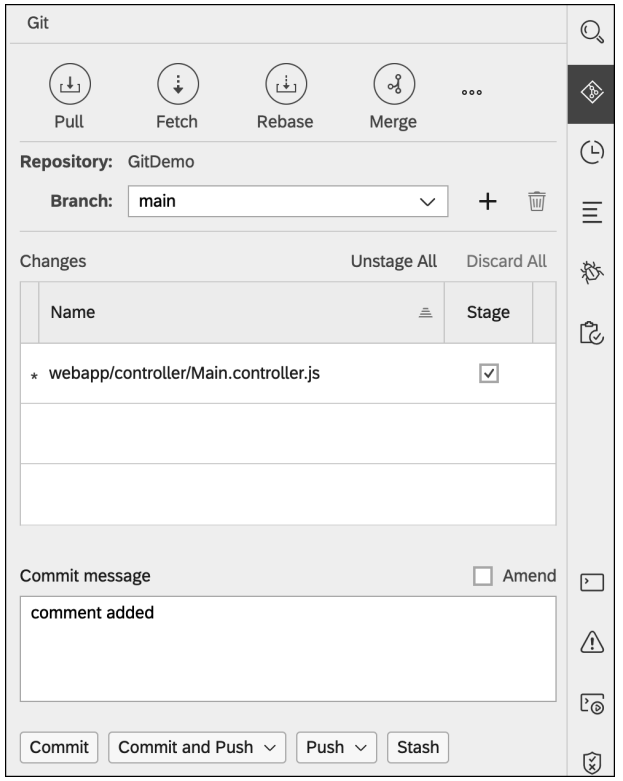

Abbildung 4.12 Commit und Push der Änderungen

Pull Request

Nach erfolgreicher Ausführung des Pushs sind die Änderungen im Remote-Repository verfügbar. Da Sie nun das Initialisieren des lokalen Repositorys, das Setzen des Remote-Repositorys und das Pushen von Änderungen kennen, fehlt noch ein wichtiger Punkt: das Durchführen eines Pull Requests. Es ist durchaus üblich, dass mehrere Personen an einem Projekt arbeiten. Um die Änderungen Ihrer Teammitglieder in Ihr lokales Repository zu transportieren, ist ein Pull Request erforderlich.

Wird ein Pull Request durchgeführt und wurden die gleichen Dateien bearbeitet, kann es auch zu einem Merge-Konflikt kommen. Dieser Konflikt muss manuell gelöst werden, indem bestätigt wird, welche Änderungen übernommen werden sollen. Wie dies in der SAP Web IDE aussieht, sehen Sie in den nächsten Schritten. Detailliertere Informationen zu Konflikten finden Sie in Abschnitt 2.5, »Konflikte lösen«.

Konflikte lösen

Zunächst arbeiten Sie lokal an Ihren Dateien und führen dann einen Commit und einen Push durch. Da es zur gleichen Datei im Remote-Repository allerdings bereits Änderungen gibt, die Sie mit einem Pull Request noch nicht in Ihr lokales Repository geladen haben, wird es beim Push zu einem Fehler kommen. Deshalb ist es erforderlich, den aktuellen Stand des Remote-Repositorys zunächst in Ihr lokales Repository zu laden. Dies geschieht mit einem Pull Request.

Mit einem Klick auf **Pull** bzw. über den Menüpfad **Git • Pull** führen Sie ein Pull Request durch. Es erscheint ein Dialog, der Sie darauf hinweist, dass Konflikte existieren (siehe Abbildung 4.13).

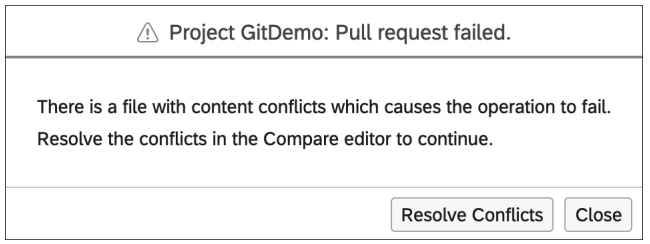

Abbildung 4.13 Merge-Konflikt in der SAP Web IDE

Mit einem Klick auf **Resolve Conflicts** öffnet sich eine Vergleichssicht der Datei, die den Konflikt verursacht. Sie sehen eine Gegenüberstellung der verschiedenen Dateistände: Links sehen Sie den aktuellen Stand der Datei, in der Mitte die Version der Datei nach dem Merge und rechts den Stand der Datei vor dem Merge (siehe Abbildung 4.14). Über die Schaltflächen oben rechts bestimmen Sie, ob die eigenen Änderungen oder Änderungen aus dem Remote-Repository akzeptiert werden sollen. Dies können Sie entweder generell für das gesamte Dokument oder nur für einzelne Änderungen entscheiden.

Abbildung 4.14 Lösen von Merge-Konflikten

Nachdem Sie die Änderungen gespeichert haben, kann die Datei wieder zu einem Commit hinzugefügt und anschließend gepusht werden. In unserem Fall wurde der Konflikt durch eine einzige Datei ausgelöst. Es kann allerdings auch vorkommen, dass mehrere Dateien Konflikte verursachen. Dann müssen Sie natürlich jede Datei prüfen, bevor die Änderungen wieder gepusht werden können.

Konflikte vermeiden

Natürlich ist es nicht möglich, Konflikte zur Gänze zu vermeiden. Wir empfehlen aber, immer einen Pull Request durchzuführen, bevor Sie lokale Änderungen vornehmen, um sicherzustellen, dass Sie mit dem aktuellen Stand arbeiten. So ist die Wahrscheinlichkeit, dass es zu einem Konflikt kommt, zumindest etwas geringer.

Git History

Neben der Git Pane, die sämtliche Funktionen wie Pull, Push, Commit, Stash usw. beinhaltet, finden Sie in der SAP Web IDE eine weitere nützliche Sicht: die sogenannte *Git History*.

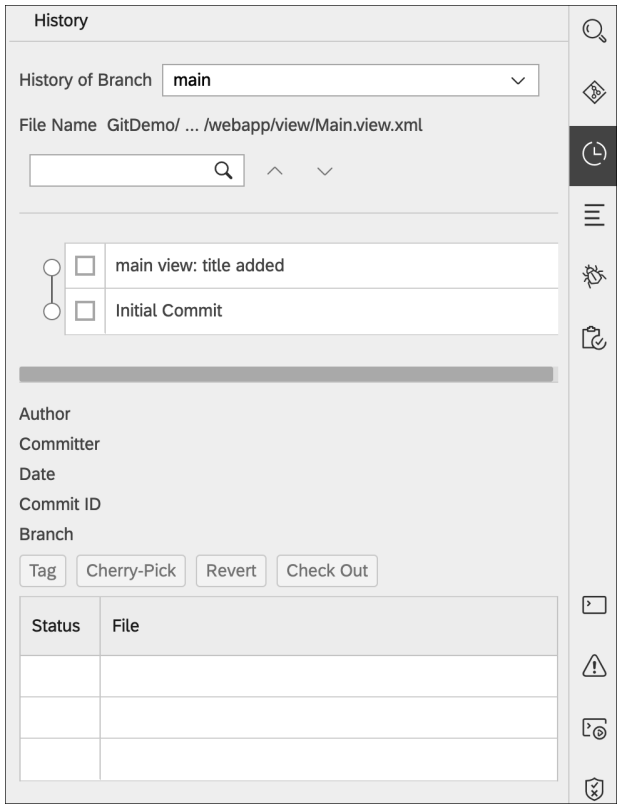

Abbildung 4.15 Git History in der SAP Web IDE

Wie in Abbildung 4.15 erkennbar, erhalten Sie dort eine Übersicht über alle durchgeführten Commits. Diese sind nach dem Zeitpunkt des Commits absteigend sortiert. Durch die Auswahl eines Commits erhalten Sie weitere Informationen wie:

- Autor (Name und E-Mail-Adresse)
- Commiter (Name und E-Mail-Adresse)
- Zeitpunkt des Commits
- Prüfsumme des Commits
- Branch, auf den der Commit durchgeführt wurde

Der Autor und der Commiter sind zumeist die gleiche Person. Allerdings kann es vorkommen, dass es sich um zwei unterschiedliche Personen handelt. Der Autor ist die Person, die den Code geschrieben hat, und der Commiter ist jene Person, die den Code in einen Commit verpackt hat.

Neben diesen Metadaten erhalten Sie eine Auflistung der Dateien, die in diesem Commit bearbeitet wurden. Auch hier können Sie per Auswahl einer Datei eine Vergleichssicht öffnen, um die durchgeführten Änderungen zu identifizieren. Diese Übersicht über die Commits bringt aber noch weitere nützliche Funktionen mit sich. So können Sie vergangene Commits erneut auschecken, um beispielsweise Hotfixes durchzuführen. Außerdem können Sie Commits mit Tags versehen und einen Cherry-Pick oder einen Revert durchführen (siehe Abschnitt 2.6.6, »Revert«).

4.3 Einbindung in SAP Business Application Studio

In diesem Abschnitt widmen wir uns den Versionsverwaltungswerkzeugen in SAP Business Application Studio. Ähnlich wie die SAP Web IDE bietet auch diese IDE eine grafische Oberfläche, in der alle relevanten Funktionen per Knopfdruck durchgeführt werden.

In SAP Business Application Studio gibt es standardmäßig zwei Sichten, die Git-relevante Funktionen abdecken. Diese Sichten finden Sie in den Reitern **Source Control** bzw. **GitLens** in der linken Menüleiste.

Analog z. B. zu Abschnitt 4.2, »Einbindung in die SAP Web IDE«, verwenden wir wieder GitHub. Das dazugehörige Remote-Repository wurde bereits, wie in Abschnitt 4.1, »Erstellen eines Git-Repositorys«, beispielhaft durchgeführt, angelegt. Nun gilt es, innerhalb von SAP Business Application Studio ein lokales Repository für das Projekt anzulegen und das bereits angelegte Remote-Repository entsprechend zu setzen. Um in SAP Business Application Studio das lokale Repository zu initialisieren, wählen Sie, nachdem Sie

die jeweilige Anwendung geöffnet haben, den Reiter **Source Control**. Jetzt können Sie per Klick auf die Schaltfläche **+** das lokale Repository initialisieren (siehe Abbildung 4.16). Eine weitere Möglichkeit wäre, über den Menüpfad **View • Find Command** nach dem passenden Kommando zu suchen und es auszuführen.

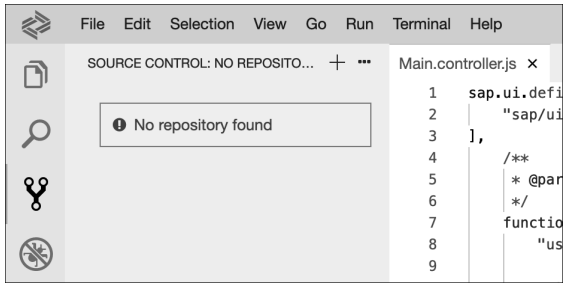

Abbildung 4.16 Das lokale Repository in SAP Business Application Studio initialisieren

Lokales Repository initialisieren

Nach einem Klick auf **Initialize Repository** ändert sich der Inhalt der Ansicht. Sofort werden sämtliche Dateien unter **Changes** aufgelistet. Nun können diese in die Staging Area aufgenommen werden, um sie später in einen Commit zu packen. Dies können Sie entweder generell für alle Elemente unter **Changes** oder für einzelne Dateien durchführen. Nachdem die Änderungen in die Staging Area überführt worden sind, finden Sie sie nicht mehr unter **Changes**, sondern unter **Staged Changes** (siehe Abbildung 4.17).

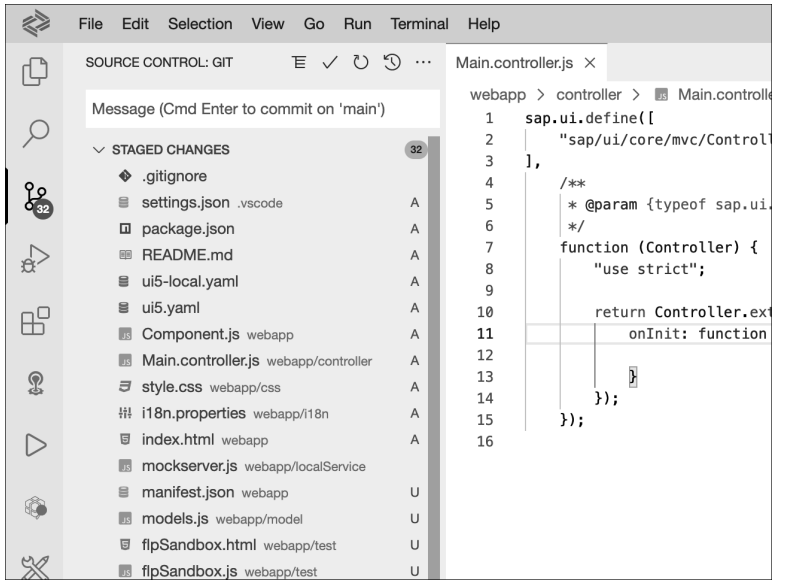

Abbildung 4.17 Staged Changes in SAP Business Application Studio

Remote-Repository setzen

Bevor Sie einen Push Request in das Remote-Repository durchführen können, muss dieses noch gesetzt werden: Wie Sie in Abbildung 4.18 sehen, finden Sie über den Pfad **More Actions • Remote • Add Remote...** die Möglichkeit dazu. Auch diese Option ist wieder unter **Find Command** zu finden.

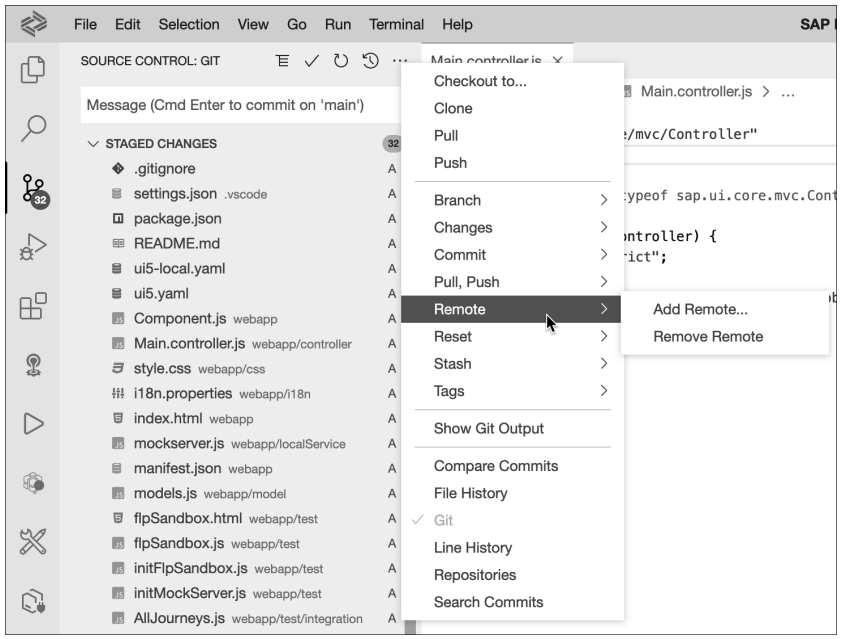

Abbildung 4.18 Remote-Repository hinzufügen

Nach der Auswahl der Option geben Sie im folgenden Eingabefeld die URL zum Remote-Repository ein (siehe Abbildung 4.19).

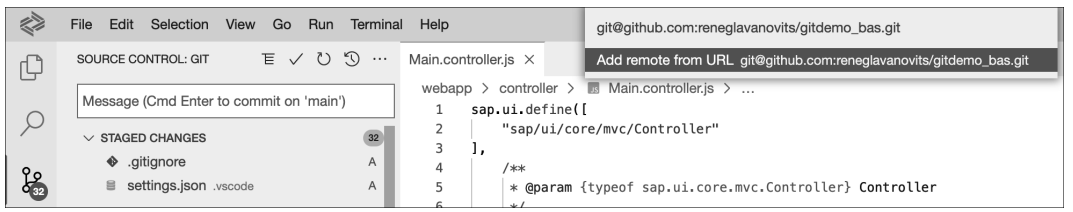

Abbildung 4.19 URL des Remote-Repositorys setzen

Remote-Branch erstellen

Außerdem werden Sie aufgefordert, einen Namen für den Remote-Branch anzugeben (siehe Abbildung 4.20). In unserem Fall vergeben wir den Namen »main«. Danach müssen Sie noch Ihre Zugangsdaten angeben.

Nachdem das lokale Repository initialisiert und das Remote-Repository hinzugefügt worden ist, können Sie den ersten Push durchführen.

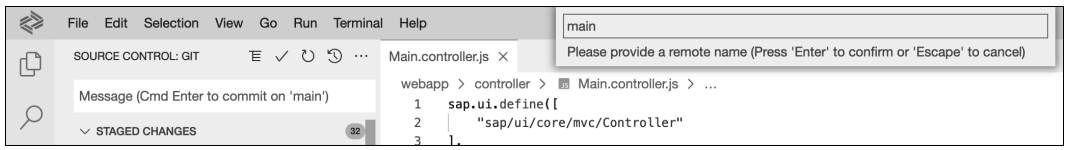

Abbildung 4.20 Den Remote-Branch in SAP Business Application Studio angeben

Dazu fügen Sie die Dateien, die sich in der Staging Area befinden, einem Commit hinzu. Anschließend führen Sie einen Push aus. Wie Sie in Abbildung 4.21 sehen, sind beide Optionen im Kontextmenü zu finden.

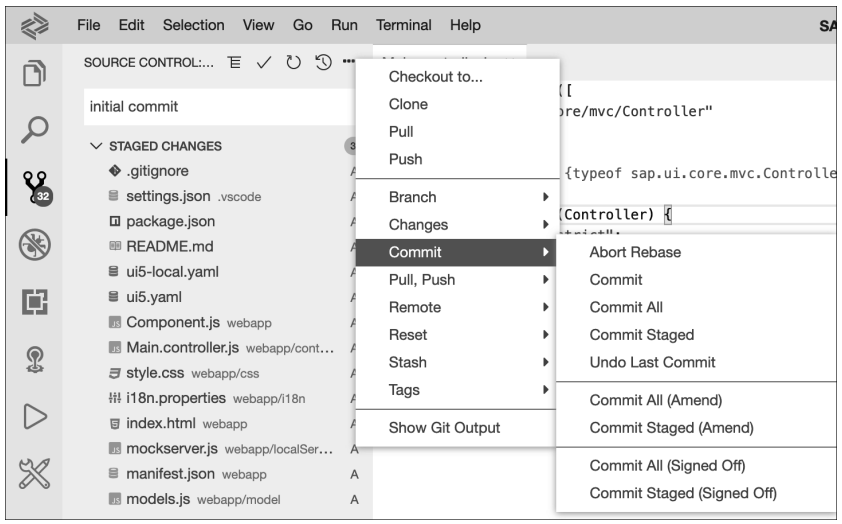

Abbildung 4.21 Commit in SAP Business Application Studio durchführen

Das Projekt ist nun im Remote-Repository verfügbar. Neben den Aktionen aus dem Kontextmenü und den Funktionen aus dem Punkt **Find Command** können Sie die Git-Kommandos alternativ direkt in der Kommandozeile ausführen.

Konfliktlösung

Natürlich muss das Lösen von Merge-Konflikten auch in SAP Business Application Studio manuell durchgeführt werden. Verfassen Sie zu Testzwecken einen Commit, in dem eine Datei bearbeitet wurde, die im Remote-Repository ebenfalls verändert wurde. Nach dem Ausführen des Commits und des Push Requests erscheint eine Fehlermeldung (siehe Abbildung 4.22).

Wie die Fehlermeldung schon sagt, muss vor dem Push ein Pull Request durchgeführt werden, um die Änderungen zu mergen. Wie Sie in Abbildung 4.23 sehen, unterscheidet sich diese Ansicht zur Lösung von Konflikten von der Ansicht der SAP Web IDE.

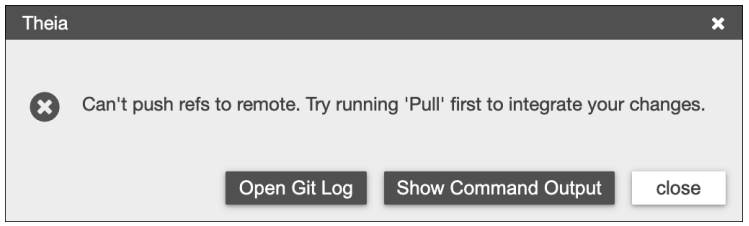

Abbildung 4.22 Merge-Konflikt in SAP Business Application Studio

Farblich unterschiedlich werden jeweils die aktuellen, aber auch die alten Codeänderungen angezeigt. Zusätzlich zu dieser Kennzeichnung steht es Ihnen frei, welche Änderungen Sie übernehmen. So können Sie entweder die aktuellen, die eingehenden oder aber auch beide Änderungen akzeptieren.

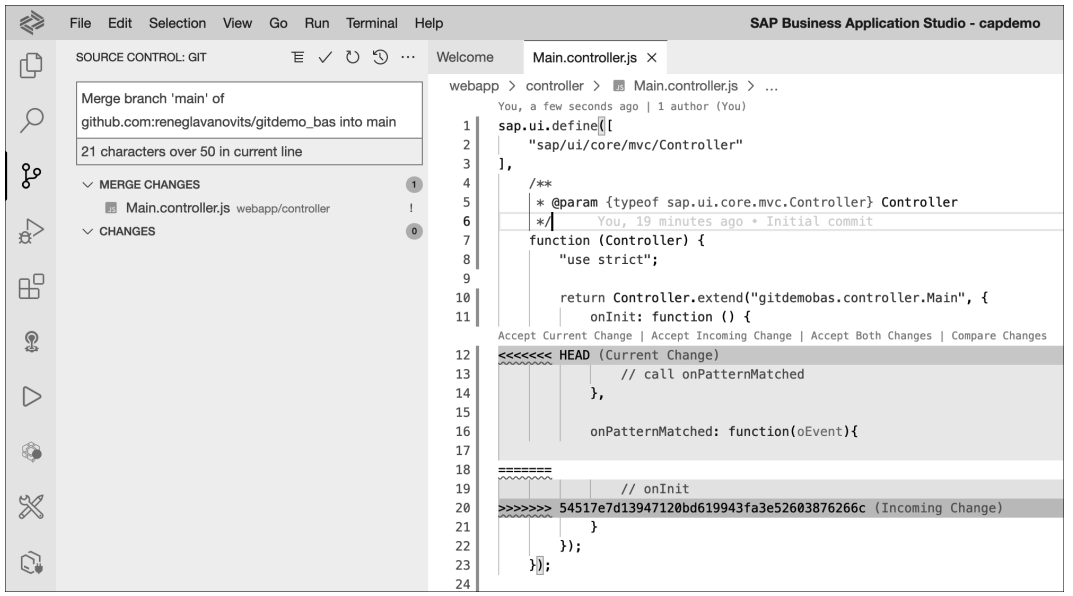

Abbildung 4.23 Konfliktlösung in SAP Business Application Studio

Mit **Compare Changes** finden Sie hier außerdem eine Aktion, mit der die Änderungen etwas übersichtlicher gegenübergestellt werden können (siehe Abbildung 4.24).

Nachdem der Konflikt entsprechend gelöst worden ist, müssen erneut ein Commit und ein Push Request ausgeführt werden. Danach sind das lokale und das Remote-Repository wieder auf dem gleichem Stand.

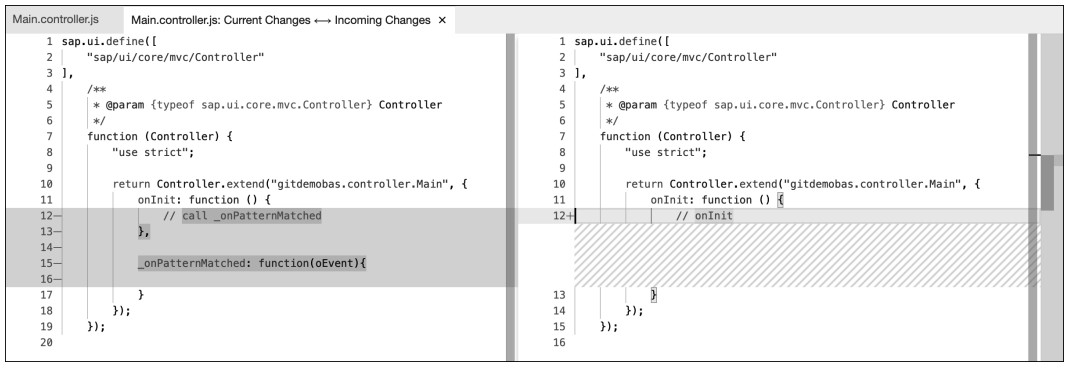

Abbildung 4.24 Gegenüberstellung zur Konfliktlösung

4.3.1 Git History

Ähnlich zur Historie in der SAP Web IDE weist auch das SAP Business Application Studio eine ähnliche Ansicht auf. Über die Schaltfläche erhalten Sie eine Übersicht über sämtliche durchgeführten Commits. Hier finden Sie folgende Funktionen (siehe Abbildung 4.25):

- Soft und Hard Reset auf einen bestimmten Commit durchführen
- einen Commit mit einem Tag versehen
- einen neuen Branch, ausgehend von einem spezifischen Commit, anlegen
- verschiedene weitere Funktionen wie Cherry-Pick, Checkout, Vergleich mehrerer Commits usw.

Zusätzlich können Sie über ein Suchfeld und verschiedene Auswahlboxen (Branch, Autor) nach bestimmten Commits suchen.

Soft und Hard Reset

Um Änderungen rückgängig zu machen, kann ein sogenannter *Reset* durchgeführt werden. Zusätzlich kann angegeben werden, ob ein Soft Reset oder ein Hard Reset vorgenommen werden soll.

Wird ein Soft Reset durchgeführt, werden die Änderungen aus dem Commit entfernt. Das heißt, dass die Änderungen aus dem Commit wieder zurück in die Staging Area wandern und somit nicht verloren sind.

Bei einem Hard Reset werden hingegen die Änderungen aus dem Commit und aus der Staging Area entfernt und letztendlich auch endgültig gelöscht. Somit sind die Änderungen nach einem Hard Reset nicht mehr verfügbar.

Abbildung 4.25 Git History in SAP Business Application Studio

Änderungen eines Commits

Um sich die Änderungen, die im Zuge eines Commits implementiert wurden, im Detail anzusehen, müssen Sie den entsprechenden Commit selektieren. Daraufhin wird Ihnen eine Liste der Änderungen angezeigt (siehe Abbildung 4.26).

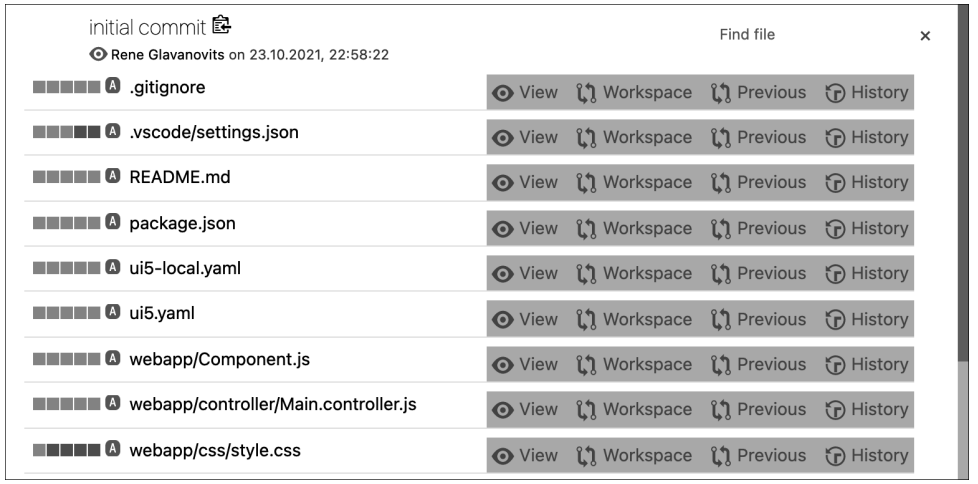

Abbildung 4.26 Detaillierte Git History

Neben den Metadaten (Autor, Zeitpunkt des Commits) sehen Sie u. a. sämtliche Dateien, die von diesem Commit betroffen sind. Zudem gibt es zu jeder Datei verschiedene Funktionen: Sie können sich z. B. den Inhalt der Datei anzeigen lassen oder eine Vergleichssicht mit der Datei aus dem Workspace und der Datei aus dem vorherigen Commit öffnen. Wenn Sie wissen möchten, in welchen Commits eine spezifische Datei vorkommt, finden Sie es über diese Sicht heraus.

4.3.2 GitLens

Neben dem Reiter **Source Control** finden Sie in der Menüleiste auf der linken Seite auch die sogenannte *GitLens*. GitLens ist eine Open-Source-Erweiterung für Visual Studio Code, die standardmäßig mit SAP Business Application Studio ausgeliefert wird. Sie vereinfacht den genaueren Einblick in die unterschiedlichen Versionen von Codeständen. Sie können Metadaten, wie z. B. die an einem Repository beteiligten Autoren auslesen. In der geöffneten Datei wird der Zeitpunkt des letzten Commits und der Name der bearbeitenden Person angezeigt. Diese Informationen werden am Anfang der Datei und in der Zeile, in der Sie sich mit dem Cursor befinden, angezeigt. Für diese Information gibt es in dem Reiter **GitLens** auch einen eigenen Unterpunkt.

Diese Erweiterung bringt eine Sicht mit sich, die basierend auf der aktuell geöffneten Datei und der ausgewählten Zeile, alle Commits anzeigt, die diese Datei bzw. diese Zeile betreffen. Wie aus Abbildung 4.27 ersichtlich, werden Ihnen hier die Commit-Nachricht, der Autor und der Zeitpunkt des entsprechenden Commits aufgelistet. Außerdem erhalten Sie die Möglichkeit, den Stand der Datei aus einem alten Commit zu öffnen, um diesen mit der aktuellen Version zu vergleichen.

Abbildung 4.27 Commit-Historie über GitLens

Mit dieser Erweiterung können Sie auch nach bestimmten Commits suchen. Suchkriterien können der Autor, die Prüfsumme (auch Commit ID genannt), geänderte Dateien oder auch die Commit-Nachricht sein. Diese Funktion ist besonders dann von Nutzen, wenn es darum geht, Fehler in bereits gepushten Commits zu identifizieren. In Verbindung mit der Suchfunktion kann auch die Funktion zum Vergleichen einzelner Commits nützlich sein. So können Sie nach der Identifikation eines fehlerbehafteten Commits feststellen, wodurch dieser Fehler aufgetreten ist, indem Sie ihn mit einem Commit vergleichen, der keinen Fehler hat. Wie die beiden Sichten in GitLens aussehen, sehen Sie in Abbildung 4.28.

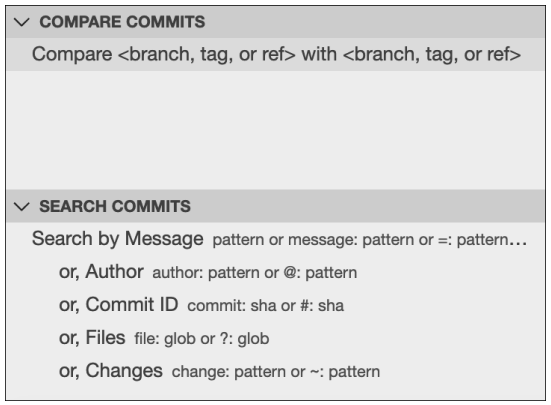

Abbildung 4.28 Commit-Suche und Commit-Vergleich mit GitLens

Die Erweiterung bietet noch einen weiteren Reiter mit der Bezeichnung **Repositories** (siehe Abbildung 4.29).

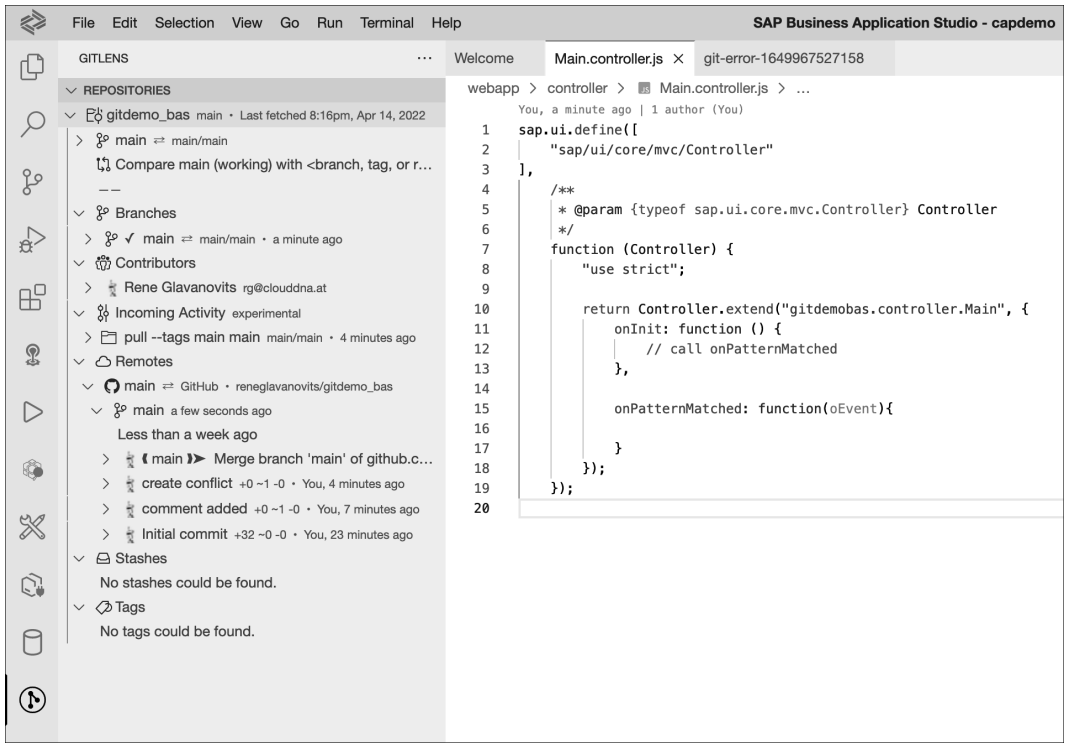

Abbildung 4.29 Reiter »Repositories«

Darauf finden Sie verschiedene allgemeine Informationen zum Repository. So erhalten Sie u. a. einen Überblick über die beteiligten Autor*innen und können ermitteln, welche Commits durch welche Personen erfolgt sind. Des Weiteren sehen Sie unter **Remotes** bzw. **Branches** die lokalen und remotefähigen Branches zu diesem Repository. Auch für durchgeführte Stashes und hinzugefügte Tags finden Sie hier Unterpunkte, unter denen diese auflistet sind.

In diesem Reiter können Sie u. a. die folgenden Aktionen durchführen:

- Pull Requests ausführen
- Push Requests ausführen
- Tags erstellen
- Branches erstellen und zwischen diesen wechseln
- zu früheren Commits zurückspringen
- verschiedene Commits vergleichen

4.4 Einbindung in Visual Studio Code

Nachdem wir in den beiden vorangegangenen Abschnitten die Verwendung von Git in den beiden Entwicklungsumgebungen SAP Web IDE und SAP Business Application Studio betrachtet haben, widmen wir uns in diesem Abschnitt Visual Studio Code. Visual Studio Code ist eine gute Alternative, um lokal SAPUI5 zu entwickeln. Diese IDE eignet sich nicht nur für die SAPUI5-Entwicklung, sondern kann auch für alle möglichen anderen Entwicklungen abseits von SAP verwendet werden. Außerdem sind für Visual Studio Code durchaus viele verschiedene Erweiterungen verfügbar. Da sowohl Visual Studio Code als auch SAP Business Application Studio auf dem Framework Eclipse Theia basieren, ähneln sich diese IDEs in ihrer Erscheinung sehr stark. Nichtsdestotrotz betrachten wir in diesem Abschnitt die Verwendung von Git in Visual Studio Code im Detail.

Auch in diesem Abschnitt verwenden wir wieder ein Repository, das analog zu dem Repository aus Abschnitt 4.1 erstellt wurde. Auch die verwendete SAPUI5-Anwendung wurde bereits erstellt.

Lokales Repository initialisieren

Da das Remote-Repository bereits existiert, ist der nächste Schritt, ein lokales Repository in Visual Studio Code anzulegen und das Remote-Repository entsprechend zu setzen. Dazu wechseln Sie zunächst in den Reiter **Source Control** – erkennbar am Icon – In diesem Reiter wählen Sie die Schaltfläche **Initialize Repository** (siehe Abbildung 4.30).

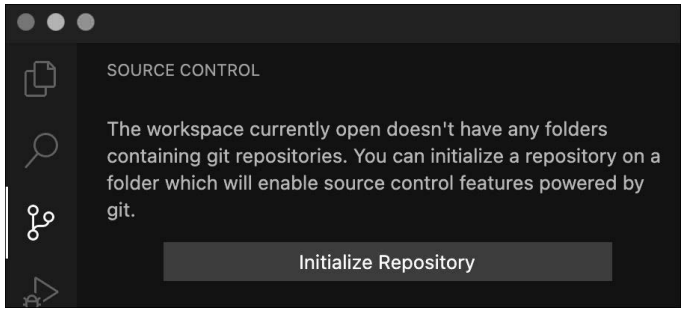

Abbildung 4.30 Initialisieren des Repositorys in Visual Studio Code

Verzeichnisauswahl

Nach einem Klick auf die Schaltfläche muss im folgenden Eingabefeld der Projektordner ausgewählt werden (siehe Abbildung 4.31). Diese Auswahl sagt aus, in welchem Verzeichnis das Repository initialisiert wird. Wählen Sie das Projektverzeichnis aus, das als Vorschlagswert erscheint.

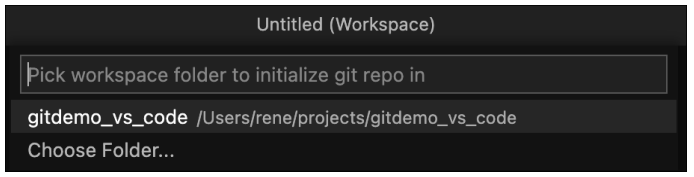

Abbildung 4.31 Ordner für die Initialisierung des Repositorys auswählen

Nachdem das entsprechende Verzeichnis ausgewählt worden ist, sehen Sie die Auswirkungen auf den Reiter **Source Control**. Einerseits wird im Icon durch eine Zahl gekennzeichnet, wie viele Dateien von Änderungen betroffen sind. Andererseits sehen Sie in der Liste die betreffenden Dateien (siehe Abbildung 4.32).

Remote-Repository hinzufügen

Nun ist zwar das lokale Repository initialisiert, aber um die Änderungen auch in das Remote-Repository zu übertragen, muss es zunächst hinzugefügt werden. In dem Reiter **Source Control** öffnen Sie über die Schaltfläche ··· ein Kontextmenü. Wählen Sie den Menüpfad **Remote • Add Remote**, um die Adresse zum Remote-Repository anzugeben. Geben Sie anschließend im Eingabefeld, das im mittleren, oberen Bereich erscheint, die URL zum Repository an (siehe Abbildung 4.33).

Ist dieser Schritt abgeschlossen, muss noch ein Name für das Remote-Repository vergeben werden. Wie bereits zuvor erscheint ein Eingabefeld (siehe Abbildung 4.34). Wir tippen hier den Namen »main«, als Bezeichner für das Remote-Repository ein und bestätigen die Eingabe mit ↵.

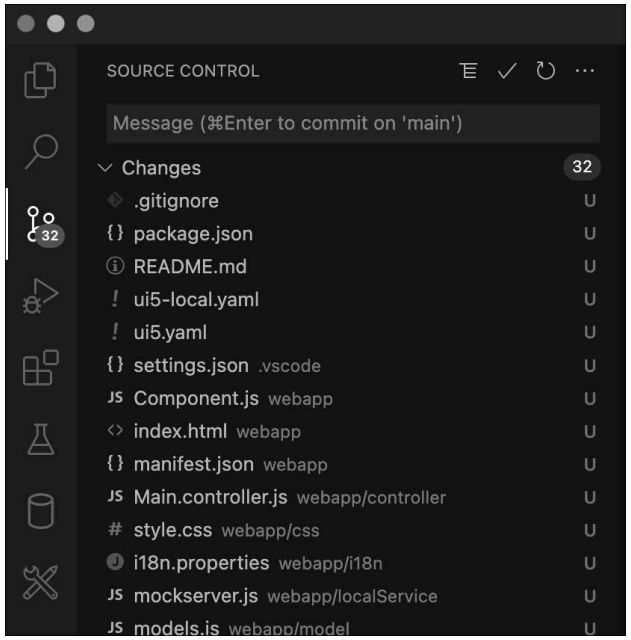

Abbildung 4.32 Source Control nach dem Initialisieren des Repositorys

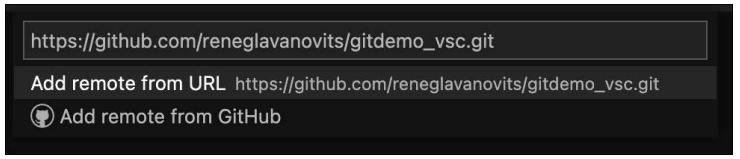

Abbildung 4.33 Remote-Repository in Visual Studio Code hinzufügen

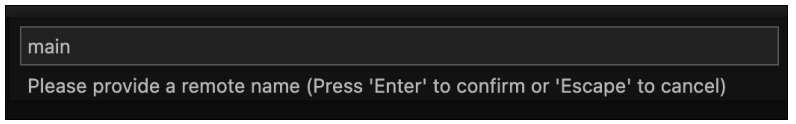

Abbildung 4.34 Namen für das Remote-Repository angeben

Nachdem nun dieser Schritt abgeschlossen ist, müssen gegebenenfalls noch Anmeldeinformationen eingegeben werden. Anschließend ist das Remote-Repository entsprechend gesetzt, und Sie können die notwendigen Push und Pull Requests durchführen.

Initialer Commit

Zu diesem Zeitpunkt ist die SAPUI5-Applikation weder im lokalen noch im Remote-Repository vorhanden. Um die Applikation zu übertragen, muss ein Commit, gefolgt von einem Push, durchgeführt werden. Wählen Sie dazu alle relevanten Dateien in dem Reiter **Source Control** aus. Da dies in un-

serem Fall der initiale Commit ist und wir die gesamte Applikation in das Repository übertragen möchten, wählen wir hier alle verfügbaren Dateien aus. Die betreffenden Dateien können Sie einzeln über die Schaltfläche + neben der jeweiligen Zeile für einen Commit vorselektieren.

Sie können auch alle Dateien, die von den Änderungen betroffen sind, auf einmal über die Schaltfläche in der Zeile **Changes** auswählen (siehe Abbildung 4.35).

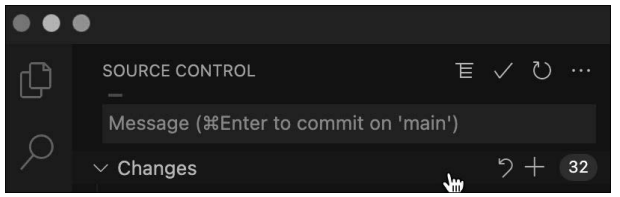

Abbildung 4.35 Alle Änderungen in den Commit aufnehmen

Staging Area

In dem Reiter **Source Control** sehen Sie nun, dass alle ausgewählten Dateien nicht mehr im Bereich **Changes** vorhanden sind. Stattdessen befinden sich diese Einträge nun im Bereich **Staged Changes** (siehe Abbildung 4.36). Wird ein Commit durchgeführt, werden alle Dateien, die sich in diesem Bereich befinden, mit in den Commit aufgenommen.

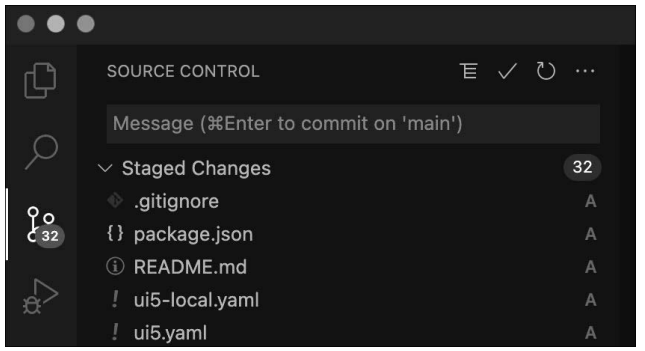

Abbildung 4.36 Staged Changes in Visual Studio Code

Commit-Nachricht eingeben

Geben Sie nun noch eine möglichst sprechende Commit-Nachricht ein. Da dies in unserem Fall der erste Commit ist, geben wir als Nachricht »Initial Commit« an. Die Nachricht wird wie in Abbildung 4.37 direkt in das dafür vorgesehene Eingabefeld eingetragen.

Nachdem die Nachricht eingetragen worden ist, wird der Commit im Kontextmenü über den Menüpfad **Commit • Commit Staged** durchgeführt (siehe Abbildung 4.38). Dadurch werden alle zuvor selektierten Einträge mit der eingegebenen Nachricht dem Commit hinzugefügt.

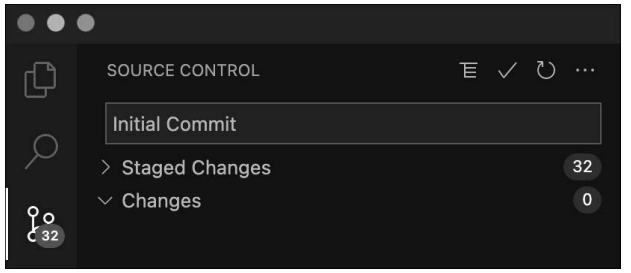

Abbildung 4.37 Commit-Nachricht in Visual Studio Code eintragen

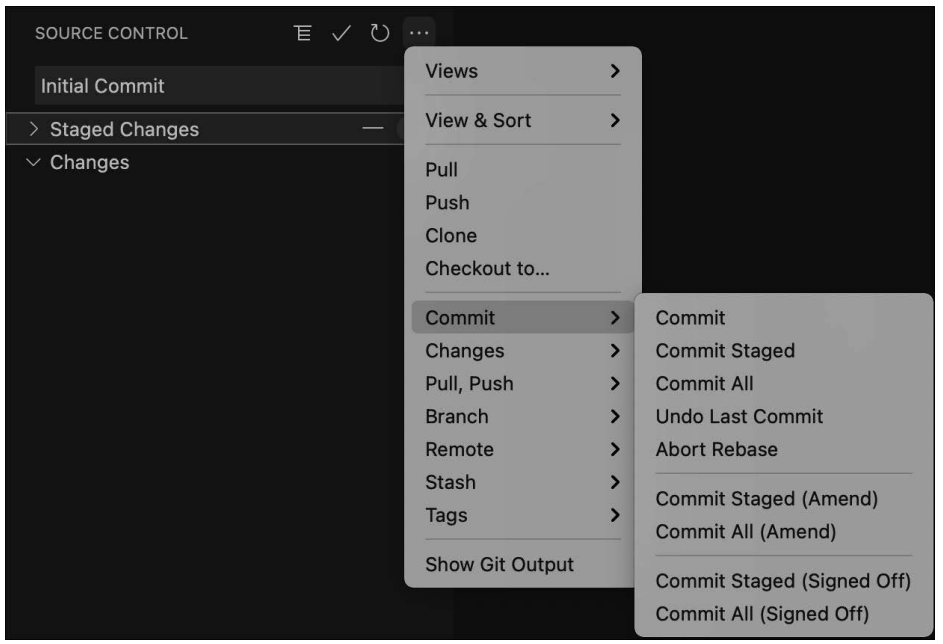

Abbildung 4.38 Commit in Visual Studio Code durchführen

Nach erfolgreicher Durchführung des Commits sehen Sie in dem Reiter **Source Control**, dass es keine Unterteilung in verschiedene Sektionen mehr gibt. Über die Schaltfläche **Publish Branch** transferieren Sie durchgeführte Commits in das Remote-Repository. Alternativ können Sie für diesen Push Request auch im Kontextmenü über den Menüpunkt **Push** verwenden (siehe Abbildung 4.39).

Nach erfolgreicher Durchführung des Requests finden Sie in dem Reiter **Source Control** wieder die gleiche Ansicht wie vor der Durchführung des Commits vor (siehe Abbildung 4.40).

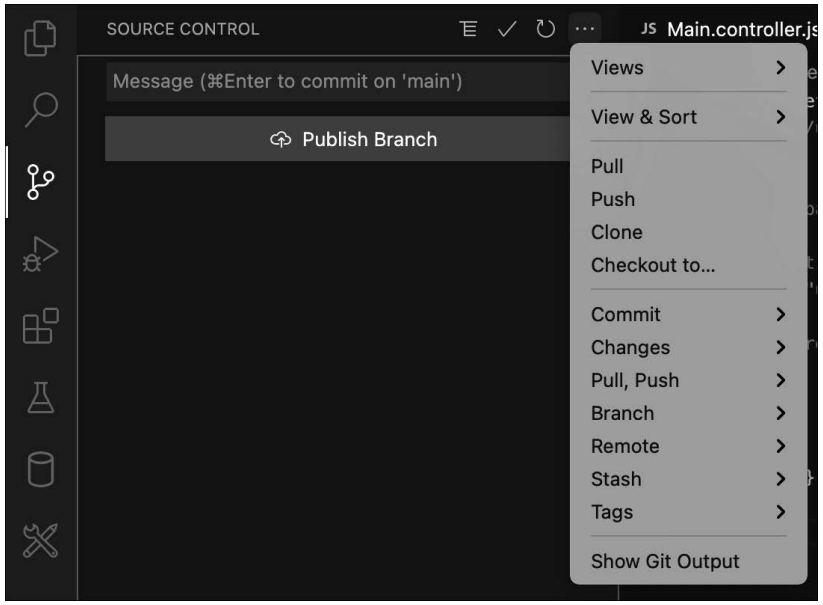

Abbildung 4.39 Push Requests in Visual Studio Code durchführen

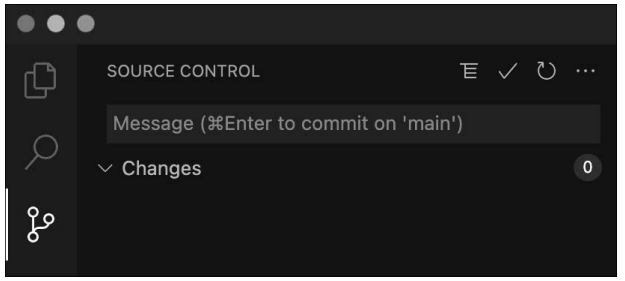

Abbildung 4.40 Ansicht der Source Control nach dem Push Request

Zudem ist der durchgeführte Commit und somit auch der aktuelle Datenstand im Remote-Repository vorhanden. Wenn Sie sich, wie in Abbildung 4.41 gezeigt, die Historie der Commits im Repository ansehen, erkennen Sie, dass der zuvor durchgeführte Commit mit der Nachricht »Initial Commit« existiert.

Standardmäßig bringt Visual Studio Code keine weiteren Oberflächen mit, mit denen die Git History eingesehen werden oder Git-spezifische Funktionen durchgeführt werden könnten. Allerdings können Sie solche Funktionen über verschiedene Erweiterungen hinzufügen, die Sie über den Menüpunkt **Extensions** finden. Die Vorgehensweise zur Lösung von Konflikten ist in Visual Studio Code identisch zu der Vorgehensweise in SAP Business Application Studio.

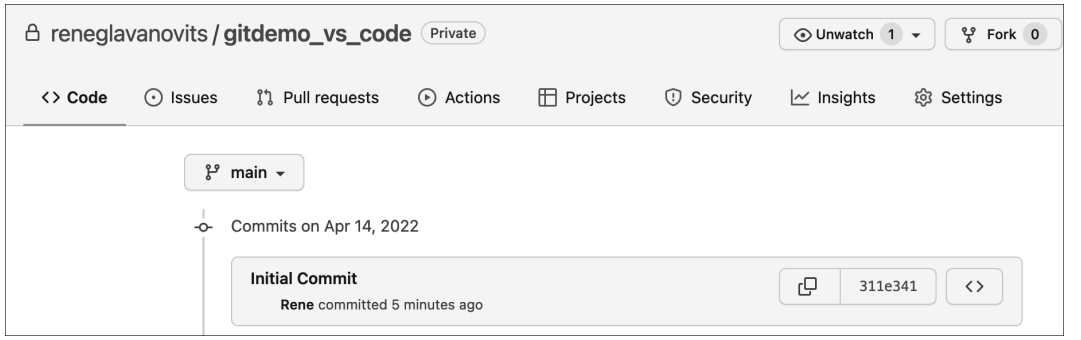

Abbildung 4.41 Übersicht der Commits im Repository

Wie Sie sehen konnten, weisen die verschiedenen Entwicklungsumgebungen doch einige Gemeinsamkeiten in der Verwendung von Git auf. Wenn auch das neuere SAP Business Application Studio eine übersichtlichere Aufbereitung der Git-relevanten Informationen bietet, bieten die anderen Entwicklungsumgebungen dennoch die gleichen Funktionalitäten.

Nachdem Sie sich in diesem Kapitel mit der Verwendung von Git in den gängigen Entwicklungsumgebungen von SAP, SAP Web IDE, SAP Business Application Studio und Visual Studio Code beschäftigt haben, werden Sie sich im nächsten Kapitel mit der Verwendung von abapGit auseinandersetzen. Dabei werden Sie von der Installation über die Einbindung bis zur Verwendung dieses Git-Clients geführt.